АО "НТЦ ИТ РОСА"

**Операционная система**

**РОСА "ХРОМ" 12**

**Основы администрирования**

РСЮК.10301-01 95 01

Листов 188

2024

АННОТАЦИЯ

Данный документ является справочно-информационным описанием основ администрирования операционной системы РОСА "ХРОМ" версии 12 РСЮК.10301-01 (далее – ОС, Система) в вариантах "рабочая станция" (далее – ОС рабочей станции) и "сервер" (далее – серверная ОС).

Данная ОС зарегистрирована в реестре российского ПО от 05.09.2016 №1607. Предполагается, что читатель является системным администратором ОС Linux с опытом работы с интерфейсом командной строки.

Простейшие административные действия с помощью графического интерфейса описаны в документе "ОС РОСА "ХРОМ". Руководство пользователя. РСЮК.10301-01 34 01".

Сведения об установке и настройке ОС описаны в документе "ОС РОСА "ХРОМ". Руководство администратора. РСЮК.10301-01 32 01".

В данном руководстве содержится дополнительная информация об основах взаимодействия с ОС РОСА "ХРОМ" на примере базовых задач администрирования.

Для разработки документа использованы ссылки на следующие стандарты:

* ГОСТ Р 2.105-2019 "Единая система конструкторской документации (ЕСКД). Общие требования к текстовым документам";
* ГОСТ 2.601 "Единая система программной документации. Виды программных документов";
* ГОСТ 19.101-77 "Единая система программной документации.
* Виды программ и программных документов";
* ГОСТ 19.105-78 "Единая система программной документации. Общие требования к программным документам";
* [ГОСТ 19.503-79](https://docs.cntd.ru/document/1200007674) "Единая система программной документации. Руководство системного программиста".

Настоящий документ подготовлен в соответствии с технологической инструкцией "РОСА. Регламент формирования документации к программным продуктам РСЮК.11001-01 90 01".

Содержание

[Содержание 3](#_Toc178240234)

[1 Общие сведения 8](#_Toc178240235)

[1.1 Общее описание 8](#_Toc178240236)

[1.2 Требования к техническим средствам 8](#_Toc178240237)

[1.3 Состав ОС 8](#_Toc178240238)

[2 Оболочка 9](#_Toc178240239)

[2.1 Общие сведения 9](#_Toc178240240)

[2.2 Эмулятор терминала 10](#_Toc178240241)

[2.3 Телетайп 11](#_Toc178240242)

[2.4 Переключение между консолями 11](#_Toc178240243)

[3 Команды 12](#_Toc178240244)

[3.1 Выполнение команд 12](#_Toc178240245)

[3.1.1 Без опций и аргументов 12](#_Toc178240246)

[3.1.2 С аргументами 12](#_Toc178240247)

[3.1.3 С опциями 12](#_Toc178240248)

[3.1.4 С опциями и аргументами 14](#_Toc178240249)

[3.2 Фоновые процессы 14](#_Toc178240250)

[3.3 Код завершения (Exit Code) 15](#_Toc178240251)

[3.4 Выполнение нескольких команд 15](#_Toc178240252)

[3.5 Работа с каталогами 17](#_Toc178240253)

[3.6 Перенос строк 18](#_Toc178240254)

[3.7 Множитель вывода 18](#_Toc178240255)

[4 Переменные окружения 20](#_Toc178240256)

[4.1 Просмотр переменных 20](#_Toc178240257)

[4.1.1 Команда env 20](#_Toc178240258)

[4.1.2 Встроенная команда set в bash 20](#_Toc178240259)

[4.2 Значения переменных 21](#_Toc178240260)

[4.3 Установка значений 21](#_Toc178240261)

[4.4 Использование справочных ресурсов 22](#_Toc178240262)

[4.4.1 Команда man 22](#_Toc178240263)

[4.4.2 Хранение страниц интерактивного руководства 23](#_Toc178240264)

[4.4.3 Команда TLDR 23](#_Toc178240265)

[5 Инструменты управления пакетами 25](#_Toc178240266)

[5.1 Высокоуровневый менеджер 25](#_Toc178240267)

[5.2 Низкоуровневый менеджер 26](#_Toc178240268)

[6 Потоки 28](#_Toc178240269)

[7 Поиск информации 31](#_Toc178240270)

[7.1 Команды which, whereis, whatis 31](#_Toc178240271)

[7.2 Команда locate 32](#_Toc178240272)

[7.3 Особенности 32](#_Toc178240273)

[7.4 Команда wc 33](#_Toc178240274)

[7.5 Команда sort 33](#_Toc178240275)

[7.6 Команда cut 35](#_Toc178240276)

[7.7 Команда find 36](#_Toc178240277)

[7.8 Команда grep 37](#_Toc178240278)

[7.9 Комбинации команд find и grep 38](#_Toc178240279)

[8 Работа с пользователями 39](#_Toc178240280)

[8.1 Общие сведения 39](#_Toc178240281)

[8.2 Создание нового пользователя 41](#_Toc178240282)

[8.3 Изменение настроек пользователя 45](#_Toc178240283)

[8.4 Удаление пользователя 46](#_Toc178240284)

[8.5 Блокировка пользователя 47](#_Toc178240285)

[9 Группы пользователей 48](#_Toc178240286)

[9.1 Общие сведения 48](#_Toc178240287)

[9.2 Создание групп 48](#_Toc178240288)

[9.3 Редактирование групп пользователей 49](#_Toc178240289)

[9.4 Удаление групп пользователей 49](#_Toc178240290)

[9.5 Управление группами 49](#_Toc178240291)

[9.6 Списки 51](#_Toc178240292)

[10 Управление учетными записями 52](#_Toc178240293)

[10.1 Команда useradd 52](#_Toc178240294)

[10.2 Команда usermod 52](#_Toc178240295)

[10.3 Команда chage 53](#_Toc178240296)

[11 Привилегии 54](#_Toc178240297)

[11.1 Общие сведения 54](#_Toc178240298)

[11.2 Суперпользователь 54](#_Toc178240299)

[11.3 Команда sudo 54](#_Toc178240300)

[11.4 Настройки файла sudoers 55](#_Toc178240301)

[11.5 Присвоение привилегий 57](#_Toc178240302)

[11.6 Создание правил в файле sudoers 58](#_Toc178240303)

[11.6.1 Группировка 58](#_Toc178240304)

[11.6.2 Свойства 58](#_Toc178240305)

[12 Аутентификация 60](#_Toc178240306)

[12.1 Файл /etc/passwd/ 60](#_Toc178240307)

[12.2 Файл /etc/shadow/ 61](#_Toc178240308)

[12.3 Права доступа 63](#_Toc178240309)

[12.4 Смена владельца 65](#_Toc178240310)

[12.5 Смена прав 66](#_Toc178240311)

[12.6 Изменение атрибутов 68](#_Toc178240312)

[12.6.1 SUID (Set User ID) 71](#_Toc178240313)

[12.6.2 SGID (Set Group ID) 71](#_Toc178240314)

[12.6.3 UMASK 71](#_Toc178240315)

[12.6.4 Изменение SUID и SGID 72](#_Toc178240316)

[13 Контроль доступа 73](#_Toc178240317)

[13.1 Общие сведения 73](#_Toc178240318)

[13.2 Списки контроля доступа 74](#_Toc178240319)

[13.2.1 Команды getfacl и setfacl 74](#_Toc178240320)

[13.2.2 Маски 75](#_Toc178240321)

[13.2.3 Наследование 75](#_Toc178240322)

[13.2.4 Совместимость 76](#_Toc178240323)

[13.3 Расширенные атрибуты 76](#_Toc178240324)

[13.3.1 Команды setfattr и getfattr 76](#_Toc178240325)

[13.3.2 Совместимость с расширенными атрибутами 77](#_Toc178240326)

[14 Процессы 78](#_Toc178240327)

[14.1 Общие сведения 78](#_Toc178240328)

[14.2 Жизненный цикл процесса 78](#_Toc178240329)

[14.3 Виды межпроцессорного обмена 81](#_Toc178240330)

[14.3.1 Локальный уровень 81](#_Toc178240331)

[14.3.2 Сигналы 81](#_Toc178240332)

[14.3.3 Удаленные 82](#_Toc178240333)

[14.4 Управление процессами 84](#_Toc178240334)

[14.4.1 Получение информации о процессе 84](#_Toc178240335)

[14.4.2 Приоритезация 86](#_Toc178240336)

[14.4.3 Команда free 88](#_Toc178240337)

[14.4.4 Команда htop 91](#_Toc178240338)

[14.4.5 Команда df 94](#_Toc178240339)

[14.4.6 Команда du 96](#_Toc178240340)

[14.4.7 Комбинации команд 99](#_Toc178240341)

[15 Каналы 100](#_Toc178240342)

[15.1 Общие сведения 100](#_Toc178240343)

[15.2 Конвейеры 101](#_Toc178240344)

[15.3 Логика 101](#_Toc178240345)

[15.4 Именованные каналы 102](#_Toc178240346)

[15.4.1 Временные именованные каналы 103](#_Toc178240347)

[15.4.2 Условия использования каналов 104](#_Toc178240348)

[15.5 Программные каналы 104](#_Toc178240349)

[16 Отладка 106](#_Toc178240350)

[16.1 Консольный отладчик GDB 106](#_Toc178240351)

[16.2 Утилита strace 107](#_Toc178240352)

[16.3 Псевдо-файловая система 108](#_Toc178240353)

[17 Файловая система 110](#_Toc178240354)

[17.1 Каталоги 110](#_Toc178240355)

[17.2 Типы файлов 110](#_Toc178240356)

[17.3 Регистр имен файлов 111](#_Toc178240357)

[18 Система инициализации 113](#_Toc178240358)

[18.1 Общие сведения 113](#_Toc178240359)

[18.2 Управление службами 113](#_Toc178240360)

[18.2.1 Запуск/остановка 113](#_Toc178240361)

[18.2.2 Перезапуск 114](#_Toc178240362)

[18.2.3 Включение/выключение 114](#_Toc178240363)

[18.2.4 Проверка статуса 115](#_Toc178240364)

[18.3 Обзор состояния Системы 116](#_Toc178240365)

[18.3.1 Текущие модули 116](#_Toc178240366)

[18.3.2 Управление модулями 116](#_Toc178240367)

[18.3.3 Цели 120](#_Toc178240368)

[19 Указатели 123](#_Toc178240369)

[19.1 Общие сведения 123](#_Toc178240370)

[19.2 Символические ссылки 123](#_Toc178240371)

[19.3 Жесткие ссылки 124](#_Toc178240372)

[19.4 Отличия символической и жесткой ссылок 125](#_Toc178240373)

[19.5 Создание ссылок 126](#_Toc178240374)

[19.6 Удаление ссылок 126](#_Toc178240375)

[19.7 Файлы-устройства 126](#_Toc178240376)

[19.8 Идентификаторы 126](#_Toc178240377)

[19.9 Дескрипторы 127](#_Toc178240378)

[19.10 Система udev 128](#_Toc178240379)

[20 Архивирование 130](#_Toc178240380)

[20.1 Общие сведения 130](#_Toc178240381)

[20.2 Утилита Tar 130](#_Toc178240382)

[20.3 Утилита Shar 132](#_Toc178240383)

[20.4 Утилита Ar 132](#_Toc178240384)

[20.5 Утилита Cpio 133](#_Toc178240385)

[21 Сжатие 134](#_Toc178240386)

[21.1 Утилита Gzip 134](#_Toc178240387)

[21.2 Утилита Bzip 134](#_Toc178240388)

[21.3 Утилита Lzma 135](#_Toc178240389)

[21.4 Утилита Xz 135](#_Toc178240390)

[21.5 Утилита Zip 135](#_Toc178240391)

[21.6 Утилита RAR 135](#_Toc178240392)

[22 Создание разделов диска 138](#_Toc178240393)

[22.1 Общие сведения 138](#_Toc178240394)

[22.2 MBR 138](#_Toc178240395)

[22.2.1 Управление разделами MBR 138](#_Toc178240396)

[22.2.2 Печать текущей таблицы разделов 139](#_Toc178240397)

[22.2.3 Первичные и расширенные разделы 140](#_Toc178240398)

[22.2.4 Создание раздела 141](#_Toc178240399)

[22.2.5 Проверка нераспределенного пространства диска 141](#_Toc178240400)

[22.2.6 Удаление разделов 142](#_Toc178240401)

[22.2.7 Изменение типа раздела 143](#_Toc178240402)

[22.3 GPT 143](#_Toc178240403)

[22.3.1 Управление разделами GPT 143](#_Toc178240404)

[22.3.2 Таблица разделов 144](#_Toc178240405)

[22.3.3 Создание раздела 145](#_Toc178240406)

[22.3.4 Удаление раздела 145](#_Toc178240407)

[22.3.5 Варианты восстановления 145](#_Toc178240408)

[22.3.6 Работа с дисками 146](#_Toc178240409)

[22.3.7 Образы дисков 149](#_Toc178240410)

[23 Написание bash-скриптов 152](#_Toc178240411)

[23.1 Команды 152](#_Toc178240412)

[23.2 Сценарии 153](#_Toc178240413)

[23.3 Ввод и вывод данных 156](#_Toc178240414)

[23.4 Функции и аргументы 156](#_Toc178240415)

[23.5 Области видимости и потоки 159](#_Toc178240416)

[23.6 Операторы 160](#_Toc178240417)

[23.7 Циклы и конструкции 162](#_Toc178240418)

[23.8 Арифметические операции и массивы 164](#_Toc178240419)

[Приложение А. Список доступных консольных команд 167](#_Toc178240420)

[Перечень терминов и сокращений 186](#_Toc178240421)

1. Общие сведения
   1. Общее описание

Операционная система РОСА "ХРОМ" предназначена для оснащения серверов и рабочих мест пользователей.

Для ОС представлены версии в исполнении "рабочая станция" и "сервер".

Для ОС доступны образы с окружением рабочего стола "KDE Plasma 5".

* 1. Требования к техническим средствам

Для корректной установки и функционирования ОС необходима следующая минимальная конфигурация оборудования (ПК):

* процессор с архитектурой x86\_64 или AArch64;
* оперативная память 2 Гб (4 Гб при загрузке ОС по сети);
* дисковое пространство 20 Гб;
* VGA-адаптер и монитор с поддержкой разрешения 1024×768 пикс. (24 бит);
* устройство чтения оптических дисков или возможность загрузки с USB-носителя;
* клавиатура;
* "мышь".
  1. Состав ОС

ОС включает в себя следующие основные компоненты:

* ядро и системные библиотеки;
* системные сервисы и программы;
* пользовательские программы.

Ядро и системные библиотеки обеспечивают взаимодействие ОС с аппаратной частью и выделение ресурсов для всех других компонентов.

Системные сервисы и программы дают возможность настройки и реконфигурации ОС в зависимости от задач пользователя и обслуживание повседневной работы.

Пользовательские программы предназначены для решения задач конечного пользователя.

1. Оболочка
   1. Общие сведения

Командная строка или консоль термин, обозначающий взаимодействие пользователя с применением стандартных устройств ввода-вывода, обеспечивающих ввод команд и получение результатов их выполнения.

В ОС семейства Linux подобное взаимодействие обеспечивается специальным программным продуктом оболочкой (shell). Наиболее распространенной из них является оболочка проекта GNU bash.

Свойства оболочки bash делают ее наиболее универсальным и удобным средством взаимодействия с ОС:

* обеспечивает редактирование командной строки;
* поддерживает режим истории команд, позволяя отображать и изменять ранее введенные команды;
* обеспечивает завершение частично введенных слов;
* обеспечивает гибкое управление процессами, позволяя приостанавливать их, перезапускать, переключаться между задачами переднего плана и фоновыми, и даже продолжать выполнение задачи при завершении родительского процесса;
* позволяет использовать функции и псевдонимы, выполнять арифметические операции и вводить арифметические выражения в качестве команд;
* позволяет создавать целые наборы последовательно выполняемых команд (сценарии или скрипты) с использованием гибкого и функционального языка программирования данной оболочки;
* имеет гибкие настройки изменения внешнего вида командной строки.

Оболочка bash может выполняться как в интерактивном, так и в изолированном режимах. В первом случае программа bash взаимодействует с пользователем, во втором используется для выполнения скриптов (текстовых файлов с последовательностью команд).

При входе в консоль пользователя встречает стандартное приглашение командной строки (prompt):

# user@hostname ~ $

где:

* user – имя текущего пользователя;
* hostname – имя хоста, которое по умолчанию состоит из случайных символов;
* "@" – знак-разделитель;
* "~" – текущий каталог (в bash символ тильды соответствует домашнему каталогу, путь к нему /home/user);
* "$" – символ, который означает конец командной строки, после которого появится ввод пользователя с клавиатуры.

В контексте командной строки символ решетки ("#") используется для обозначения команд, выполняемых суперпользователем root в режиме администратора. Если в приглашении командной строки используется символ решетки, это обычно означает, что текущий пользователь имеет привилегии суперпользователя root.

* 1. Эмулятор терминала

Эмулятор терминала – это программа, которая позволяет использовать терминал в графической среде ОС.

Konsole – это эмулятор терминала, предоставляемый в рамках среды рабочего стола KDE Plasma; является одним из основных инструментов командной строки в KDE Plasma и предоставляет мощные возможности для взаимодействия с операционной системой через командную строку.

Konsole также интегрирована во многие другие приложения KDE, что упрощает доступ и делает ее более удобной. Например, программы KDevelop, Kate и Dolphin используют Konsole в качестве встроенного эмулятора терминала.

Для того чтобы открыть эмулятор командной строки в графическом режиме работы ОС, следует перейти в "Меню приложений → Konsole".

Интерфейс окна Konsole включает:

* **Меню**, в котором располагаются различные команды и опции, позволяющие настроить терминал, а также дополнительные опции позволяющие взаимодействовать с другими программами.
* **Панель инструментов**, содержащая кнопки для выполнения действий, таких как создание новой вкладки, разделение окна, копирование и вставка, а также поиск по терминалу.
* **Основная часть** окна представляет собой область вывода, в которой отображается результат выполнения команд, текстовый вывод и другая информация.
  1. Телетайп

В ОС по умолчанию доступно двенадцать так называемых TTY-консолей, которые можно использовать для выполнения команд. Такая консоль называется виртуальным терминалом.

В среде рабочего стола ОС и других Unix-подобных ОС эмуляторы терминала, такие как x-term и Konsole, являются примерами псевдотелетайпов, т.е. эмулируются программно.

* 1. Переключение между консолями

У каждой консоли есть своя комбинация клавиш Ctrl+Alt+Fn, где n **–** номер консоли от 1 до 12.

Таким образом, чтобы открыть первую консоль следует использовать комбинацию Ctrl+Alt+F1, вторую Ctrl+Alt+F2, двенадцатую Ctrl+Alt+F12 и так далее.

1. Команды
   1. Выполнение команд

Команды можно выполнять с помощью командной строки, указав имя файла (бинарного файла или сценария).

Каждая запущенная команда является отдельным процессом. Если команда выполняется на переднем плане (по умолчанию все команды выполняются именно так), то пользователь не сможет работать с терминалом, пока команда не будет выполнена.

Важно отметить, что в ОС почти всегда учитывается регистр, включая имена файлов и каталогов, команды, аргументы и опции. Если что-то не работает, следует проверить написание команды.

Полный список доступных консольных команд приведен в Приложении А настоящего руководства.

* + 1. Без опций и аргументов

Чтобы запустить команду без аргументов и опций, нужно просто ввести саму команду и нажмите ENTER.

В таком случае команда выполнит своё стандартное действие. У каждой команды индивидуальное поведение по умолчанию. Например, команда cd без аргументов откроет домашний каталог текущего пользователя. Команда ls без аргументов выводит список файлов и каталогов, хранящихся в текущем каталоге.

* + 1. С аргументами

Многие команды принимают аргументы или параметры, которые могут влиять на их поведение. Например, в качестве аргумента команды cd часто используется имя каталога, в который нужно перейти. Например, чтобы перейти в /usr/bin, нужно ввести:

cd /usr/bin

Сначала нужно ввести саму команду, а после неё указать аргументы. Перейдя в новый каталог, путь в командной строке изменится.

* + 1. С опциями

Большинство команд имеют список опций или флагов, которые позволяют изменять поведение команды.

Опции указываются после команды через дефис ("-"). Они бывают верхнего и нижнего регистра. Некоторые опции начинаются с двух дефисов ("--"). После одного дефиса можно указать несколько опций подряд, а после двух дефисов обычно идёт одна опция, состоящая из нескольких символов (обычно это описательное слово).

* + - 1. Использование опции для получения справки

Многие консольные команды предоставляют подробную справочную информацию, которая может быть получена с использованием опции -h или --help. Эти опции позволяют пользователям быстро ознакомиться с основными параметрами и функциональностью команды без необходимости изучения руководства.

Эти опции обычно выводят краткую справочную информацию, включающую список доступных опций, их описание и примеры использования. Этот подход облегчает работу с новыми командами и помогает быстро освоить основные возможности.

Рекомендуется использовать опции -h или --help перед тем, как начать работу с новой командой, чтобы узнать больше о ее функциональности и опциях, которые могут быть полезны в конкретном контексте.

Синтаксис опции -h:

command -h

где command – команда, по которой требуется справка.

Некоторые команды предоставляют дополнительную опцию --helpдля вывода справочной информации:

command --help

где command – команда, по которой требуется справка.

* + - 1. Другие опции команд

В качестве примера можно привести некоторые опции команды:

* "-l" – включает подробный листинг, который предоставляет дополнительные детали (права доступа, права собственности, размер файлов и т.п.).
* "-a" – выводит полный список файлов, которые хранятся в каталоге (включая скрытые файлы).

Флаг -l команде ls передается так:

ls -l

Такая команда выведет много дополнительной информации о файлах.

Опции можно группировать, например, передать команде ls флаги -l и –a одновременно:

ls -la

Эта команда выведет подробные данные обо всех файлах, включая скрытые.

* + 1. С опциями и аргументами

Опции и аргументы почти всегда можно комбинировать при запуске команд.

Например, чтобы просмотреть полное содержимое каталога /home, можно использовать:

ls -la /home

Такая команда вернёт подробные данные о каждом файле заданного каталога.

* 1. Фоновые процессы

Обычно для запуска команды необходимо ввести ее имя. В командной строке можно указать сколько угодно специальных символов, которые будут интерпретированы bash еще до запуска команды. Таким же образом можно запускать команды в фоновом режиме.

Важнейший и самый востребованный подстановочный символ &. Если указать его в конце командной строки, bash запустит программу в фоновом режиме. Это имеет смысл в первую очередь в случае, если на выполнение команды уходит много времени, чтобы можно было продолжать работу, не дожидаясь окончания выполнения такой программы.

find / -name '\*sh' > result &

[1] 3345

Вышеуказанная команда ищет по всей файловой системе файлы, названия которых оканчиваются на sh. Список найденных файлов записывается в файле result. Поскольку команда выполняется в фоновом режиме, работу можно не прерывать. Вывод "[1] 3345"означает, что фоновый процесс имеет номер "PID 3345". Здесь PID означает идентификатор процесса (process ID). Номер PID интересен в том случае, если процесс был аварийно завершен командой kill. Номер в квадратных скобках – это номер фонового процесса, запущенного в bash. Как правило, этот номер не важен.

Если при запуске команды не поставлен символ &, не следует ни дожидаться окончания выполнения программы, ни принудительно завершать программу нажатием Ctrl+C. Гораздо лучше приостановить выполнение программы, нажав Ctrl+Z, а затем продолжить ее работу в виде фонового процесса с помощью команды bg.

* 1. Код завершения (Exit Code)

Когда команда завершает свое выполнение в ОС, то возвращает целое число, называемое кодом завершения (exit code). Этот код информирует об успешном или неудачном выполнении команды.

Диапазон значений exit code:

* **Exit code 0** – обычно означает успешное выполнение команды без каких-либо ошибок.
* **Exit code от 1 до 255** – обозначает различные ошибочные состояния или условия. Но exit code может быть и любым другим целым числом, и каждая команда может определить свой собственный набор exit code. Следует ознакомиться со справочной информацией о команде (с помощью -h или --help), чтобы узнать о ее exit code.

Пользователь может инициировать проверку кода завершения команды, используя переменную $? для получения exit code последней выполненной команды.

Таким образом, exit code – это механизм, который позволяет программам и скриптам ОС передавать информацию об успешности выполнения операций и об условиях ошибок для дальнейшей обработки.

* 1. Выполнение нескольких команд

ОС позволяет производить запуск нескольких команд в одной строчке. Существует два способа:

* использование символа ";":

sudo dnf update; sudo dnf upgrade -y

Вторая команда будет запущена, даже если первая команда не будет выполнена. Если такой вариант не подходит, рекомендуется использовать второй вариант.

* использование символы "&&":

sudo dnf update && sudo dnf upgrade -y

В этом случае будут выполняться обе версии команды, но вторая форма будет выполнять вторую команду только в том случае, если первая завершится успешно.

При использовании символа & также можно указать следующую команду. В таком случае первая команда будет выполняться в фоновом режиме, а вторая на первом плане. Например, команда find запускается в фоновом режиме, а ls одновременно выводит содержимое текущего каталога:

find / -name '\*sh' > result & ls

Если вместо символа & поставить точку с запятой, то bash поочередно выполнит команды в фоновом режиме:

rls; date

Эта команда сначала отображает содержание текущего каталога, а затем выводит текущий файл.

Если необходимо перенаправить всю эту информацию в файл с помощью ">", то обе команды ставятся в круглых скобках. В таком случае они выполняются одной и той же оболочкой:

user$ (ls; date) > content

В файле contentтеперь находится список файлов, созданный ls, а также текущая дата (из команды date**)**. Благодаря круглым скобкам обе команды выполняются одной и той же оболочкой и выдают общий результат (как правило, ситуация иная при запуске каждой новой команды активируется новая оболочка).

* использование символов "**||**":

Оператор **||** используется для выполнения второй команды только в том случае, если первая завершилась неуспешно (код завершения не равен 0), например:

команда1 || команда2

Если команда1 завершится успешно, то команда2 не выполнится. В противном случае, если команда1 завершится с ошибкой, команда2 будет выполнена.

Эти операторы && и || предоставляют удобные способы создания цепочек команд с учетом их успешного или неуспешного завершения, что может быть полезным в скриптах и командной строке для управления логикой выполнения команд, то есть в зависимости от результата другой команды.

Например,

user$ команда1 && команда2

выполняет команда1. Только в случае, если эта команда была выполнена успешно (без ошибки, без выдачи 0 в качестве возвращаемого значения), выполняется команда2.

Ниже приведены все варианты использования двух команд (Таблица 1).

Таблица 1 - Использование двух команд

|  |  |
| --- | --- |
| **Команда** | **Функция** |
| команда1 ; команда2 | Выполняет команду друг за другом |
| команда1 && команда2 | Выполняет команда2, если команда1 была выполнена успешно |
| команда1 || команда2 | Выполняет команда2, если команда1 была выполнена с ошибкой |
| команда & | Запускает команду в фоновом режиме |
| команда1 & команда2 | Запускает команда1 в фоновом режиме, команда2 на виду |
| (команда1 ; команда2) | Выполняет обе команды в одной и той же оболочке |

Другие возможности для создания условий и ветвления команд связаны с использованием оболочковой команды if, которая интересна администраторам, которые собираются заниматься программированием на языке оболочки.

* 1. Работа с каталогами

Таблица 2 представляет основные команды по работе с каталогами.

Таблица 2 - Команды для работы с каталогами

|  |  |
| --- | --- |
| **Команда** | **Функция** |
| cd | Смена актуального каталога |
| cp | Копирование файлов |
| less | Постраничный показ текстовых файлов |
| ls | Показ всех файлов каталога |
| mkdir | Создание нового каталога |
| mv | Перемещение файлов или изменение их имен |
| rm | Удаление файлов |
| rmdir | Удаление каталогов |

Корневой каталог Системы **~** аналогичен каталогу /home/\*username\* и обозначает домашний каталог пользователя.

* 1. Перенос строк

В Системе позволяется вводить команды, расположенные на нескольких строках, используя символ **"\"**. Для сравнения можно привести пример стандартного выполнения команды и с использованием символа **"\"**:

ping 127.0.0.1

ping \

> 127.0.0.1

* 1. Множитель вывода

Иногда программный вывод нужно сохранить в файле, но при этом необходимо отслеживать выполнение программы на экране. В таком случае требуется удвоение ввода, причем одна копия отображается на экране, а вторая сохраняется в файле. Эту задачу выполняет команда tee:

user$ ls | tee content

Содержание текущего каталога отображается на экране и одновременно сохраняется в файле content. При этом сначала происходит переадресация стандартного вывода к команде tee. По умолчанию стандартный вывод отображается на экране, а копия этой информации сохраняется в указанном файле. Поскольку в данном случае речь в действительности идет о размножении вывода, следует обратить внимание, как переадресовать стандартный вывод teeв файл:

user$ ls | tee content1 > content2

В итоге получаются два одинаковых файла: content1и content2. Следующий пример более сложен, но более практичен:

user$ ls -l | tee content1 | sort +4 > content2

В content1 расположено обычное оглавление, которое автоматически отсортировано командой lsпо имени. Копия этого вывода передается sort, а затем происходит сортировка по размеру файла (пятый столбец, то есть параметр +4) и сохранение информации в файле content2.

1. Переменные окружения

Переменные окружения – это значения, которые влияют на выполнение команд и процессов. При первом запуске сервера ОС установит несколько переменных окружения в соответствии с конфигурационными файлами по умолчанию.

* 1. Просмотр переменных
     1. Команда env

Чтобы просмотреть все переменные окружения, установленные для определенной сессии терминала, используют команду env:

env

Вывод этой команды обычно очень объёмен: требуется найти запись PATH вида:

PATH=/root/.local/bin:/root/bin:/sbin:/bin:/usr/sbin:/usr/bin

Переменная окружения PATH – это разделённый двоеточиями список каталогов, в которых оболочка будет искать исполняемые программы или сценарии при выполнении команды. Например, команда env выполняется в каталоге /usr/bin, но при запуске команды не нужно указывать путь к ней, поскольку он уже указан в переменной PATH.

* + 1. Встроенная команда set в bash

В bash команда set используется для управления настройками оболочки, включая переменные окружения.

Просмотр всех настроек оболочки доступен по команде:

set

Эта команда выводит все настройки оболочки, включая переменные окружения.

Просмотр только переменных окружения:

set | grep "^VAR="

Пример:

set | grep "^PATH="

Эта команда использует grep для фильтрации вывода команды set и выводит только строки, начинающиеся с "VAR=" (например, "PATH="). Таким образом можно увидеть значение переменной PATH.

* 1. Значения переменных

Чтобы извлечь значение переменной, нужно указать перед её именем символ "$".

Например, чтобы вывести значение переменной PATH, можно использовать команду echo:

echo $PATH

Чтобы узнать значение переменной HOME (по умолчанию это домашний каталог текущего пользователя), следует перейти в домашний каталог:

cd $HOME

Если запрашиваемая переменная не установлена, терминал вернёт пустую строку.

* 1. Установка значений

Чтобы установить значение переменной окружения, вводят имя переменной, символ "=", а затем указывают нужное значение:

ПЕРЕМЕННАЯ=значение

Следует обратить внимание, что, если устанавливается существующая переменная, её исходное значение будет перезаписано. Если переменная не существует, она будет создана.

Bash предоставляет команду export, которая экспортирует переменные, после чего дочерние процессы используют их значения. Проще говоря, эта команда позволяет использовать сценарии, которые ссылаются на экспортируемые переменные окружения в текущей сессии.

Также на существующие переменные можно ссылаться при установке переменной. Например, если приложение установлено в /opt/app/bin, можно добавить этот каталог в конец переменной PATH с помощью такой команды:

export PATH=$PATH:/opt/app/bin

* 1. Использование справочных ресурсов

Справочные страницы онлайн-руководства (так называемые man-страницы, поскольку их можно просмотреть с помощью команды man) составляют традиционную “интерактивную” документацию. Эти материалы в основном инсталлируются вместе с ОС, а справочные страницы отдельных программ – вместе с соответствующим пакетом.

Интерактивное руководство содержит краткое описание отдельных команд, драйверов, форматов файлов и библиотечных функций.

Во всех ОС справочные страницы обычно делятся на разделы, при этом возможны небольшие вариации на тему принципов такого разделения.

Некоторые разделы делятся на подразделы. Точная структура разделов не так уж важна, поскольку команда man находит соответствующую страницу в большинстве случаев. При этом в нескольких разделах может быть одно и то же имя: в этом случае важно указать нужный раздел. Например, passwd– это и команда, и файл конфигурации, поэтому необходимые статьи есть в соответствующих разделах.

* + 1. Команда man

Команда:

man раздел заголовок

вызывает справочную страницу из указанного раздела. Например, команда "man sync" вызывает справочную страницу для команды sync, а команда "man 2 sync" для системного вызова sync.

Команда "man -k ключевoe\_слово" или "apropos ключевое\_слово" выводит список справочных страниц, в строке пояснений к которым имеется указанное ключевое слово.

man -k translate

objcopy (1) copy and translate object files

dcgettext (3) translate message

tr (1) translate or delete characters

snmptranslate (1) translate SNMP OID values into more useful information

tr (lp) translate characters

База данных ключевых слов может устаревать. При добавлении новых справочных страниц к ОС, возможно, придется перестроить (перекомпоновать) БД с помощью команды makewhatis.

* + 1. Хранение страниц интерактивного руководства

Неформатированная информация для справочных страниц (входные данные команды nroff) обычно хранится в подкаталогах каталога /usr/share/man. В целях экономии места на диске ОС сжимает страницы с помощью утилиты Gzip (команда man может очень быстро разархивировать их). Команда man поддерживает кеш отформатированных страниц в каталогах /var/cache/man или /usr/share/man, если соответствующие каталоги доступны для записи, но эти операции рискованны с точки зрения безопасности.

Команда man ищет страницы в ряде каталогов. В ОС выяснить путь поиска позволяет команда manpath:

manpath

/usr/local/man:/usr/local/Share/man:/usr/Share/man

Эта установка хранится в переменной среды MANPATH, и в случае необходимости ее можно изменить.

export MANPATH=/home/Share/localman:/usr/Share/man

* + 1. Команда TLDR

В качестве альтернативы для получения справочной информации используют утилиту tldr ("Too long; didn't read") для лаконичности документации в консоли.

Утилита tldr доступна из репозитория ОС и устанавливается командой:

sudo dnf in tldr -y

Например, для получения tldr-справки по tar выполняется команда:

tldr tar

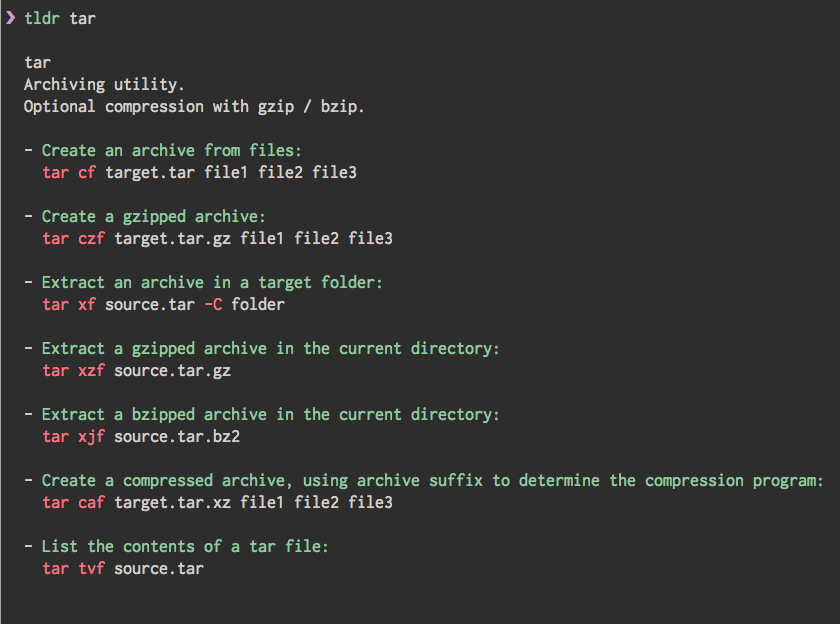


Рисунок 1 - Справка по tar

Вывод команды содержит минимальное описание назначения утилиты и компактный список из самых частых команд (Рисунок 1).

1. Инструменты управления пакетами

При решении различных задач по управлению пакетами ПО необходимо учитывать, что существуют два типа утилит: низкоуровневые инструменты (производящие фактическую установку, обновление и удаление файлов пакетов) и высокоуровневые инструменты (отвечающие за выполнение задач по разрешению зависимостей и поиска метаданных, так называемые "данные о данных").

* 1. Высокоуровневый менеджер

Высокоуровневые менеджеры применяются для поиска и скачивания пакетов из репозиториев. В процессе работы могут быть задействованы низкоуровневые менеджеры для инсталляции загруженных программ. В ОС РОСА ХРОМ в качестве такого менеджера используется dnf.

Синтаксис команд dnf:

dnf <действие> <пакет> <ключ>

Важные и часто используемые команды менеджера пакетов dnf представлены ниже (Таблица 3).

Таблица 3 - Команды dnf

|  |  |
| --- | --- |
| Команда | Описание |
| dnf install имя пакета | Установка нового ПО |
| dnf remove имя пакета | Удаление установленного пакета с его зависимостями |
| dnf groupremove имя группы | Удаление пакетов определенной группы |
| dnf update | Обновление всех установленных пакетов |
| dnf check-update | Проверка обновлений пакетов |
| dnf update имя пакета | Обновление конкретного пакета |
| dnf search имя пакета | Поиск пакетов в репозиториях |
| dnf info имя пакета | Получение информации о пакете |
| dnf list installed | Вывод списка установленных пакетов |
| dnf list available | Вывод списка доступных для установки пакетов |
| dnf upgrade | Обновление системы до новой версии |
| dnf repolist | Просмотр списка репозиториев |
| dnf history | Просмотр истории транзакций |
| dnf clean all | Очистка кеша и временных файлов |
| dnf config-manager | Управление настройками репозиториев |

* 1. Низкоуровневый менеджер

Низкоуровневый менеджер используется для установки локальных пакетов, загруженных вручную пользователем, или высокоуровневым пакетным менеджером. В качестве такого менеджера в ОС используется rpm:

Синтаксис команд rpm:

rpm <ключ> <пакет>

Для установки пакета через rpm необходимо наличие предварительно загрузить нужный пакет формата .rpm.

Основные команды rpm приведены ниже (Таблица 4).

Таблица 4 - Команды rpm

| Команда | Описание |
| --- | --- |
| **Установка пакетов** | |
| rpm -i package.rpm | Установка пакета |
| rpm -i --nodeps package.rpm | Установка пакета без его зависимостей |
| rpm -U package.rpm | Обновление существующего пакета или установка нового |
| **Удаление пакетов** | |
| rpm -e package.rpm | Удаление установленного пакета |
| rpm --erase package.rpm | Альтернативный вариант удаления пакета |
| rpm -e --nodeps package.rpm | Удаление пакета без учёта его зависимостей |
| **Сбор информации о пакете** | |
| rpm -qi <название пакета> | Краткая информация о пакете |
| rpm -q <название пакета> | Полное наименование пакета и его версия |
| rpm -ql <название пакета> | Список файлов пакета |
| rpm -qf <путь до файла> | Узнать какому пакету принадлежит файл |
| rpm -qa | grep ‘часть названия’ | Вспомнить полное название пакета |
| rpm -qip <название пакета>.rpm | Информация об неустановленном пакете |
| rpm -qal | grep -i monitors | Вспомнить каталог пакета |
| rpm -qRp <название пакета>.rpm | Узнать зависимости пакета |
| rpm -qs <название пакета> | Проверка текущего состояния пакета |
| rpm -Vf /путь/к/программе | Проверка целостности указанного пакета |

1. Потоки

В ОС понимание концепции потоков является фундаментальным для работы с командной строкой и написания эффективных скриптов. Потоки подразумевают каналы передачи данных между программами. Потоки предоставляют механизм обмена данными между программами и обеспечивают важный способ передачи информации внутри Системы.

Общие понятия потоков описываются следующими возможностями:

* **Стандартные потоки**:
* **Стандартный ввод (**stdin**)** – поток данных, который принимает ввод от пользователя (например, с клавиатуры) или других программ.
* **Стандартный вывод (**stdout**)** – поток данных, который представляет собой вывод программы и обычно отображается в терминале.
* **Стандартный поток ошибок (**stderr**)** – поток данных, используемый для вывода сообщений об ошибках и предупреждений.
* **Потоки ввода и вывода:**
* **Поток ввода (Input Stream)** – источник данных, из которого программа читает информацию, например stdin.
* **Поток вывода (Output Stream)** – канал, в который программа выводит результаты, например stdout и stderr.
* **Перенаправление потоков:**
* **Перенаправление ввода (**<**)** – изменение источника данных для программы, используя данные из файла вместо stdin.
* **Перенаправление вывода (**>**)** – запись вывода программы в файл вместо stdout.
* **Перенаправление ошибок (**2>**)** – перенаправление сообщений об ошибках в файл вместо stderr.

Например, возможен случай, при котором содержание текущего каталога должно не отображаться на экране, а сохраняться в файле. Таким образом, стандартный вывод должен переадресовываться в настоящий файл. В bash это делается с помощью символа ">":

ls \*.txt > content.txt

В файл content выведен список всех текстовых файлов, расположенных в текущем каталоге. Чаще всего применяется именно такой способ переадресации вывода.

Есть еще два варианта:

* 2> файл – переадресовывает сообщения об ошибках в указанный файл;
* >& файл или &> файл – переадресовывают в указанный файл и сообщения об ошибках, и программный вывод.

Если использовать вместо ">" двойной символ ">>", то весь ввод дописывается в конец уже имеющегося файла.

Переадресация ввода осуществляется с помощью "< файл": команды, ожидающие ввода с клавиатуры, считывают ввод из указанного таким образом файла.

Следует обратить снимание, что невозможно одновременно обрабатывать файл и записывать в него получаемый результат.

Команды вида:

sort dat > dat

или

sort < dat > dat

приводят к удалению файла dat.

Возможности переадресации ввода и вывода приведены ниже (Таблица 5).

Таблица 5 - Переадресация ввод и вывода

|  |  |
| --- | --- |
| Команда | Функция |
| Команда > файл | Переводит стандартный вывод в указанный файл |
| Команда < файл | Считывает ввод из указанного файла |
| Команда 2> файл | Переводит сообщения об ошибках в указанный файл |
| Команда >& файл | Переадресовывает вывод и ошибки |
| Команда >> файл | Дописывает стандартный вывод в конце имеющегося файла |
| Команда &>> файл | Дописывает стандартный вывод и ошибки в конце имеющегося файла |
| Команда1 | Команда 2 | Передает вывод команды1 на ввод команде2 (stdout команды1 передается stdin команде2) |
| Команда | tee файл | Показывает вывод и одновременно сохраняет копию файла |

В ОС РОСА "ХРОМ" новый процесс создается путем копирования атрибутов текущего процесса. Новый процесс может быть клонирован и при этом такие ресурсы, как файлы, обработчики сигналов и виртуальная память, используются совместно. Если два процесса пользуются одной и той же виртуальной памятью, они функционируют как потоки в рамках одного и того же процесса. Однако для потоков структуры данных отдельно не задаются. Таким образом, в ОС РОСА ХРОМ потоки и процессы не различаются.

Более подробный разбор потоков приведен в п.14.2.

1. Поиск информации

В Системе предусмотрено множество возможностей поиска файлов. Выбор команды, лучше всего подходящей для конкретного случая, зависит от типа файла (текстовый файл, программа и т. д.) и его содержания (фрагменты названия, фрагменты текста и т. д.).

Ниже приведены инструменты для поиска файлов (Таблица 6).

Таблица 6 - Команды поиска файлов

|  |  |
| --- | --- |
| Команда | Функция |
| grep | Поиск текста в текстовом файле |
| find | Поиск файлов по имени, дате, размеру и т.д. |
| locate | Поиск фалов по имени |
| whereis | Поиск файлов заданных каталогах |
| which | Поиск программ в PATH-каталогах |

* 1. Команды which, whereis, whatis

Команда whichищет указанную команду и возвращает полное имя команды, которая была бы выполнена, если было набрано ее имя без указания пути. Команда whichосуществляет поиск только по каталогам, указанным в PATH, работая при этом исключительно быстро. PATH содержит список каталогов, в которых находятся программы. Однако необходимо учитывать, что PATHдля администратора содержит больше каталогов, чем для обычного пользователя. Таким образом, для поиска системных команд необходимо войти в Систему с правами администратора.

user$ which emacs

/usr/bin/emacs

Команда whereisпросматривает все пути, обычно используемые для двоичных файлов, файлов конфигурации, man-страниц и исходного кода. Для этого ее нужно ввести после имени файла. Команда whereisохватывает больше файлов, чем which, при этом поиск не ограничивается одними только программами. Она не может искать в каталогах, которые специально не указаны программе как места для поиска.

user$ whereis fstab

fstab: /etc/fstab /usr/include/fstab.h

Команда whatisвыводит краткое однострочное описание из руководства (man).

# whatis ls cp rm mv

ls (1) list directory contents

cp (1) copy files and directories

rm (1) remove files or directories

mv (1) move (rename) files

Также можно использовать команду:

command -V команда1

которая выводит путь к исполняемому файлу (или команде) в командной строке.

Например, команда:

command -V ls

может быть полезна, чтобы узнать место загрузки определенной команды в Системе.

* 1. Команда locate

Команда locateищет файлы, в полном названии которых (путь + имя файла) содержится указанная схема поиска. Поиск осуществляется очень быстро: locateне ищет по файловой системе, а обращается к БД, в которой содержится список имен всех файлов из файловой системы. В зависимости от дистрибутива locateпоказывает все файлы, к которым пользователь может получить доступ. Если нужно найти системные файлы, то следует войти в Систему как администратор и выполнить команду locate. Использовать ее можно только в том случае, если в Системе установлен соответствующий пакет, который по умолчанию имеется не во всех дистрибутивах.

Например, следующая команда ищет конфигурационный X-файл xorg.conf*:*

user$ locate xorg.conf

/etc/X11/xorg.conf

/etc/X11/xorg.conf.backup

* 1. Особенности

Внедрение команд locateи updatedbотличается от дистрибутива к дистрибутиву. В ОС РОСА "ХРОМ" эти команды входят в состав стандартного пакета mlocate. Файловая база данных находится в файле /var/lib/mlocate/mlocate.db и ежедневно обновляется программой Cron-Job /etc/cron.daily/mlocate. Конфигурационный файл /etc/updatedb.conf определяет, какие каталоги и файловые системы не будут учитываться при поиске (например, CD, DVD, различные буферные каталоги).

* 1. Команда wc

Команда wc позволяет выполнять подсчёт, считывая стандартный ввод или составной список и генерируя одну или несколько из следующих статистических данных: количество строк, количество слов и количество байтов.

Общий вид команды:

user$ wc [ПАРАМЕТР]… [ФАЙЛ]…

Параметры команды wc представлены ниже (Таблица 7):

Таблица 7 - Параметры команды wc

|  |  |
| --- | --- |
| Параметр | Значение |
| -c, --bytes | Вывести количество байтов |
| -m, --chars | Вывести количество символов |
| -l, --lines | Вывести количество новых строк |
| -w, --words | Вывести количество слов |

Вывод команды wc представляется в следующем виде:

* первая колонка, количество строк (-l);
* вторая колонка, количество слов (-w);
* третья колонка, количество символов (-m).
  1. Команда sort

Командаsort используется в ОС для сортировки содержимого текстовых файлов. Это позволяет выводить текстовые строки в определённом порядке.

Общий вид команды:

sort [ПАРАМЕТР]… [ФАЙЛ]…

Команда направляет сортированное слияние всех [ФАЙЛ] на стандартный вывод.

Если [ФАЙЛ] не задан или задан как **"-"**, читает стандартный ввод.

Параметры команды sortпредставлены ниже (Таблица 8).

Таблица 8 - Параметры команды sort

|  |  |
| --- | --- |
| **Параметр** | **Значение** |
| -b, --ignore-leading-blanks | Игнорировать начальные пропуски |
| -d, --dictionary-order | Рассматривать только пропуски, буквы и цифры |
| -f, --ignore-case | Игнорировать регистр букв |
| -g, --general-numeric-sort | Сравнивать в соответствии с общим числовым значением |
| -i, --ignore-nonprinting | Рассматривать только печатные символы |
| -h, --human-numeric-sort | Сравнивать числа в читабельном виде (например, 2K 1G) |
| -n, --numeric-sort | Сравнивать по числовым значениям строк |
| -r, --reverse | Изменить результаты сравнения на противоположные |
| -c, --check, --check=diagnose-first | Проверять, сортированы ли входные файлы |
| -k, --key=ОПР\_КЛЮЧА | Сортировать по ключу; в ОПР\_КЛЮЧА задано расположение и тип |
| -m, --merge | Объединить уже отсортированные файлы |
| -o, --output=ФАЙЛ | Выводить в ФАЙЛ, а не на стандартный вывод |
| -u, --unique | С "-c" проверять порядок строго; без "-c" выводить только первое среди нескольких равных |

Примечание – ОПР\_КЛЮЧА задается как F[.C][OPTS][,F[.C][OPTS]] для начала и конца положения, где F номер поля, а C позиция в поле; оба отсчитываются от **1**, а конечная позиция по умолчанию равна концу строки.

Если не задан параметр -t или -b, то символы в поле отсчитываются от начала предшествующего пробельного символа. OPTS составляется из одной или нескольких букв, задающих порядок сортировки [bdfgiMhnRrV]; он переопределяет соответствующие глобальные параметры сортировки для данного ключевого поля.

Сортировка по умолчанию (без опций):

sort example.txt

Обратная сортировка:

sort -r example.txt

* 1. Команда cut

Команда cut предназначена для вырезания части текста из файла и записи результата в стандартный вывод.

Общий вид команды:

cut [опция] [файл]

Параметры команды cutпредставлены ниже (Таблица 9).

Таблица 9 - Параметры команды cut

|  |  |
| --- | --- |
| Параметр | Значение |
| -b, --bytes=СПИСОК | Выбрать только заданные байты |
| -c, --characters=СПИСОК | Выбрать только заданные символы |
| -d, --delimiter=РАЗДЕЛИТЕЛЬ | Использовать для разделения полей РАЗДЕЛИТЕЛЬ вместо табуляции |
| -f, --fields=СПИСОК | Выбрать только заданные поля; также печатать все строки, не содержащие разделителей, если только не задан параметр -s |
| --complement | Дополнить множество выбранных байтов, символов или полей |
| -s, --only-delimited | Не выводить строки, не содержащие разделителей |
| --output-delimiter=СТРОКА | Использовать СТРОКА для разделения полей при выводе, по умолчанию используется разделитель для ввода |
| -z, --zero-terminated | Разделитель строк NULL, а не символ новой строки |

* 1. Команда find

Если не указывать дополнительные параметры, findвыдает список всех файлов, находящихся в текущем каталоге и всех его подкаталогах:

Общий вид команды:

user$ find

Следующая команда ищет все файлы в текущем каталоге и во всех подкаталогах, названия которых начинаются с".t":

user$ find -name '.e\*'

./.thunderbird

./.thunar

./.textgen

…

В следующем примере findищет все каталоги, находящиеся в /etc/. Обычные файлы, располагающиеся в /etc/., среди результатов не показываются. Список результатов упорядочивается по алфавиту с помощью команды sort (по умолчанию такой сортировки не происходит).

root# find /etc -type d | sort

/etc

/etc/alsa

/etc/alsa/conf.d

/etc/alternatives

...

В следующем примере findищет все файлы в (под)каталогах /home, принадлежащих пользователям группы users, причем искомые файлы должны были каким-либо образом быть изменены в течение последних пяти дней (содержание, права доступа и т. д.):

root# find /home -group users -mtime -5

...

Команда "find -mtime +5" находит такие файлы, которые были изменены ранее, чем пять дней назад, а команда "-mtime 5" возвращает файлы, которые были изменены ровно пять дней назад. При этом во многих случаях "вчера" означает для find "24 и более часов от данного момента". Если вместо -mtime применить параметр -ctime, то точкой изменения времени будет считаться "время изменения индексного дескриптора". Эта точка изменяется и тогда, когда изменились лишь права доступа, а содержимое каталога осталось прежним.

Следующая команда удаляет все резервные копии, содержащиеся в данном каталоге и во всех подкаталогах. При этом findстроит список всех сомнительных файлов и передает его команде rmчерез подстановку команды ($(команда)):

user$ rm $(find. -name '\*~')

Если речь идет об очень большом количестве файлов, то при выполнении вышеуказанной команды возникает ошибка: запись со всеми *\*~*-файлами может получиться настолько длинной, что превысит размер командной строки. В таких случаях следует воспользоваться либо параметром -exec- команды find, либо командой xargs.

* 1. Команда grep

Команда grep ищет в текстовом файле соответствия для заданной поисковой схемы. В зависимости от того, какие параметры настроены в конкретном случае, команда может дополнительно показывать найденные фрагменты текста или просто сообщать, в каком количестве строк была найдена заданная поисковая схема. Поисковая схема является так называемым регулярным выражением.

Следующая команда просматривает все tex-файлы текущего каталога в поисках последовательности символов emacs. Список всех найденных строк (перед каждой из которых указывается имя файла) отображается на экране:

grep emacs \*.tex

Команда grepопределяет, как часто применяется функция arctanв указанных с-файлах.

grep -c arctan\(.\*\) \*.c

Команда grepс параметром -v возвращает в качестве результата все строки, в которых отсутствует заданный шаблон поиска.

В приведенном далее примере grepудаляет из configfileвсе строки, которые начинаются с символа"#"(то есть все комментарии). Следующая команда catдополнительно удаляет все пустые строки. Конечный результат сохраняется в файле nocomments.

grep -v '^#' configfile | cat -s > nocomments

Эта команда очень удобна, если всего несколько строк конфигурационного кода приходится на сотни или тысячи строк комментариев.

* 1. Комбинации команд find и grep

Команды findи grepможно комбинировать, чтобы выполнять расширенный поиск. В следующем примере команда findпросматривает все файлы на предмет того, нет ли в них последовательности символов emacs. Если такая последовательность обнаруживается, то название файла выводится на экран. Следует обратить внимание на то, что нельзя указывать параметр –print перед–exec(в отличие от предыдущего примера, команда"grep emacs \*.tex"учитывает все файлы с расширением \*.tex, независимо от глубины вложения подкаталога, в котором они могут находиться).

find -name '\*.tex' -type f -exec grep -q emacs {} \; -print

Следующая команда просматривает в текущем каталоге все файлы размером менее 10 Кбайт на предмет наличия в них регулярного выражения case.\*in. Список найденных файлов сохраняется в файле result. Ограничение размера файла вводится для того, чтобы исключить из поиска двоичные файлы (обычно они гораздо больше 10 Кбайт).

user$ find -name '\*' -maxdepth 1 -size -10k -exec grep -q \

> case.\*in {} \; -print > result

1. Работа с пользователями
   1. Общие сведения

В данном разделе рассмотрены основные моменты, касающиеся работы с пользователями в Системе.

Любая работа выполняется от имени одного из пользователей, зарегистрированного и авторизованного в Системе.

В Системе существуют три типа пользователей:

* **Администраторы** – привилегированные пользователи с полным доступом к Системе. По умолчанию на серверной ОС после ее установки всегда есть один такой пользователь root.
* **Локальные пользователи** – непривилегированные пользователи. Их учётные записи создаёт администратор. Особенность таких аккаунтов в ограниченном доступе к серверу в том, что они не могут пользоваться некоторыми системными утилитами, могут работать только с теми файлами и папками, к которым у них открыт доступ. Тем не менее привилегии локального пользователя можно повысить при необходимости.
* **Системные пользователи** – учётные записи, автоматически создаваемые Системой для работы внутренних процессов и служб. Например, после установки веб-сервера Apache, который входит в состав стека LAMP, на сервере появляется пользователь www-data, от имени которого потом работает веб-сервер. Такие пользователи нужны для повышения безопасности.

Каждый пользователь имеет свой уникальный идентификатор пользователя UID. Идентификатор отличается в зависимости от типа пользователя:

* администратор – 0;
* обычный (локальный) пользователь – от 100;
* системный пользователь – от 1 до 100.

Чтобы упростить процесс настройки прав для новых пользователей, их объединяют в группы. Каждая группа имеет свой набор прав и ограничений. Любой пользователь, создаваемый или добавляемый в такую группу, автоматически их наследует. Если при добавлении пользователя для него не указать группу, то у него будет своя, индивидуальная группа с именем пользователя. Один пользователь может одновременно входить в несколько групп.

Информацию о каждом пользователе сервера можно посмотреть в файле /etc/passwd. Пользователи в нём перечислены в следующем формате:

test-user:x:1000:1000::/home/test-user:/bin/bash

Значения параметров вывода информации о пользователях приведены ниже (Таблица 10).

Таблица 10 - Параметры данных о пользователе

|  |  |
| --- | --- |
| Параметр | Значение |
| test-user | Имя пользователя |
| x | Признак зашифрованного пароля (хранится в /etc/shadow) |
| 1000:1000 | Идентификатор пользователя (UID) и идентификатор группы (GID), к которой он принадлежит |
| :: | Комментарий к пользователю и дополнительная информация (например, контакты, имя и прочее) |
| /home/test-user | Адрес домашний папки пользователя |
| /bin/bash | Указывает, что пользователь имеет доступ к командной оболочке сервера |

Для отображения только учетных записей пользователей можно ввести команду:

users

Если известно имя пользователя и требуется узнать о нём больше информации, необязательно читать /etc/passwd. Всё то же самое можно посмотреть с помощью команды "pinky -l":

pinky -l test-user

В качестве альтернативы можно использовать команду "w"она тоже показывает всех авторизованных в текущий момент пользователей, включая создаваемую ими нагрузку на сервер.

Узнать UIDпользователя можно с помощью команды id :

id username

Вывод команды будет похожим на следующий:

uid=1001(username) gid=1001(username) groups=1001(username),27(sudo)

в котором содержится информация о том, что учетная запись username существует, она находится в основной группе username и дополнительной группе sudo; идентификатор учетной записи – 1001.

Также можно получить строку о пользователе из файла /etc/passwd:

getent passwd username

Вывод команды будет похожим на следующий:

username:x:1001:1001::/home/username:/bin/sh

в котором пользователь usernameимеет идентификатор пользователя и группы 1001, его домашняя директория – /home/username, командная оболочка – /bin/sh.

Для именования пользователей в ОС есть набор стандартных правил:

* имя пользователя может содержать только английские буквы [a-z] в верхнем и нижнем регистре, цифры, символ"\_", тире "-" и точку;
* имя пользователя может оканчиваться символом "$";
* имя пользователя не может начинаться с тире, содержать только цифры или состоять из "." или "..";
* не рекомендуется использовать точку в начале имени пользователя;
* имя пользователя может включать до 32 символов.
  1. Создание нового пользователя

Процесс добавления пользователя состоит из двух шагов: создания пользователя и настройки пароля.

На первом шаге используется команда useraddc набором опций для настройки нового пользователя и его именем (логином):

useradd <имя пользователя> [опции]

Опции не являются обязательными при создании пользователя.

Эта команда имеет ряд настроек по умолчанию, которые задаются с помощью файлов /etc/default/useradd и /etc/login.defs. Вывести на экран основные можно с помощью команды (Рисунок 2):

useradd -D



Рисунок 2 - Добавление нового пользователя

Ниже приведены параметры, которые Система выводит при создании нового пользователя (Таблица 11).

Таблица 11 - Параметры нового пользователя

|  |  |
| --- | --- |
| Параметр | Значение |
| GROUP | GIDгруппы, в которую пользователь будет добавлен после создания |
| HOME | Базовый каталог, в котором будет размещена директория пользователя |
| INACTIVE | Время до блокировки пользователя, когда его пароль станет недействителен. Значение "-1" отключает опцию |
| EXPIRE | Дата, до которой действителен аккаунт. По умолчанию не установлена, то есть без ограничений |
| SHELL | Настройка доступа к командной оболочке |
| SKEL | Путь к директории, в которой хранятся файлы по умолчанию. После создания пользователя они будут автоматически скопированы в его домашнюю папку |
| CREATE\_MAIL\_SPOOL | Определяет, нужно ли создать папку для писем этого пользователя в/var/spool/mail/ |

Все эти настройки применяются, если использовать самый простой вариант команды создания пользователя без параметров:

useradd test-user

Но обычно требуется добавить пользователя со специфическими настройками, для чего используется расширенный вариант этой команды (Таблица 12).

Таблица 12 - Основные опции useradd

| Параметр | Значение |
| --- | --- |
| -m | Создаёт указанную домашнюю директорию, если она ещё не существует |
| -d /home/test-user | Устанавливает /home/test-user в качестве домашней директории |
| -c "Пользователь" | Добавляет комментарий, например, с именем пользователя |
| -g test | Указывает группу, в которую попадёт пользователь после создания. Можно использовать с GID или именем группы. Указанная группа должна существовать. Используется в сочетании с ключом -N (отменяет автоматическое создание группы с именем пользователя) |
| -G users,wheel | Указывает список дополнительных групп пользователя. Они перечисляются через запятую без пробелов |
| -s /bin/bash | Позволяет настроить доступ к shell |
| -r | Создаёт системного пользователя. Используется, если нужно настроить службу на работу из-под конкретного пользователя. По умолчанию данные таких пользователей не вносятся в /etc/shadow, для них не создаётся домашняя папка |
| -u | Позволяет указать свой UID, который будет присвоен новому пользователю. В качестве UID указывается положительное целое число. UID должен быть уникален |
| -e 2021-01-01 | Указывает дату, до которой аккаунт будет активен. Дата задаётся в формате YYYY-MM-DD |
| -f 3 | Указывает количество дней до блокировки пользователя, когда его пароль станет недействителен |
| -b | Задает базовый каталог для домашнего каталога  useradd test-user -b /var/home |
| -D | Позволяет показать или изменить настройки по умолчанию, которые будут применяться при последующем создании пользователей  useradd test-user -Ds /bin/bash |
| -k | Путь к источнику скелета (файлы с шаблонами для нового пользователя)  useradd test-user -k /var/skel |
| -M | Не создавать домашний каталог  useradd test-user -M |
| -N | Не создавать основную группу с таким же именем, как у пользователя  useradd test-user -N |
| -o | Разрешает создание учетной записи с повторяющимся UID  useradd test-user -u 15 -o |
| -p | Задает пароль  useradd test-user -p pass |
| -R | Каталог, в который выполняется chroot  useradd test-user -R /var/chroot/home |
| -U | Имя группы будет таким же, как у пользователя  useradd test-user -U |

Пример использования:

useradd -m -u 666 -d /home/users/test-user -c "Тестовый пользователь" -e 2060-01-01 -s /bin/bash test-user

В этом примере создается тестовый пользователь test-user с идентификатором 666, домашней папкой в /home/users/test-user, комментарием "Тестовый пользователь" и доступом к командной оболочке. Учётная запись будет действительна до 1 января 2060 года.

Более подробную информацию о доступных опциях для useradd можно увидеть с помощью команды:

man useradd

Очень важно после создания пользователя настроить для него надёжный пароль командой (Рисунок 3):

passwd user

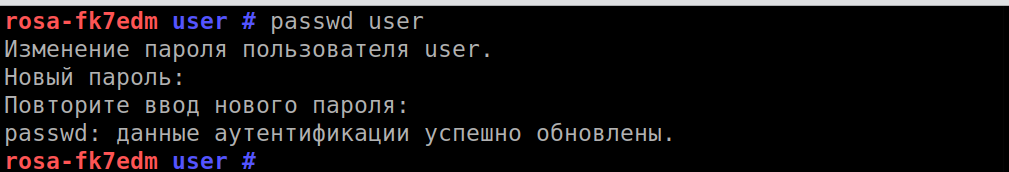


Рисунок 3 - Настройка пароля пользователя после его создания

Система предложит ввести и подтвердить пароль. На этом процесс создания пользователя можно считать завершённым.

* 1. Изменение настроек пользователя

Изменение настроек для активного пользователя может привести к сбою Системы, поэтому перед изменением данных важно убедиться, что в текущий момент редактируемый пользователь не авторизован, от его имени отсутствуют запущенные процессы, редактируемые файлы. Посмотреть список запущенных процессов пользователя user можно следующим образом:

pgrep -l -u user

Проверить, авторизован ли пользователь, можно с помощью команды:

pinky user

Отредактировать данные существующего пользователя можно с помощью команды usermod**.** По структуре она похожа на предыдущую команду:

usermod <имя пользователя> [опции]

Набор параметров расширен дополнительными опциями (Таблица 13).

Таблица 13 - Дополнительные опции команды usermod

|  |  |
| --- | --- |
| Параметр | Значение |
| -m | Создает новую директорию, указанную в качестве домашней (если её не существует), и переносит туда данные из старой |
| -d /home/users/new-test-user | Меняет домашнюю директорию пользователя на /home/users/new-test-user |
| -c "Изменение UID" | Меняет комментарий к пользователю |
| -a -G users,wheel,sudo | Добавляет пользователя в дополнительные группы |
| -s /bin/bash | Меняет командную оболочку пользователя |
| -u 100500 | Изменяет UIDпользователя |
| -e 2060-01-01 | Меняет дату, до которой аккаунт будет активен |
| -f 7 | Меняет количество дней до блокировки пользователя, когда его пароль станет недействителен |
| -l new-test-user | Меняет имя пользователя на new-test-user |
| -L | Блокирует аккаунт пользователя. Для этого в файле /etc/shadow перед хешем пароля пользователя ставится символ "!" |
| -U | Снимает блокировку с аккаунта (удаляет символ "!" из пароля в /etc/shadow) |

Таким образом, если необходимо отредактировать данные пользователяtest-user, созданного ранее, это будет выглядеть так:

usermod -l new-test-user -m -d /home/new-test-user -c " Изменение UID " -u 100500 -e 3000-01-01 -f -1 test-user

В примере изменены: логин имя пользователя наnew-test-user, домашняя папка на /home/new-test-user с копированием файлов, комментарий,UID пользователя, срок жизни аккаунта и блокировка в случае устаревания пароля.

При изменении данных пользователя Система предпримет попытку автоматически указать новые данные для всех файлов и папок пользователя. Тем не менее после редактирования нужно всё перепроверить и исправить права доступа, если они не изменились автоматически.

* 1. Удаление пользователя

Как и в случае с редактированием, перед удалением пользователя необходимо убедиться, что от его имени отсутствуют активные процессы, не редактируются файлы, иначе существует риск сбоя Системы. В программу встроен механизм защиты, поэтому она не позволит удалить пользователя, если он авторизован или от его имени работают какие-то службы.

Для удаления пользователей используется команда userdel. Её структура аналогична предыдущим:

userdel <имя пользователя> [опции]

Основные параметры команды userdel приведены ниже (Таблица 14).

Таблица 14 - Основные параметры команды userdel

|  |  |
| --- | --- |
| Параметр | Значение |
| -r | Удаляет папки пользователя: домашнюю директорию, почтовую очередь |
| -f | Отключает механизм защиты. При использовании этой опции пользователь будет удалён даже при наличии запущенных процессов и пр. Используется на свой страх и риск, так как может привести к сбою Системы |

После удаления пользователей важно вручную проверить, что на сервере не осталось файлов или директорий, принадлежащих удалённому пользователю:

man userdel

* 1. Блокировка пользователя

Блокировку пользователя можно выполнить, не удаляя его из Системы:

usermod -L <имя пользователя>

Например, чтобы заблокировать пользователя test-user, нужно выполнить команду:

usermod -L test-user

Чтобы разблокировать пользователя, выполняют следующую команду:

usermod -U <имя пользователя>

1. Группы пользователей
   1. Общие сведения

Информация о группах хранится в файле /etc/group**.**

Группы применяются для делегирования прав доступа на определённые файлы, папки, скрипты сразу нескольким пользователям.

В качестве примера можно рассмотреть работу группы пользователей с FTP-сервером. Для работы с файлами нужно выбрать какую-либо директорию, создать группу пользователей и присвоить группе выбранную директорию. При использовании группы не нужно отдельно настраивать права каждому новому пользователю достаточно добавить его в эту группу, и он автоматически получит доступ к FTP-каталогу.

* 1. Создание групп

Для создания групп используется командаgroupadd:

groupadd new-group

Параметры команды приведены ниже (Таблица 15).

Таблица 15 - Параметры команды groupadd

|  |  |
| --- | --- |
| Параметр | Значение |
| -f | Если группа с указанным именем или GID уже существует, опция прерывает выполнение команды без соответствующей ошибки |
| -g 100500 | Позволяет назначить свой GID для создаваемой группы |
| -r | Создает системную группу |
| -p p@ssw0rd | Устанавливает для группы парольp@ssw0rd. Пароль запрашивается системой при попытке входа в группу с помощью команды newgrp.  Не рекомендуется к использованию из-за проблем с безопасностью. Настроенный таким образом пароль можно увидеть в истории команд. |

* 1. Редактирование групп пользователей

Для редактирования групп используется команда groupmod. Список изменений задаётся с помощью параметров, перечисленных ниже (Таблица 16).

Таблица 16 - Параметры команды groupmod

|  |  |
| --- | --- |
| Параметр | Значение |
| -g 100500 | Меняет GID группы на 100500 |
| -n another-name | Меняет имя группы на another-name |

Например, если нужно изменить имя группыtest-group на имя named-group, команда будет выглядеть так:

groupmod -n named-group test-group

* 1. Удаление групп пользователей

Нельзя удалить группу, если она указана в качестве основной для какого-либо существующего пользователя. Предварительно необходимо удалить этого пользователя из группы.

Непосредственно удаление группы выполняется одной командой:

groupdel test-group

Как и в случае удаления пользователей, необходимо вручную проверить, что на сервере не осталось данных, принадлежащих удалённой группе.

* 1. Управление группами

Базовым инструментом для управления группами является утилита gpasswd. Большинство параметров команды (кроме-A и-M) не сочетаются между собой, то есть в команде может быть только один параметр при разовом выполнении.

Общий вид команды:

gpasswd [что сделать] [в какой группе]

Параметры команды приведены ниже (Таблица 17).

Таблица 17 - Параметры команды gpasswd

|  |  |
| --- | --- |
| Параметр | Значение |
| -a new-user | Добавляет пользователя new-user в группу |
| -d bad-user | Удаляет пользователя bad-user из группы |
| -A user1,user2,... | Доступна для использования привилегированным пользователям (с правами root). Назначает список пользователей-администраторов группы |
| -M user1,user2,... | Доступна для использования привилегированным пользователям. Назначает список участников группы |
| -r | Отключает пароль группы. После этого только члены группы смогут использовать команду newgrp для подключения к группе |
| -R | Отключает внешний доступ к группе. После этого только члены группы смогут использовать команду newgrpдля подключения к группе |

Если потребуется добавить пользователя в новую группу, достаточно будет использовать следующую команду:

gpasswd -a new-user test-group

Также для добавления пользователей в новую группу используется описанная выше команда usermod. Следующий пример добавляет пользователяtest-user в группуnew-group:

usermod -a -G new-group test-user

Или, если нужно указать группу new-group в качестве основной группы пользователя test-user:

usermod -g new-group test-user

Помимо этого, любой пользователь может сам авторизоваться и добавиться в новую группу с помощью команды:

newgrp new-group

Эта команда позволяет переключить группу пользователя в рамках текущей сессии, а также автоматически добавляет запрошенную группу в список групп пользователя.

Для определения в каких группах состоит пользователь, необходимо воспользоваться командой groups:

user@rosa:~$ groups

user adm cdrom sudo dip plugdev lxd lpadmin sambaShare

Из этого примера видно, что пользователь user состоит в группах user и т.д. Если требуется посмотреть, в каких группах состоит другой пользователь, то нужно передать его имя в качестве аргумента:

user@rosa:~$ groups root

root : root

* 1. Списки

Посмотреть список пользователей можно в файле /etc/passwd:

cat /etc/passwd

Вывод команды будет похож на следующий:

root:x:0:0:root:/root:/bin/bash

daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin

bin:x:2:2:bin:/bin:/usr/sbin/nologin

...

Как правило, большая часть данных пользователей является системными их UID меньше 1000и больше 60000.

Более функциональная команда для отображения содержимого файла passwd getent:

getent passwd

Можно найти пользователя по идентификатору:

getent passwd 1000

Получить список несистемных пользователей:

getent passwd {1000..60000}

Получить только список логинов несистемных учетных записей:

getent passwd {1000..60000} | awk -F: '{ print $1}'

1. Управление учетными записями

Для управления учетными записями (УЗ) в Системе используются следующие утилиты:

* whoami – показать имя текущего пользователя;
* who – показать активных пользователей в Системе;
* w – показать активных пользователей в Системе и их процессы;
* last – показать последние входы в Систему.

Ниже приведены примеры команд для работы с учетными записями пользователей.

* 1. Команда useradd

Создать пользователя, добавить его в группу и создать домашнюю директорию:

useradd test-user -G printer -m

Создать учетную запись с возможностью получения привилегий суперпользователя (командой "sudo su"):

useradd test-user -G wheel -m

Создать пользователя с определенными UID и GID (соответственно идентификаторы пользователя и группы):

useradd test-user -u 900 -g 950

где группа с используемым идентификатором (в данном примере 950) уже должна быть создана заранее с использованием команды groupadd.

Создать пользователя и указать путь к домашней директории:

useradd test-user -d /home/newdmosk

Создать учетную запись без возможности входа в Систему:

useradd test-user -s /sbin/nologin

* 1. Команда usermod

Потребовать сменить пароль при следующем входе в Систему:

chage -d 0 test-user

Поменять пользователю основную группу:

usermod test-user -g kdonewgroup

Задать пользователю дополнительную группу с правами root:

usermod test-user -G sudo

Добавить пользователя в группу:

usermod -a -G group test-user

* 1. Команда chage

Автоматическая блокировка учетной записи:

chage -E 2025-05-01 test-user

где указано, что учетная запись test-user перестанет действовать после 1 мая 2025 года.

Просмотр информации о дате окончания срока действия учетной записи:

chage -l test-user

Задание учетной записи статуса бессрочной:

chage -E -1 test-user

1. Привилегии
   1. Общие сведения

Разделение привилегий – одна из основных парадигм безопасности в ОС. Обычные пользователи работают с ограниченными привилегиями и могут влиять только на собственную рабочую среду, но не на Систему в целом.

Специальный пользователь с именем root имеет привилегии административной учетной записи без ограничений, действующих для обычных пользователей. Пользователи могут выполнять команды с привилегиями суперпользователя или root разными способами.

В данном разделе рассмотрены вопросы правильного и безопасного получения привилегий root, и редактирование файла/etc/sudoers**.**

* 1. Суперпользователь

Вход в систему от имени суперпользователя root по умолчанию заблокирован. Для повышения полномочий пользователя до административных рекомендуется использовать команду sudo с вводом пароля Администратора Системы. Первый созданный в Системе пользователь-администратор по умолчанию имеет право на использование этой команды и повышения своих прав.

* 1. Команда sudo

Команда sudo (superuser do) позволяет временно получить права суперпользователя для выполнения одной команды. Этот подход обеспечивает дополнительный уровень безопасности, так как пользователи не работают постоянно под учетной записью root, что снижает риски случайного или вредоносного изменения системы.

Пример базового использования команды sudo, в котором пользователь вводит необходимую команду, требующую прав суперпользователя, предварив её ключевым словом sudo:

sudo команда

Если пользователь входит в группу, имеющую соответствующие права, система запросит ввод пароля, после чего команда будет выполнена с правами администратора.

Для настройки прав доступа к sudo используется конфигурационный файл /etc/sudoers, который определяет, какие пользователи или группы могут использовать sudo и какие команды они могут выполнять. Редактирование этого файла должно происходить с использованием команды visudo, которая предотвращает ошибки при сохранении и синтаксические ошибки.

Для добавления пользователей в группу sudoers:

usermod -aG wheel username

Пользователи группы wheel по умолчанию имеют право выполнять команды с использованием sudo.

В целях безопасности рекомендуется ограничить список пользователей, имеющих доступ к команде sudo. Кроме того, можно настроить специальные правила для выполнения только определенных команд.

Пример команды, которая разрешает пользователю user выполнять только команду обновления системы:

user ALL=(ALL) NOPASSWD: /usr/bin/dnf update

Все команды, выполненные с использованием sudo, записываются в системный журнал. Логи можно просмотреть в файле /var/log/auth.log или /var/log/secure. Это позволяет отслеживать действия пользователей, имеющих права суперпользователя, и выявлять возможные попытки несанкционированного доступа.

Несколько практических рекомендаций по использованию sudo:

* sudo не используется для выполнения команд, не требующих привилегий суперпользователя;
* всегда проверяйте команду перед её выполнением с правами root;
* правила использования sudo должны быть настроены так, чтобы пользователи могли выполнять только те действия, которые необходимы для их работы.

Подробное описание всех команд приведено в man sudo.

* 1. Настройки файла sudoers

Файл /etc/sudoersоткрывается в любом выбранном текстовом редакторе.

Пример содержимого файла:

Defaults env\_reset

Defaults mail\_badpass

Defaults secure\_PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin"

root ALL=(ALL:ALL) ALL

%admin ALL=(ALL) ALL

%sudo ALL=(ALL:ALL) ALL

#includedir /etc/sudoers.d

Строки по умолчанию:

* Первая строка "Defaults env\_reset" сбрасывает среду терминала для удаления переменных пользователя. Эта мера безопасности используется для сброса потенциально опасных переменных среды в сеансе sudo**.**
* Вторая строка "Defaults mail\_badpass" предписывает Системе отправлять уведомления о неудачных попытках ввода пароля sudo для настроенного пользователя mailto. По умолчанию это учетная запись root.
* Третья строка, начинающаяся с "Defaults secure\_PATH=…", задает переменную PATH, которая будет использоваться для операций sudo. Это предотвращает использование пользовательских путей, которые могут быть вредоносными.

Четвертая строка, которая определяет для пользователя root привилегии sudo, отличается от предыдущих строк, определяет следующие значения полей в "root ALL=(ALL:ALL) ALL":

* root – имя пользователя, которое правило будет применять к root;
* первое "ALL" – данное правило применяется ко всем хостам;
* второе "ALL" – пользователь root может запускать команды от лица всех пользователей;
* третье "ALL" – пользователь root может запускать команды от лица всех групп;
* четвертое "ALL" – данные правила применяются всем командам.

Это означает, что пользователь root сможет выполнять любые команды с помощью sudo после ввода пароля.

Следующие две строки похожи на строки привилегий пользователя, но задают правила sudo для групп. Имена, начинающиеся с **"%"**, означают названия групп.

В этом примере группа admin может выполнять любые команды от имени любого пользователя на любом хосте. Группа sudo имеет те же привилегии, но может выполнять команды от лица любой группы.

Последняя строка в файле выглядит как комментарий:

/etc/sudoers

...

#includedir /etc/sudoers.d

Однако данная строка означает, что файлы в каталоге /etc/sudoers.d также рассматриваются как источники и применяются.

Файлы в этом каталоге следуют тем же правилам, что и сам файл /etc/sudoers. Любой файл, который не заканчивается на **"~"** и не содержит символ ".", также считывается и добавляется в конфигурацию sudo.

В основном это нужно, чтобы приложения могли изменять привилегии sudo после установки. Размещение всех правил в одном файле в каталоге /etc/sudoers.d позволяет видеть, какие привилегии присвоены определенным учетным записям, а также легко изменять учетные данные без прямого изменения файла /etc/sudoers.

Как и в случае с файлом /etc/sudoers, другие файлы в каталоге /etc/sudoers.d также следует редактировать с помощью команды visudo. Для редактирования этих файлов применяется следующий синтаксис:

sudo visudo -f /etc/sudoers.d/file\_to\_edit

* 1. Присвоение привилегий

Чаще всего при управлении разрешениями sudo используется операция предоставления новому пользователю общего доступа sudo. Это полезно, если требуется предоставить учетной записи полный административный доступ к Системе.

Добавляя пользователя в группу администраторы, ему предоставляются такие же привилегии:

sudo usermod -aG sudo username

Также можно использовать командуgpasswd:

sudo gpasswd -a username sudo

Обе команды выполняют одно и то же действие.

В ОС РОСА "ХРОМ" эта группа называетсяwheel, а не sudo:

sudo usermod -aG wheel username

Также можно использовать gpasswd**:**

sudo gpasswd -a username wheel

Если в ОС РОСА "ХРОМ" добавление пользователя в группу не срабатывает сразу же, может потребоваться отредактировать файл /etc/sudoers, чтобы убрать символ комментария перед именем группы:

sudo visudo

/etc/sudoers

...

%wheel ALL=(ALL) ALL

...

* 1. Создание правил в файле sudoers
     1. Группировка

Файл /etc/sudoers можно организовать более эффективно, группируя элементы с помощью разнообразных псевдонимов.

Например, можно создать три разных группы пользователей с некоторыми общими участниками:

...

User\_Alias GROUPONE = abby, brent, carl

User\_Alias GROUPTWO = brent, doris, eric,

User\_Alias GROUPTHREE = doris, felicia, grant

...

Имена групп должны начинаться с заглавной буквы. Затем можем дать участникам группы GROUPTWOразрешение на обновление базы данных dnf, создав следующее правило:

...

GROUPTWO ALL = /usr/bin/dnf update

...

Если не указать пользователя или группу для запуска, команда sudo по умолчанию использует пользователя root.

* + 1. Свойства

Есть ряд способов, которые позволяют более точно контролировать реакцию sudo на вызов.

Команда updatedb, связанная с пакетом mlocate**,** относительно безопасна при ее выполнении в Системе с одним пользователем. Если нужно разрешить пользователям выполнять ее с привилегиями root без ввода пароля, можно создать правило следующего вида:

...

GROUPONE ALL = NOPASSWD: /usr/bin/updatedb

...

NOPASSWD – это свойство, означающее, что пароль не запрашивается. У него есть сопутствующее свойство PASSWD, которое используется по умолчанию и требует ввода пароля. Данное свойство актуально для остальной части строки, если его действие не переопределяется дублирующим тегом в этой же строке.

Например, можно использовать следующую строку:

...

GROUPTWO ALL = NOPASSWD: /usr/bin/updatedb, PASSWD: /bin/kill

...

Также полезно свойство NOEXEC**,** которое можно использовать для предотвращения опасного поведения некоторых программ.

Например, некоторые программы, такие как less**,** могут активировать другие команды, вводя их через свой интерфейс:

!command\_to\_run

При этом все команды пользователя выполняются с теми же разрешениями, что и командаless, что может быть довольно опасно.

Чтобы ограничить такое поведение, можно использовать следующую строку:

...

username ALL = NOEXEC: /usr/bin/less

...

1. Аутентификация

В ОС можно использовать несколько различных схем аутентификации. Наиболее часто используемая и стандартная схема аутентификация по файлам/etc/passwdи/etc/shadow**.**

* 1. Файл /etc/passwd/

/etc/passwd – это база данных на основе обычного текста, которая содержит информацию для всех учетных записей пользователей в Системе. Он принадлежит пользователю rootи имеет права 644. Файл может быть изменен только пользователем root или пользователями с привилегиями sudo и доступен для чтения всем пользователям Системы.

Следует избегать изменения файла /etc/passwd вручную без уверенности в своих действиях. Рекомендуется всегда использовать команду, предназначенную для этой цели.

Например, чтобы изменить учетную запись пользователя, используют командуusermod, а чтобы добавить новую учетную запись пользователя, используют команду useradd**.**

Файл/etc/passwd – это текстовый файл с одной записью в строке, представляющий учетную запись пользователя. Чтобы просмотреть содержимое файла, используют текстовый редактор или команду, например cat:

cat /etc/passwd

Обычно первая строка описывает пользователя root, за ней следуют учетные записи системного и обычного пользователей. Новые записи добавляются в конец файла.

Каждая строка файла /etc/passwd содержит семь полей, разделенных запятыми:

rosa:x:1001:1001:rosa,,,:/home/user:/bin/bash

[--] [--] [--] [-----] [--------] [--------]

| | | | | | |

| | | | | | +-> 7. Login shell

| | | | | +----------> 6. Home directory

| | | | +--------------------> 5. GECOS

| | | +--------------------------> 4. GID

| | +-------------------------------> 3. UID

| +-----------------------------------> 2. Password

+----------------------------------------> 1. Username

* **Имя пользователя** – Логин при входе в Систему. Каждое имя пользователя должно быть уникальной строкой на ПК. Максимальная длина имени пользователя ограничена 32 символами.
* **Пароль –** В старых Linux-подобных системах зашифрованный пароль пользователя хранился в /etc/passwd. В ОС для этого поля установлено значение "x" а пароль пользователя хранится в/etc/shadow.
* **UID** – Идентификатор пользователя – это номер, присвоенный каждому пользователю. Он используется ОС для обозначения пользователя.
* **GID** – Номер идентификатора группы пользователя, относящийся к основной группе пользователя. Когда пользователь создает файл, группа файла устанавливается в эту группу. Обычно название группы совпадает с именем пользователя. Вторичные группы пользователей перечислены в /etc/groups**.**
* **GECOS** – Это поле содержит список значений, разделенных запятыми, со следующей информацией:
* полное имя пользователя или название приложения;
* номер комнаты;
* рабочий номер телефона;
* номер домашнего телефона;
* другая контактная информация.
* **Домашний каталог –** Абсолютный путь к домашнему каталогу пользователя. Он содержит файлы и конфигурации пользователя. По умолчанию домашние каталоги пользователей названы по имени пользователя и создаются в каталоге /home.
* **Оболочка входа** – Абсолютный путь к оболочке входа пользователя. Это оболочка, которая запускается, когда пользователь входит в Систему. В большинстве дистрибутивов Linux оболочкой входа по умолчанию является Bash.
  1. Файл /etc/shadow/

/etc/shadow– это текстовый файл, содержащий информацию о паролях пользователей Системы. Он принадлежит пользователю root и теневой группе и имеет права 640.

Файл содержит по одной записи в каждой строке, каждая из которых представляет учетную запись пользователя. Возможно просмотреть содержимое файла с помощью текстового редактора или такой команды, какcat :

sudo cat /etc/shadow

Обычно первая строка описывает пользователя root, за которым следуют учетные записи системного и обычного пользователя. Новые записи добавляются в конец файла.

Каждая строка файла /etc/shadow содержит девять полей, разделенных запятыми:

rosa:$6$.n.:17736:0:99999:7:::

[--] [----] [---] [---] ----

| | | | | |||+-----> 9. Unused

| | | | | ||+------> 8. Expiration date

| | | | | |+-------> 7. Inactivity period

| | | | | +--------> 6. Warning period

| | | | +------------> 5. Maximum password age

| | | +----------------> 4. Minimum password age

| | +--------------------> 3. Last password change

| +---------------------------> 2. Encrypted Password

+----------------------------------> 1. Username

* **Имя пользователя** – Строка, которую вводят при входе в систему; учетная запись пользователя, существующая в Системе.
* **Зашифрованный пароль** – Пароль использует формат $type$salt$hashed. $type – это алгоритм криптографического хеширования метода и может иметь следующие значения:
* **$1$** –5 MDL;
* **$2a$** – Blowfish;
* **$2y$** –Eksblowfish;
* **$5$** – SHA-256;
* **$6$** – SHA-512.

Если в поле пароля есть звездочка "\*" или восклицательный знак "!", пользователь не сможет войти в Систему, используя аутентификацию по паролю. Другие методы входа в Систему, такие как аутентификация на основе ключей или переключение на пользователя, по-прежнему разрешены.

* **Последнее изменение пароля** **–** Это дата последней смены пароля. Число дней отсчитывается с 1 января 1970 года (дата эпохи).
* **Минимальный возраст пароля** **–** Количество дней, которое должно пройти, прежде чем можно будет изменить пароль пользователя. Обычно он равен нулю, что означает отсутствие минимального возраста пароля.
* **Максимальный возраст пароля** **–** Количество дней после смены пароля пользователя. По умолчанию этот номер установлен на 99999.
* **Период предупреждения** **–** Количество дней до истечения срока действия пароля, в течение которых пользователя предупреждают о необходимости изменения пароля.
* **Период бездействия** **–** Количество дней после истечения срока действия пароля пользователя до отключения учетной записи пользователя. Обычно это поле пусто.
* **Дата окончания срока** **–** Дата, когда учетная запись была отключена. Он представлен как дата эпохи.
* **Не используется** **–** Это поле игнорируется. Оно зарезервировано для использования в будущем.

Файл/etc/shadow не следует редактировать вручную, если нет уверенности в своих действиях. Всегда используют команду, предназначенную для этой цели. Например, чтобы изменить пароль пользователя, используют команду passwd, а чтобы изменить информацию об устаревании пароля, используют команду chage.

Например, запись в файле /etc/shadow:

linuxize:$6$zHvrJMa5Y690smbQ$z5zdL...:18009:0:120:7:14:

содержит следующие сведения о пользователе:

* информация о пароле пользователя linuxize;
* пароль зашифрован с помощью SHA-512 (пароль усечен для лучшей читаемости);
* последний раз пароль изменялся 23 апреля 2019 г. 18009;
* нет минимального возраста пароля;
* пароль необходимо менять не реже, чем каждые 120 дней;
* пользователь получит предупреждающее сообщение за семь дней до истечения срока действия пароля;
* если пользователь не попытается войти в систему через 14 дней после истечения срока действия пароля, учетная запись будет отключена;
* срок действия учетной записи отсутствует.
  1. Права доступа

Для пользователей и групп пользователей настраиваются права доступа. Общая схема прав одинакова и для файлов, и для папок.

Система отображает права доступа в выводе команды "ls -l", которая показывает содержимое директории (Рисунок 4):

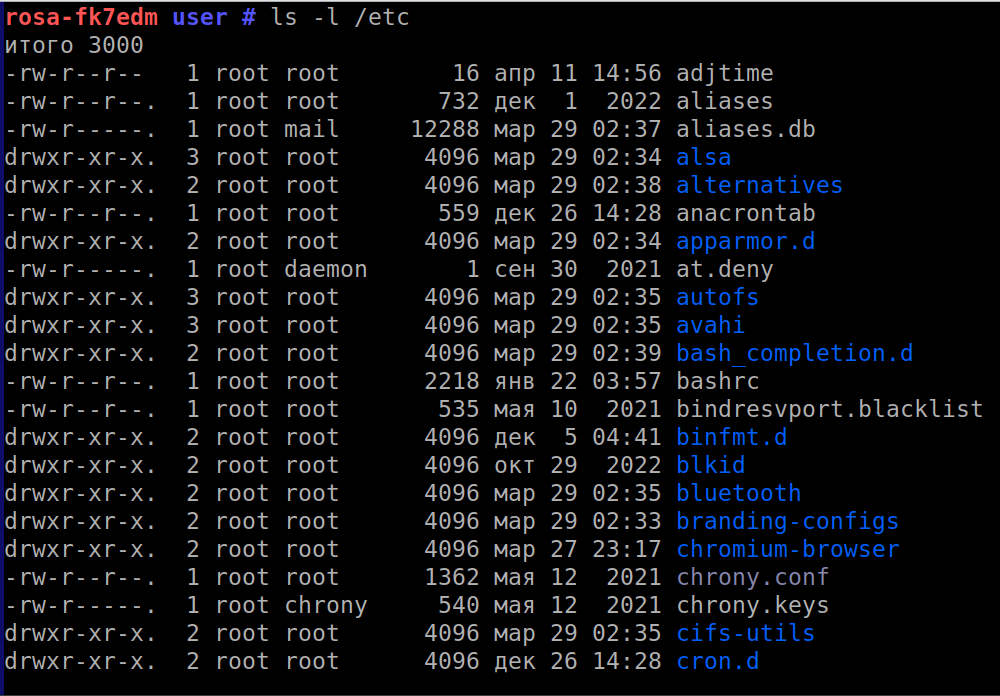


Рисунок 4 - Вывод команды "ls -l"

Первые 10 символов в строке содержат информацию о правах к файлу или каталогу (Таблица 18).

Таблица 18 - Информация о правах

|  |  |  |  |
| --- | --- | --- | --- |
| **-** | **rw-** | **r--** | **r--** |
| тип файла | права пользователя-владельца | права пользователей группы-владельца | права всех остальных пользователей |

Далее следует имя пользователя-владельца и группы-владельца (Таблица 19).

Таблица 19 - Имена пользователя и группы

|  |  |
| --- | --- |
| **root** | **root** |
| файл принадлежит пользователю **root** | файл принадлежит группе **root** |

Исходя из этого, иерархия доступа на уровне пользователей разбивается следующим образом: права пользователя-владельца, права участников группы-владельца и права всех остальных.

Соответственно, для каждой категории указывается, какие операции с файлом ей доступны: **чтение (r)**, **запись (w)** или **выполнение (x)** для исполняемых файлов. Для директорий параметры те же, но обозначают немного другое: **просмотр директории (r)**, **создание папок / файлов (w)** внутри директории, **переход в директорию (x)**.

Каждый из этих уровней доступа можно выразить в восьмеричной системе с помощью числового значения: 4 (r), 2 (w), 1 (x) (Таблица 20).

Таблица 20 - Схема прав

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **a** – любые пользователи | | | | | | | | |
| **u** – права пользователя | | | **g** – права группы | | | **o** – права всех остальных | | |
| **r** | **w** | **x** | **r** | **w** | **x** | **r** | **w** | **x** |
| **4** | **2** | **1** | **4** | **2** | **1** | **4** | **2** | **1** |

Для отображения уровня прав помимо формата rwxrwxrwxиспользуется упомянутый восьмеричный формат. Для этого достаточно сложить все уровни прав по категориям:

rwxrwxrwx = (4+2+1), (4+2+1), (4+2+1) = 777

Теперь можно перейти к инструментам, которые позволяют управлять правами на файлы и директории:

* 7 = 4 + 2 + 1 (read/write/execute);
* 6 = 4 + 2 (read/write);
* 5 = 4 + 1 (read/execute);
* 4 = 4 (read);
* 3 = 2 + 1 (write/execute);
* 2 = 2 (write);
* 1 = 1 (execute).
  1. Смена владельца

Для смены владельца доступны два инструмента. Первый chown позволяет изменить пользователя и группу файла или папки:

chown [новый пользователь]:[новая группа] [файл или папка]

Если не указывать группу, изменится только пользователь-владелец. Если не указывать пользователя **(**:[новая группа]), изменится только группа-владелец.

При изменении прав на директории можно использовать параметр -R. Он рекурсивно изменит владельца всех вложенных директорий и файлов. Если представить, что нужно предоставить права на папку new-user для одноимённого пользователя и его группы, то команда для этого будет выглядеть так:

chown -R new-user:new-group /home/users/new-user

Второй инструмент chgrp. В отличие от первого, меняет только группу-владельца:

chgrp [группа] [файл или папка]

Список распространённых параметров команды chgrp:

* **-h** – работа непосредственно с самими символьными ссылками;
* **--dereference** – работа с файлами, а не самими символьными ссылками (используется по умолчанию);
* **-R** – рекурсивная обработка каталога со всем его содержимым;
* **-H** – переход по символической ссылке и изменение атрибутов файла/каталога; используется вместе с параметром -R;
* **-L** – переход по символической ссылке и продолжение рекурсивной обработки; используется вместе с параметром -R;
* **-P** – обработка только символической ссылки при наличии; используется вместе с параметром -R; является значением по умолчанию;
* **--reference=имя\_образца** – использование группы образца; используется вместо группы;
* **-c** – вывод при обработке только изменений;
* **-v** – вывод информации о каждом обработанном объекте.
  1. Смена прав

Для работы непосредственно с правами используется команда chmod:

chmod [настройки прав] [файл или папка]

При работе с директориями опция -R позволит изменить права на все вложенные файлы и папки.

Настройки прав вchmod можно определять двумя способами:

1. **первый способ** – указать категорию (**u** пользователь-владелец, **g** группа-владелец, **o** другие пользователи, **a** все пользователи), модификатор (**+, , =**) и, соответственно, нужные права (**r, w, x**).

Например, если есть файл example.txt с максимальным уровнем прав для всех категорий пользователей (Рисунок 5).

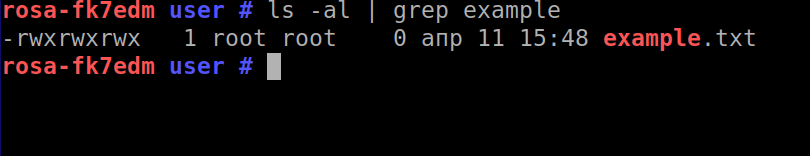


Рисунок 5 - Изменение прав на файл example.txt

Если не нужно, чтобы кто-то мог запускать этот файл на выполнение, нужно убрать параметр"x" из прав всех категорий пользователей сразу. Это можно сделать так (Рисунок 6):

chmod a-x example.txt

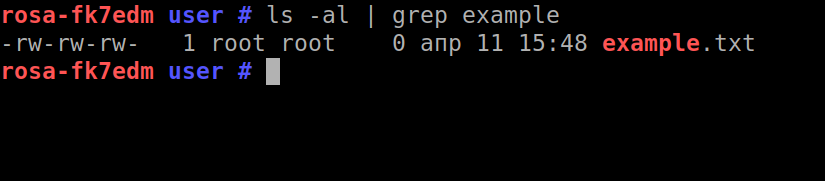


Рисунок 6 - Изменение прав на файл example.txt с запретом на выполнение для всех пользователей

Теперь, если необходимо вернуть владельцу права на запуск файла, следует добавить параметр "x" в категорию пользователя-владельца (Рисунок 7):

chmod u+x example.txt

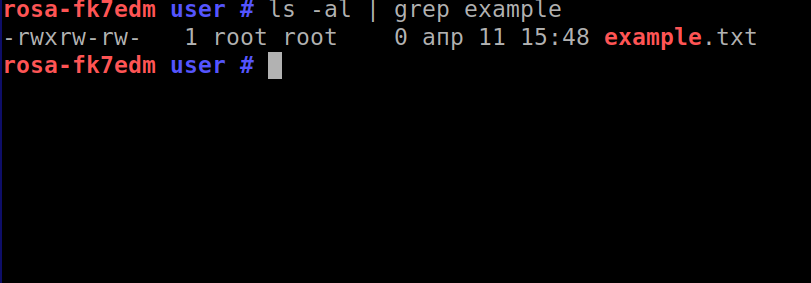


Рисунок 7 - Изменение прав на файл example.txt с добавлением критерия пользователю-владельцу

Для изменения всего набор параметров для отдельной категории сразу (Рисунок 8):

chmod u=rwx example.txt

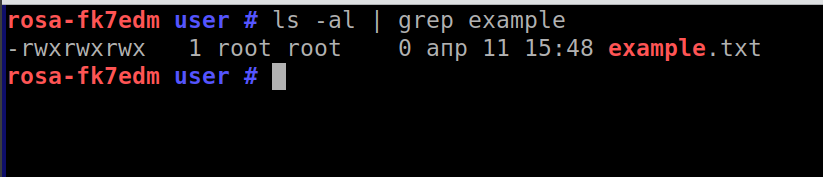


Рисунок 8 - Изменение прав на файл example.txt для отдельной категории пользователей

1. **второй способ** – указать права в виде числового значения, что является более сложным, но более быстрым способом. В этом случае в качестве параметра нужно передать цифровое выражение уровня прав:

chmod 777 example.txt

В этом примере назначен максимальный уровень доступа всем категориям пользователей.

Соответственно, для изменения уровня прав отдельной категории пользователей нужно изменить только это числовое значение. Например, чтобы запретить исполнение файла для всех пользователей:

chmod 666 example.txt

Чтобы вернуть права, но только пользователю-владельцу:

chmod 766 example.txt

Если необходимо изменить права только на все папки или только на все файлы в определённой директории, например при настройке прав на файлы сайтов, можно использовать один из вариантов:

* Смена прав на 755 для всех папок внутри /var/www/sites/example.com:

find /var/www/sites/example.com/ -type d -exec chmod 755 {} \;

* Смена прав на 644 для всех файлов внутри /var/www/sites/example.com:

find /var/www/sites/example.com/ -type f -exec chmod 644 {} \;

* 1. Изменение атрибутов

Помимо прав доступа и владельца каждый файл может иметь ряд атрибутов, определяемых на уровне файловой системы. Атрибуты показывают, какие операции могут или не могут проводиться с файлом независимо от владельца.

Посмотреть атрибуты файлов в текущей директории можно с помощью команды lsattr. Если запустить её без аргументов, она выведет атрибуты всех файлов в текущей директории. Если указать путь к файлу или папке, она перечислит свойства указанного файла или списка файлов в указанной папке соответственно (Рисунок 9):

lsattr example.txt

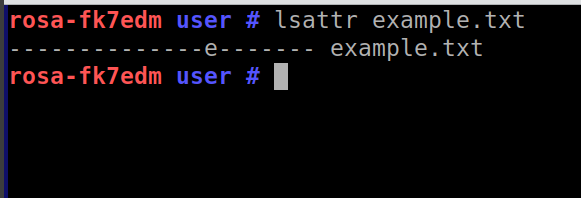


Рисунок 9 - Использование команды lsattr

Первые 20 символов в строке предназначены для отображения атрибутов файла.

Список атрибутов может отличаться в зависимости от файловой системы. Список основных представлен ниже (Таблица 21).

Таблица 21 - Атрибуты файлов

|  |  |
| --- | --- |
| Атрибут | Описание атрибута |
| i | "godmode" файл становится неуязвим для любых изменений. Его нельзя удалить, переименовать, изменить содержимое, создать символьную ссылку на него |
| a | В файл с таким атрибутом можно только добавлять новые данные. Старое содержимое изменить или удалить не получится. Это пригодится для защиты от вредоносных вставок или замен в файлах, куда постоянно записываются данные, например, в логах. Доступ к изменению старого содержимого по умолчанию есть только у суперпользователя root |
| s | Активирует безвозвратное удаление файла в том смысле, что после удаления файл нельзя будет восстановить с носителя. При удалении все использовавшиеся для хранения файла блоки на диске перезаписываются нулями. |
| u | Активирует "обратимое" удаление файлов. Это значит, что при удалении файла с этим атрибутом его содержимое можно восстановить. |
| с | Сжатый файл. Все данные, записываемые в файл, автоматически сжимаются, а данные, извлекаемые из файла, возвращаются в исходное состояние. |
| d | Настраивает для файла исключение при использовании утилиты dump, то есть файл не будет включен в архив при создании резервной копии этим способом. |
| e | Показывает, что файл в качестве указателей использует экстенты |
| j | В журналируемых файловых системах (ext3, ext4) указывает на то, что при сохранении файла он сначала будет записан в журнал файловой системы, и только потом на диск |
| A | Указывает, что при работе с файлом Система не будет обновлять информацию о времени доступа к нему |
| D | Атрибут для директорий. Указывает, что все изменения в папке синхронно записываются на диск, минуя кеш |
| S | Указывает, что все изменения в файле с этим атрибутом записываются синхронно на диск, минуя кеш |

Изменить атрибуты файла позволяет команда chattr:

chattr [модификатор][изменяемые атрибуты] [целевой файл или папка]

Таким образом, если необходимо защитить какой-то важный файл от посягательств, можно использовать такую команду (Рисунок 10):

chattr +i example.txt

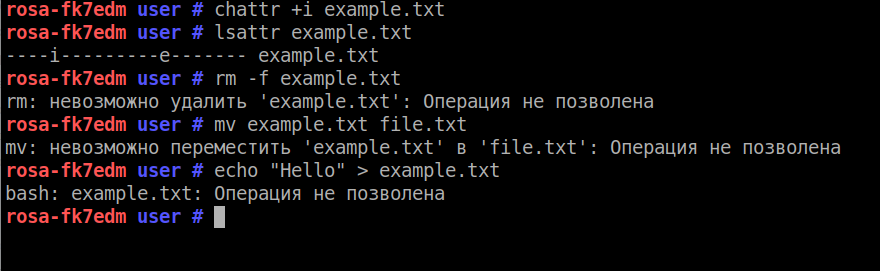


Рисунок 10 - Проверка выполнения действий над файлом example.txt

На рисунке выше видно, что теперь выполнение действий над файлом example.txt не возможно.

Если же нужно вернуть файл в нормальное состояние, необходимо выполнить обратную операцию:

chattr -i example.txt

Для просмотра более подробной информации о файловых атрибутах, их ограничениях и правилах применения используют команду:

man chattr

* + 1. SUID (Set User ID)

SUID – это атрибут исполняемого файла, позволяющий запустить его с правами владельца. В ОС РОСА "ХРОМ" приложение запускается с правами пользователя, запустившего указанное приложение. Это обеспечивает дополнительную безопасность, т.к. процесс с правами пользователя не сможет получить доступ на запись к важным системным файлам, например /etc/passwd, который принадлежит суперпользователю root.

Если на исполняемый файл установлен атрибут suid, то при выполнении эта программа автоматически меняет "эффективный userID" на идентификатор того пользователя, который является владельцем этого файла. Таким образом, независимо от того, кто запускает эту программу, она при выполнении имеет права хозяина этого файла.

* + 1. SGID (Set Group ID)

Атрибут SGID аналогичен SUID, но относится к группе. При этом, если для каталога установлен бит SGID, то создаваемые в нем объекты будут получать группу владельца каталога, а не пользователя.

* + 1. UMASK

Если процесс создает новый файл, он указывает, какие права доступа нужно задать для данного файла. Зачастую запрашиваются права 0666 (чтение и запись всеми), что дает больше разрешений, чем необходимо в большинстве случаев. Но при этом каждый раз, когда создается новый файл, Система обращается к параметру, называемому umask. Система использует значение umask, чтобы понизить изначально задаваемые разрешения на более разумные и безопасные. Просмотреть текущие настройки umask можно, набрав в командной строке:

user@user:~$ umask

0002

В ОС РОСА "ХРОМ" значением по умолчанию для umask является 0022, что позволяет другим пользователям читать новые файлы (если они могут до них добраться), но не изменять их. Чтобы автоматически обеспечивать больший уровень защищенности для создаваемых файлов, можно изменить настройки umask:

user@user:~$ umask 0077

Такое значениеumaskприведет к тому, что группа и прочие пользователи не будут иметь совершенно никаких прав доступа для всех вновь созданных файлов.

В отличие от "обычного" назначения прав доступа к файлу, umask задает, какие права доступа должны быть отключены. Таблица 22 показывает соответствия значений чисел и прав.

Таблица 22 - Соответствие чисел и прав

|  |  |
| --- | --- |
| Права | Число |
| rwx | 7 |
| rw- | 6 |
| r-x | 5 |
| r-- | 4 |
| -wx | 3 |
| -w- | 2 |
| --x | 1 |
| --- | 0 |

Воспользовавшись этой таблицей, можно определить, что последние три знака в 0077 обозначают –rwxrwx. umask показывает Системе, какие права доступа следует отключить. Совместив первое и второе становится видно, что все права для группы и остальных пользователей будут отключены, в то время как права владельца останутся нетронутыми.

* + 1. Изменение SUID и SGID

Способ установки и удаления битов suid и sgid:

* чтобы задать бит suid:

chmod u+s <путь к файлу>

* чтобы задать бит sgid:

chmod g+s <путь к директории>

1. Контроль доступа
   1. Общие сведения

Списки контроля доступа (Access Control Lists, ACL) являются типичным способом управления пользователями и группами в ОС.

ACL позволяют установить для файла или каталога любое количество правил, касающихся того, какие пользователи или группы имеют право читать или изменять файл или группу файлов, находящихся в каталоге, а какие пользователи не имеют такого права. Таким образом, ACL дополняют стандартные права доступа и могут вводить дополнительные права или отменять существующие.

Для опытных администраторов крупных сетей списки контроля доступа могут обеспечить дополнительную безопасность или как минимум упростить управление сетью, но для большинства пользователей ОС обычного метода управления правами будет вполне достаточно. Если неправильно использовать сложную систему списков контроля доступа, то могут возникнуть всевозможные бреши в сфере безопасности сети. Поэтому в настоящее время и не существует дистрибутивов, в которых списки контроля доступа использовались бы по умолчанию.

Расширенные атрибуты (Extended Attributes, EA) находятся в тесном родстве со списками контроля доступа. Они позволяют сохранять для любых файлов дополнительные сведения в форме пар "атрибут – значение". Например, можно присвоить текстовому файлу атрибут charset со значением utf8, чтобы сохранить таким образом применяемую кодировку. Конечно, это выгодно только в случае, если существуют программы, способные интерпретировать такую информацию. В зависимости от используемой файловой системой расширенные атрибуты необходимо активировать с помощью соответствующего mount-параметра; например, в случае с файловой системой ext – это параметр *user\_xattr*.

Более подробная информация о применении списков контроля доступа и расширенных атрибутов содержится на страницах справки man (acl, getfacl, setfacl, attr, getfattr, getsattr)**.**

* 1. Списки контроля доступа
     1. Команды getfacl и setfacl

В Системах с установленными списками контроля доступа обычно все равно действуют стандартные принципы управления доступом, которые часто называются минимальным списком контроля доступа. Команда getfacl отображает эти права в виде ACL:

user$ touch файл1

user$ getfacl файл1

# file: файл1

# owner: kofler

# group: kofler

user::rw-

group::r--

other::r--

user$ ls -l файл1

-rw-r--r-- 1 kofler... файл2

С помощью setfaclможно определить новые права доступа. Следующие команды предоставляют пользователю gabi и всем членам группы docuteam доступ к файлу с правом читать и изменять этот файл, однако закрывают пользователю kathrin какой-либо доступ к файлу:

user$ setfacl -m gabi:rw файл1

user$ setfacl -m g:docuteam:rw файл1

user$ setfacl -m kathrin:- файл1

Список прав команды getfaclнемного длиннее. Теперь lsпоказывает на месте прав доступа для членов группы специальную ACL-маску. За буквами, означающими доступ, следует символ "+", указывающий на наличие правил ACL.

user$ getfacl файл1

# file: файл1

# owner: kofler

# group: kofler

user::rw-

user:gabi:rw-

user:kathrin:---

group::r--

group:docuteam:rw-

mask::rw-

other::r--

user$ ls -l файл1

-rw-rw-r--+ 1 kofler... файл1

Как правило, списки контроля доступа применяются для того, чтобы предоставить определенному пользователю доступ к его файлам, не давая такого доступа всем остальным пользователям. В таком случае потребуется обратиться к администратору, чтобы он создал группу, к которой будет принадлежать сам пользователь и те пользователи, вместе с которыми предполагается обрабатывать файлы. При использовании ACL нужно выполнить команду:

setfacl -m пользователь:rw файл.

* + 1. Маски

Маска ограничивает права, предоставляемые правилами ACL. Если, например, для маски ACL устанавливается значение "r", то ни одно правило не может дать пользователю права изменять или выполнять файлы. Таким образом, ACL-маска имеет приоритет над правилами ACL. Но в любом случае она никак не влияет на обычные права доступа, которые предоставляются пользователю или группе пользователей традиционным способом.

При изменении правила ACL командой setfaclмаска автоматически рассчитывается заново так, чтобы могли выполняться все прочие правила ACL. Данная маска отображается с помощью команды getfaclи также учитывается командой:

ls -l

Можно явным образом настроить маску с помощью команды:

setfacl -m:rwx file

ограничив, таким образом, права ACL. Следует учитывать, что маска действует лишь до тех пор, пока не будет определено новое правило ACL. В таком случае маска ACL будет автоматически пересчитана (если только не запрещено с помощью параметра -n).

* + 1. Наследование

При работе с каталогами можно установить второй набор правил для стандартных списков контроля доступа. Стандартные ACL не управляют доступом к каталогу, а служат образцом для новых файлов. Любой файл, создаваемый внутри данного каталога, наследует стандартные списки контроля доступа, указанные для этого каталога. При одновременном применении нескольких списков контроля доступа новый каталог с правильно подобранными стандартными ACL может применяться в работе в качестве отправной точки.

* + 1. Совместимость

Распространение списков контроля доступа сильно осложняется тем, что многие стандартные команды и практически все пользовательские программы просто игнорируют ACL. Если просто скопировать файл с правилами ACL с помощью команды cp**,** то в скопированном файле правила ACL не сохранятся. То же самое произойдет, если открыть файл в текстовом редакторе OpenOffice или Gimp и сохранить его под другим именем. При использовании команды cp будет полезен параметр -p, но у большинства других команд и программ подобные параметры отсутствуют, либо эти программы вообще не приспособлены к работе с ACL.

Проблемы возникают и при резервном копировании. Команды tarи rsyncэлиминируют правила ACL. Файловая система CD и DVD не рассчитана на работу с ACL, так что сохраняемая в них информация теряется.

Есть два выхода: во-первых, можно применять вместо tarверсию star, совместимую с ACL, во-вторых, перед резервным копированием можно создавать отдельный текстовый файл, в который заносятся ACL-правила всех копируемых файлов. После резервного копирования правил ACL восстанавливаются на основании этого файла.

getfacl -R --skip-base. > acl-backup (Сохранение ACL-правил)

setfacl --restore=acl-backup (Восстановление ACL-правил)

* 1. Расширенные атрибуты
     1. Команды setfattr и getfattr

На следующих примерах показано, как сохранять атрибуты с помощью команды setfattrи считывать их с использованием getfattr:

user$ touch файл2

user$ setfattr -n user.language -v ru файл2

user$ setfattr --name=user.charset --value=utf8 файл2

user$ getfattr -d файл2

# file: файл2

user.charset="utf8"

user.language="ru"

Команда getfattrобычно возвращает только те атрибуты, чье название начинается с user. Если требуется увидеть другие атрибуты, нужно указывать имя атрибута вместе с меткой -n, а образец атрибута с меткой -m.

* + 1. Совместимость с расширенными атрибутами

К сожалению, пока практически не существует программ, которые сохраняли бы расширенные атрибуты при копировании, архивировании и т. д. Даже команда

cp -p

игнорирует атрибуты. При создании резервных копий лучше всего поступать так же, как и при работе с ACL создавать при копировании текстовый файл со всеми расширенными атрибутами. На основании этого файла можно потом восстановить расширенные атрибуты:

user$ getfattr -R. > ea-backup (Сохранение атрибутов)

user$ setfattr --restore=ea-backup (Восстановление атрибутов)

1. Процессы
   1. Общие сведения

В общем представлении **процесс** – это программа, выполняющаяся в оперативной памяти компьютера.

В многозадачной Системе может быть запущено множество программ. Каждая программа может запустить множество процессов (подпрограмм). При этом в единственный момент времени на ПК выполняется только один процесс.

То есть в единственный момент времени ресурсы железа (процессорное время, память, порт ввода/вывода) могут использоваться только единственным процессом. Очередью, в которой процессу выделяется определенный ресурс "железа", управляет планировщик. При этом во время прерывания одного процесса и запуска (возобновления) другого процесса состояние процесса (выполняемые действия, на каком этапе процесс приостановлен) запоминается и записывается в область памяти.

**Планировщик в ОС** – это часть ядра, отвечающая за указанную функциональность. В задачи планировщика также входит отслеживание и выделение запускаемым процессам определенного приоритета, чтобы процессы "не мешали" друг-другу работать, а также распределение пространства памяти, чтобы пространство памяти одного процесса не пересекалось с пространством другого.

Все новые процессы в ОС порождаются клонированием какого-то уже имеющегося процесса с помощью вызова системных функций clone(2*)* и fork(2). У нового (порожденного или дочернего) процесса то же окружение, что и у родителя; отличается только номер ID процесса (так называемый PID).

* 1. Жизненный цикл процесса

Алгоритм работы процессов в ОС определяется следующими условиями:

1. имеется запущенный процесс;
2. каждый процесс может запускать подпроцессы;
3. создание нового процесса создается клонированием исходного;
4. прародителем всех процессов в системе является процесс init, запускаемый ядром ОС при загрузке;
5. процессы взаимодействуют между собой посредством межпроцессного взаимодействия:

* каналы;
* сигналы;
* сокеты;
* разделяемая память;

1. каждый процесс обладает свойствами (процесс обладает контекстом):

* PID – идентификатор процесса;
* PPID – идентификатор процесса, породившего данный процесс;
* UID и GID – идентификаторы прав процесса (соответствует UID и GID пользователя, от имени которого запущен процесс);
* приоритет процесса;
* состояние процесса (выполнение, сон и т.п.).

Типичный процесс в Linux может быть представлен схемой (Рисунок 11).

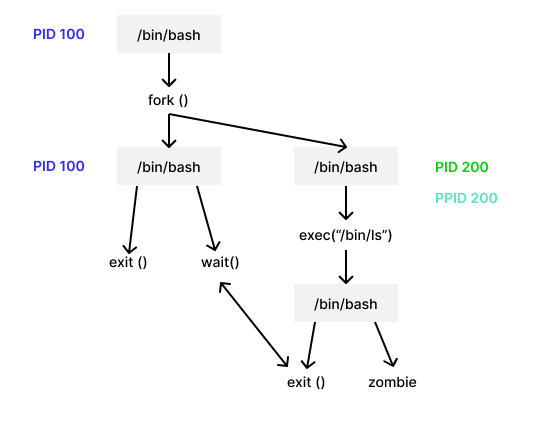


Рисунок 11 - Схема жизненного цикла процесса в ОС

Этапы состоят из следующих шагов:

1. процесс /bin/bash клонирует себя системным вызовом fork();
2. при этом создается клон /bin/bash с новым PID (идентификатор процесса) и PPID, равный PID родителя;
3. клон выполняет системный вызов exec с указанием на исполняемый файл и заменяет свой код кодом исполняемого файла (родительский процесс при этом ждет завершения потомка wait();
4. если по каким-то причинам потомок завершил свою работу, а родительский процесс не смог получить об этом сигнал, то данный процесс (потомок) не освобождает занятые структуры ядра и состояние процесса становиться zombie.

Первый процесс в Системе запускается при инициализации ядра. Данный процесс называется initи имеет PID=1 – это прародитель всех процессов в Системе.

Каждый запущенный процесс в любой момент времени может находиться в одном из следующих состояний (которое называют еще статусом процесса):

* **Активен (R=Running)** – находится в очереди на выполнение, то есть либо выполняется в данный момент, либо ожидает выделения ему очередного кванта времени центрального процессора.
* **"Спит" (S=Sleeping)** – находится в состоянии прерываемого ожидания, то есть ожидает какого-то события, сигнала или освобождения нужного ресурса.
* **Непрерываемое ожидание (D=Direct)** – ожидает определенного ("прямого") сигнала от аппаратной части и не реагирует на другие сигналы;
* **Приостановлен (T)** – находится в режиме трассировки (обычно такое состояние возникает при отладке программ);
* **"Зомби" (Z=Zombie)** – выполнение завершилось, но относящиеся к нему структуры ядра по каким-то причинам не освобождены. Одной из причин их появления в Системе может быть следующая ситуация: обычно освобождение структур ядра, относящихся к процессу, выполняет процесс-родитель после получения от потомка сигнала о завершении. Но бывают случаи, когда родительский процесс завершается раньше дочернего.
* "**Сироты**"– не имеют родителя, автоматически "усыновляются" процессом init, который и принимает сигналы об их завершении. Если процесс-родитель или init по каким-то причинам не может принять сигнал о завершении дочернего процесса, то процесс-потомок превращается в "зомби" и получает статус "Z". Процессы-"зомби" не занимают процессорного времени (т. е. их выполнение прекращается), но соответствующие им структуры ядра не освобождаются. В некотором смысле это "мертвые" процессы. Уничтожение таких процессов одна из обязанностей системного администратора. Появление данных процессов говорит о том, что в Системе что-то не в порядке, и зачастую проблема кроется в аппаратной части ПК. В этом случае необходимо проводить тестирования MHDD (например, с memtest).

Также, говоря о процессах в ОС, можно выделить особый вид процессов демоны. Данный вид процессов работает в фоне (подобно службам в Windows) без терминала и выполняет задачи для других процессов. Данный вид процессов на серверных системах является основным.

* 1. Виды межпроцессорного обмена

Все процессы в ОС обмениваются между собой какой-либо информацией. В ОС существует несколько видов межпроцессного обмена, а точнее сказатьсредств межпроцессного взаимодействия (Interprocess Communication, IPC), которые можно разбить на несколько уровней.

* + 1. Локальный уровень

Локальный вид межпроцессного обмена привязан к процессору и возможен только в пределах физического компьютера.

* + 1. Сигналы

Сигналы в ОС генерируются следующими процессами:

* с терминала, нажатием специальных клавиш или их комбинаций (например, нажатие Ctrl+C генерирует SIGINT, а Ctrl-Z SIGTSTP);
* ядром системы:
* при возникновении аппаратных исключений (недопустимых инструкций, нарушениях при обращении в память, системных сбоях и т. п.);
* при ошибочных системных вызовах;
* для информирования о событиях ввода-вывода;
* одним процессом другому (или самому себе) с помощью системного вызова kill(), в том числе из shell утилитой /bin/kill.

Примечание – Сигнал – это асинхронное уведомление процесса о каком-либо событии. Когда сигнал послан процессу, ОС прерывает выполнение процесса. Если процесс установил собственный обработчик сигнала, ОС запускает этот обработчик, передав ему информацию о сигнале. Если процесс не установил обработчик, то выполняется обработчик по умолчанию. Все сигналы начинаются на "SIG…" и имеют числовые соответствия, определяемые в заголовочном файле signal.h. Утилита kill позволяет задавать сигнал как числом, так и символьным обозначением.

* + - 1. Разделяемая память

Разделяемую память применяют для того, чтобы увеличить скорость прохождения данных между процессами. В обычной ситуации обмен информацией между процессами проходит через ядро. Техника разделяемой памяти позволяет осуществить обмен информацией не через ядро, а используя некоторую часть виртуального адресного пространства, куда помещаются и откуда считываются данные.

После создания разделяемого сегмента памяти любой из пользовательских процессов может подсоединить его к своему собственному виртуальному пространству и работать с ним, как с обычным сегментом памяти.

* + - 1. Очереди сообщений

В общих чертах обмен сообщениями выглядит примерно так: один процесс помещает сообщение в очередь посредством неких системных вызовов, а любой другой процесс может прочитать его оттуда, при условии, что и процесс-источник сообщения, и процесс-приемник сообщения используют один и тот же ключ для получения доступа к очереди.

* + 1. Удаленные
       1. Вызовы процедур

**RPC (Remote Procedure Calls)** – разновидность технологий, которая позволяет компьютерным программам вызывать функции или процедуры в другом адресном пространстве (как правило, на удалённых компьютерах). Обычно реализация RPC технологии включает в себя два компонента: сетевой протокол (чаще TCP и UDP, реже HTTP) для обмена в режиме клиент-сервер и язык сериализации объектов (или структур для необъектных RPC).

* + - 1. Межпроцессорное взаимодействие

**DBus (Desktop Bus)** – это система межпроцессного взаимодействия (IPC), предназначенная для обмена данными и вызова методов между процессами в рабочем столе ОС. Вот несколько ключевых аспектов, описывающих DBus и его применение:

* **Межпроцессное взаимодействие**:
* **Клиент-Серверная модель** – обеспечивает клиент-серверную модель взаимодействия. Система включает шину сообщений, к которой подключены как клиентские, так и серверные процессы.
* **Отправка сообщений** – процессы могут отправлять сообщения друг другу через шину DBus. Сообщения содержат данные, сигналы и запросы на вызов методов.
* **Шина системы и шина сеанса**:
* **Шина системы** – для взаимодействия между системными службами и процессами. Она используется для обмена данными на уровне всей системы.
* **Шина сеанса** – для взаимодействия между процессами, принадлежащими конкретному пользовательскому сеансу.
* **Уведомления и службы**:
* **Системные службы** – для предоставления функциональности и обмена информацией, например NetworkManager, systemd и другие.
* **Уведомления** – для отправки уведомлений и обновлений между приложениями и компонентами рабочего стола.
* **Широкое использование в среде рабочего стола**:
* **Интеграция приложений** – облегчает интеграцию различных приложений и компонентов рабочего стола, например для управления звуковыми настройками, обмена файлами, получения уведомлений и т.д.
* **Контроль за сеансом** – позволяет управлять параметрами сеанса, такими как настройки энергосбережения, языковые предпочтения и т.д.
* **Стандартизация и расширяемость**:
* **Стандарты сообщений** – существуют стандарты сообщений для определенных функций, таких как стандарты Freedesktop.org, что обеспечивает согласованный подход к взаимодействию между приложениями и службами.
* **Расширяемость** – поддерживает расширение функциональности через добавление новых интерфейсов и объектов, что делает его гибким и расширяемым.
  + - 1. Сокеты

**Сокеты** представляют собой виртуальный объект, который существует, пока на него ссылается хотя бы один из процессов. Сокеты бывают двух типов: локальные и сетевые.

При использовании **локального сокета** ему присваивается UNIX-адрес и просто будет создан специальный файл (файл сокета) по заданному пути, через который смогут сообщаться любые локальные процессы путём простого чтения/записи из него. При использовании **сетевого сокета** создается абстрактный объект, привязанный к слушающему порту ОС и сетевому интерфейсу, которому присваивается INET-адрес.

* 1. Управление процессами
     1. Получение информации о процессе

Перед тем как управлять процессами, нужно научиться получать о процессах необходимую информацию. В ОС существует псевдофайловая система procfs, которая в большинстве дистрибутивов монтируется в общую ФС в каталог /proc. У данной ФС нет физического места размещения, нет блочного устройства, такого как жесткий диск. Вся информация, хранимая в данном каталоге, находится в оперативной памяти компьютера, контролируется ядром ОС и не предназначена для хранения файлов пользователя.

Для получения информации о каком-либо процессе можно использовать существующие утилиты ps и top для просмотра информации о процессах.

Чтобы получить список всех процессов, достаточно ввести команду:

# ps aux

Следует обратить внимание, что некоторые процессы указаны в квадратных скобках "[ ]" – это процессы, которые входят непосредственно в состав ядра и выполняют важные системные задачи, например, такие как управление буферным кешем [pdflush] и организацией свопинга [kswapd], которые требуют очень ответственное отношение. Остальная часть процессов относится к пользовательским.

Информация, получаемая по каждому процессу:

* **PID, PPID** – идентификатор процесса и его родителя;
* **%CPU** – доля процессорного времени, выделенная процессу;
* **%MEM** – процент используемой оперативной памяти;
* **VSZ** – виртуальный размер процесса;
* **TTY** – управляющий терминал;
* **STAT** – статус процесса:
* R – выполняется;
* S – "спит";
* Z – "зомби";
* < – повышенный приоритет;
* + – в интерактивном режиме;
* **START** – время запуска;
* **TIME** – время исполнения на процессоре.

Команда ps делает моментальный снимок процессов в текущий момент. В отличии от нее команда topдинамически выводит состояние процессов и их активность в реальном режиме времени.

Пример вывода команды top:

14:32:49 up 35 days, 6:01, 4 users, load average: 0.65, 0.51, 0.49

Tasks: 432 total, 1 running, 431 sleeping, 0 stopped, 0 zombie

CPU0 : 1.6%us, 3.6%sy, 0.0%ni, 85.3%id, 9.2%wa, 0.0%hi, 0.3%si, 0.0%st

CPU1 : 0.9%us, 1.9%sy, 0.0%ni, 96.9%id, 0.0%wa, 0.0%hi, 0.3%si, 0.0%st

Mem: 1033596K total, 1016644K used, 16952K free, 82928K buffers

Swap: 2096376K total, 12632K used, 2083744K free, 478220K cached

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND

2788 root 15 -5 0 0 0 S 2.0 0.0 404:43.97 md3\_raid1

7961 root 20 0 5964 2528 1684 R 2.0 0.2 0:00.14 top

6629 root 20 0 8612 2884 2096 S 0.7 0.3 0:00.96 dovecot-auth

57 root 15 -5 0 0 0 S 0.3 0.0 4:36.10 kblockd/0

8703 ulogd 20 0 17700 4216 656 S 0.3 0.4 87:23.98 ulogd

11336 ldap 20 0 394M 15M 8292 S 0.3 1.5 5:29.28 slapd

25757 ldap 20 0 394M 15M 8292 S 0.3 1.5 5:11.71 slapd

10991 root 20 0 2188 1004 588 S 0.3 0.1 4:23.33 dovecot

1 root 20 0 1712 516 464 S 0.0 0.0 0:46.17 init

2 root 15 -5 0 0 0 S 0.0 0.0 0:00.00 kthreadd

3 root RT -5 0 0 0 S 0.0 0.0 0:05.92 migration/0

.....

2960 root 16 -4 1980 520 392 S 0.0 0.1 0:10.04 udevd

2993 dovecot 20 0 4412 1800 1476 S 0.0 0.2 0:00.00 pop3-login

2994 dovecot 20 0 4412 1800 1476 S 0.0 0.2 0:00.02 pop3-login

В верхней части вывода отображается астрономическое время, время, прошедшее с момента запуска Системы, число пользователей в Системе, число запущенных процессов и число процессов, находящихся в разных состояниях, данные об использовании ЦПУ, памяти и свопа. Далее следует таблица, характеризующая отдельные процессы. Число строк, отображаемых в этой таблице, определяется размером окна: сколько строк помещается, столько и выводится.

Содержимое окна обновляется каждые 5 секунд. Список процессов может быть отсортирован по используемому времени ЦПУ (по умолчанию), по использованию памяти, по PID, по времени исполнения. Переключать режимы сортировки при отображении можно с помощью следующих клавиатурных команд:

* Shift+N – по PID;
* Shift+A – по возрасту;
* Shift+P – по использованию ЦПУ;
* Shift+M – по использованию памяти;
* Shift+T – по времени выполнения.

С помощью команды <K> можно завершить некоторый процесс (его PID будет запрошен), а с помощью команды <R> можно переопределить значение niceдля некоторого процесса.

Полезную информацию о процессах также позволяет получить программа lsof, которая выдает список всех файлов, используемых сейчас процессами, включая каталоги, используемые каким-либо процессом в качестве текущего или корневого; разделяемые библиотеки, загруженные в память и т. д.

* + 1. Приоритезация

Каждому процессу при запуске устанавливается определенный приоритет, который имеет значение от -20 до +20, где +20 самый низкий*.* Приоритет нового процесса равен приоритету процесса-родителя. Для изменения приоритета запускаемой программы существуетутилитаnice. Пример ее использования:

nice [- adnice] command [args]

где adnice – значение (от –20 до +19), добавляемое к значению nice процесса-родителя. Отрицательные значения может устанавливать только суперпользователь. Если опция adnice не задана, то по умолчанию для процесса-потомка устанавливается значение nice, увеличенное на 10 по сравнению со значением nice родительского процесса.

Командаrenice служит для изменения значения nice для уже выполняющихся процессов. Суперпользователь может изменить приоритет любого процесса в Системе. Другие пользователи могут изменять значение приоритета только для тех процессов, для которых данный пользователь является владельцем. При этом обычный пользователь может только уменьшить значение приоритета. Поэтому процессы с низким приоритетом не могут породить "высокоприоритетных детей".

Одним из средств управления процессами являются сигналы. Некоторые сигналы можно сгенерировать с помощью определенных комбинаций клавиш, но такие комбинации существуют не для всех сигналов. Зато имеется командаkill, которая позволяет послать заданному процессу (указав его PID) любой сигнал:

kill [-SIG] PID [PID..]

где SIG – это номер сигнала или наименование сигнала, причем если указание сигнала опущено, то посылается сигнал 15 (SIGTERM программное завершение процесса). Часто используется сигнал *9 (KILL)*, с помощью которого суперпользователь может завершить любой процесс. Но сигнал этот очень "грубый", потому что он просто "убивает" процесс, не давая ему времени на корректное сохранение всех обработанных данных. Поэтому в большинстве случаев рекомендуется использовать сигналы TERMили QUIT, которые завершают процесс более "мягко". Если процессу необходимо как-то по-особенному реагировать на сигнал, он может зарегистрировать обработчик, а если обработчика нет, за него отреагирует Система.

Два сигнала 9(KILL) и 19(stop) всегда обрабатывает Система. Первый из них нужен для того, чтобы убить процесс наверняка (отсюда и название). Сигнал stop приостанавливает процесс: в таком состоянии процесс не удаляется из таблицы процессов, но и не выполняется до тех пор, пока не получит сигнал *18 ( CONT)* после чего продолжит работу. В ОС сигнал stopможно передать активному процессу с помощью управляющего символа "^Z".

Обычные пользователи могут посылать сигналы только тем процессам, для которых они являются владельцами. Если в командеkill воспользоваться идентификатором процесса (PID), равным -1, то указанный в команде сигнал будет послан всем принадлежащим данному пользователю процессам. Суперпользователь root может посылать сигналы любым процессам. Когда суперпользователь посылает сигнал идентификатору -1, он рассылается всем процессам, за исключением системных. Если этим сигналом будет SIGKILL, то у простых пользователей будут потеряны все открытые ими, но не сохраненные файлы данных.

При обычном запуске процесс работает на переднем плане, то есть процесс "привязывается" к терминалу, с которого он запущен, воспринимая ввод с этого терминала и осуществляя на него вывод. Но можно запустить процесс в фоновом режиме, когда он не связан с терминалом, для чего в конце командной строки запуска программы добавляют символ "&".

Есть также две встроенные команды, которые служат для перевода процессов на передний план или возврата их в фоновый режим. Команда fg переводит указанный в аргументе процесс на передний план, а команда bg переводит процесс в фоновый режим. Одной командой bg можно перевести в фоновый режим сразу несколько процессов, а вот возвращать их на передний план необходимо по одному. Аргументами команд fg и bg могут являться только номера заданий, запущенных из текущего экземпляра shell. Возможные значения заданий можно увидеть, выполнив команду jobs.

При завершении сессии оболочка посылает всем порожденным ею процессам сигнал "отбой", по которому порожденные ею процессы могут завершиться, что не всегда желательно. Если требуется запустить в фоновом режиме программу, которая должна выполняться и после выхода из оболочки, то ее нужно запускать с помощью утилиты nohup:

nohup команда &

Запущенный таким образом процесс будет игнорировать посылаемые ему сигналы (не игнорируются только сигналы SIGHUP и SIGQUIT).

* + 1. Команда free

В ОС можно использовать команду freeчтобы получить подробный отчет об использовании памяти Системой.

Команда freeпредоставляет информацию об общем объеме физической памяти и памяти подкачки, а также о свободной и используемой памяти.

Синтаксис команды следующий:

free [OPTIONS]

При использовании без какой-либо опции команда freeбудет отображать информацию о памяти и подкачке в килобайтах.

Обозначения столбцов при выводе команды:

* **total** – общий объем памяти, который может быть использован приложениями.
* **used** – используемая память; рассчитывается как: used = total-free-buffers- cache.
* **free** – свободная/неиспользуемая память.
* **Shared** – можно игнорировать, поскольку не имеет значения (только для обратной совместимости);
* **buff / cache** – объединенная память, используемая буферами ядра, а также страничным кешем и блоками. Эта память может быть освобождена в любое время, если это необходимо приложениям.
* **available** – оценка объема памяти, доступного для запуска новых приложений без подкачки.

Если требуется, чтобы буферы и кеш отображались в двух отдельных столбцах, используют параметр-w.

По умолчанию команда freeпоказывает информацию о памяти в килобайтах. Чтобы просмотреть информацию в удобочитаемом формате (обычно в мегабайтах и гигабайтах), используют параметр -h**:**

free -h

Команда free также позволяет указать единицы измерения памяти. Допустимые варианты:

* b, --bytes – отображать вывод в байтах;
* -kilo – отображать вывод в килобайтах (1 Кб = 1000 байт);
* -mega – отображать вывод в мегабайтах;
* -giga – показать вывод в гигабайтах;
* -tera – показать вывод в терабайтах;
* k, --kibi – показать вывод в кибибайтах. (1 КiB = 1024 байт) (по умолчанию);
* m, --mebi – показать вывод в мебибайтах (1 мебибайт = 2^20 байт);
* g, --gibi – показать вывод в гибибайтах (1 гибибайт=2^30 байт);
* -tebi – отображать вывод в тебибайтах (1 тебибайт = 2^40 байт);
* -peti – показать вывод в пебибайтах (1 пебибайт = 2^50 байт);
* -si – вместо 1024 используют --mebi --si 1000. Например, --mebi --si равно --mega.

Например, чтобы отобразить результат в мегабайтах, следует выполнить:

free --mega

Чтобы постоянно отображать информацию о памяти на экране, нужно вызвать freeс параметром -s(--seconds), за которым следует число, определяющее задержку.

Например, чтобы распечатывать информацию о памяти каждые пять секунд, необходимо выполнить (Рисунок 12):

free -s 5

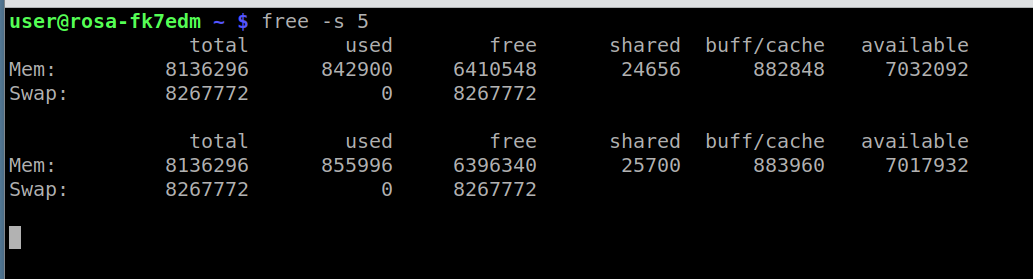


Рисунок 12 - Вывод команды free -s 5

Команда free будет продолжать отображать результат, пока не буде нажата комбинация клавиш Ctrl+C, что похоже на поведение команды watch.

Чтобы отображать результат определенное количество раз, используют параметр -c(--count). Например, эта команда выведет результат десять раз (Рисунок 13):

free -s 5 -c 3

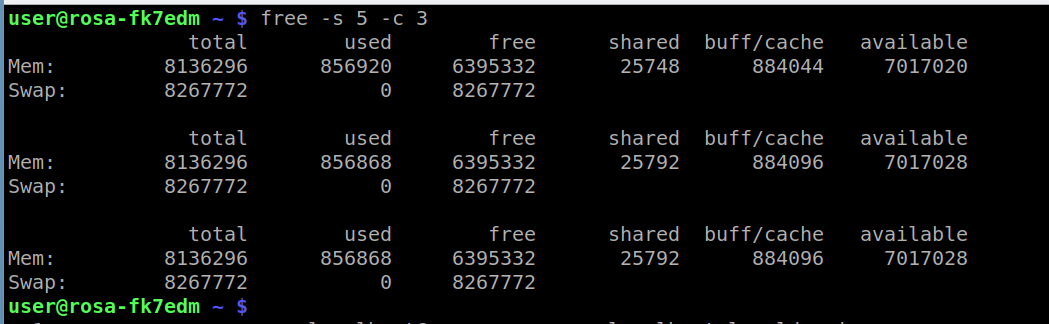


Рисунок 13 - Вывод команды free -s 5 -c 3

Команду в реальной работе следует автоматизировать и запускать непрерывно или периодически

Еще одна отличная особенность команды free – возможность автоматизировать команду. Есть два варианта, которые помогают настроить этот инструмент:

* параметр -s, который запускает команду free в течение заданного интервала секунд, пока пользователь не закроет программу (Ctrl+C);
* параметр -c, который можно использовать отдельно или в сочетании с параметром секунд. Если ввести только -c и целое число (n), команда будет запускаться n раз. По умолчанию используются интервалы в одну секунду.

Например, если требуется открыть несколько приложений и посмотреть количество используемой памяти. Для тестового вывода используется читаемый человеком формат: степени 1000 (Gb) вместо 1024 (GiB). Запись ведется в течение 20 секунд, чтобы проанализировать воздействие. Данные собираются каждые 5 секунд за 4 цикла. Пример, отформатированный для командной строки и ее вывода (Рисунок 14):

free -h --si -s 5 -c 4

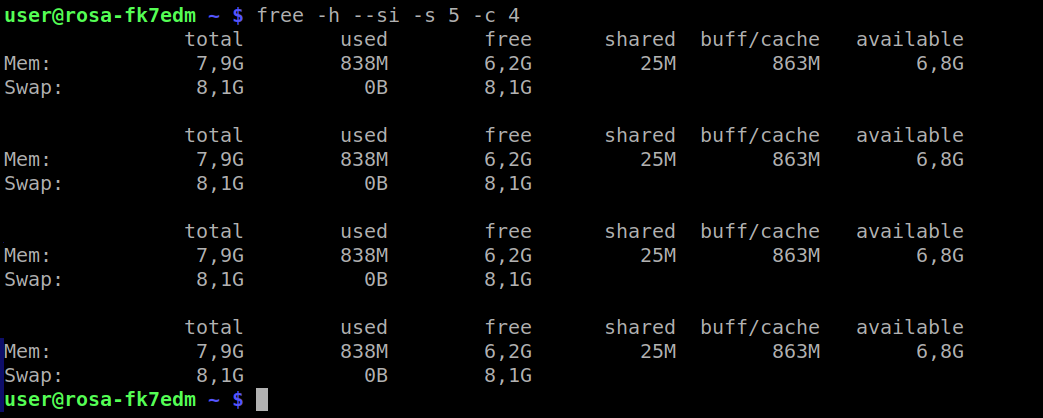


Рисунок 14 - Вывод примера команды free

В результате будет показан только один вывод для команды free, но этот вывод будет изменяться через регулярные промежутки времени (Рисунок 15).

watch free -h

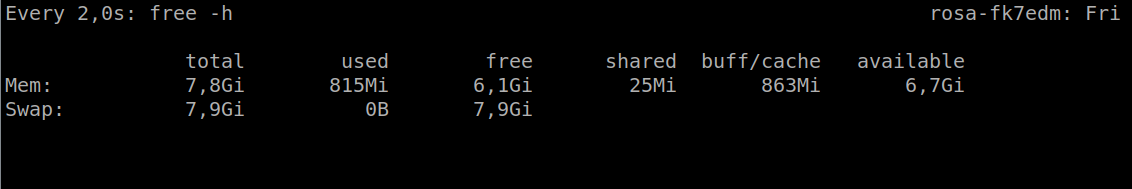


Рисунок 15 - Вывод команды watch free -h

* + 1. Команда htop

Команда htop – это интерактивный монитор Системы в терминале, который предоставляет детальную информацию о загрузке процессора, использовании памяти, активности сети и других ресурсах. Вот несколько ключевых функций и использований htop:

* Мониторинг процессов отображает список текущих процессов в Системе, а также информацию о CPU, памяти и других ресурсах, используемых каждым процессом; процессы отображаются в виде древовидной структуры, позволяя легко увидеть зависимости между процессами;
* Интерактивный интерфейс позволяет пользователю в реальном времени взаимодействовать с данными; можно использовать клавиши со стрелками и функциональные клавиши для сортировки, фильтрации и управления процессами;
* Мониторинг ресурсов показывает использование процессора, памяти (включая использование swap), сети и дисков; графики в верхней части экрана предоставляют визуальное представление общей загрузки CPU и использования памяти;
* Управление процессами позволяет завершать, приостанавливать, возобновлять и изменять приоритеты процессов напрямую из интерфейса htop; пользователь может отправлять сигналы процессам, например для перезапуска или завершения процесса;
* Поддержка цветов использует цвета для облегчения восприятия информации, что делает отображение данных более интуитивным;
* Поиск и фильтрация пользователь может использовать встроенные инструменты для поиска и фильтрации процессов по различным критериям.

Команду htopможно запускать как с правами обычного пользователя (Рисунок 16):

htop

так и с привилегиями суперпользователя:

sudo htop

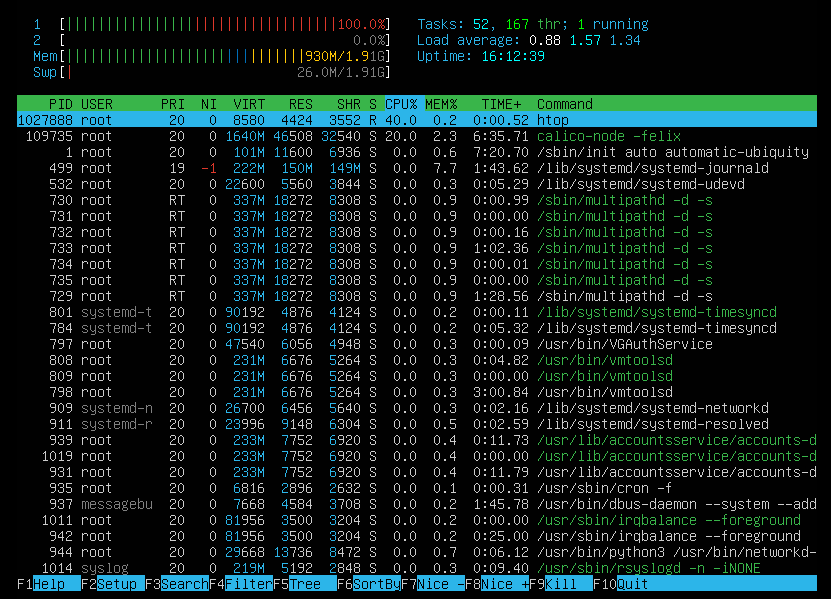


Рисунок 16 - Вывод команды htop

Чтобы программа выводила новые данные раз в 5 секунд:

sudo htop -d 50

Для вывода процессов в виде дерева используют опцию -t (Рисунок 17):

sudo htop -t

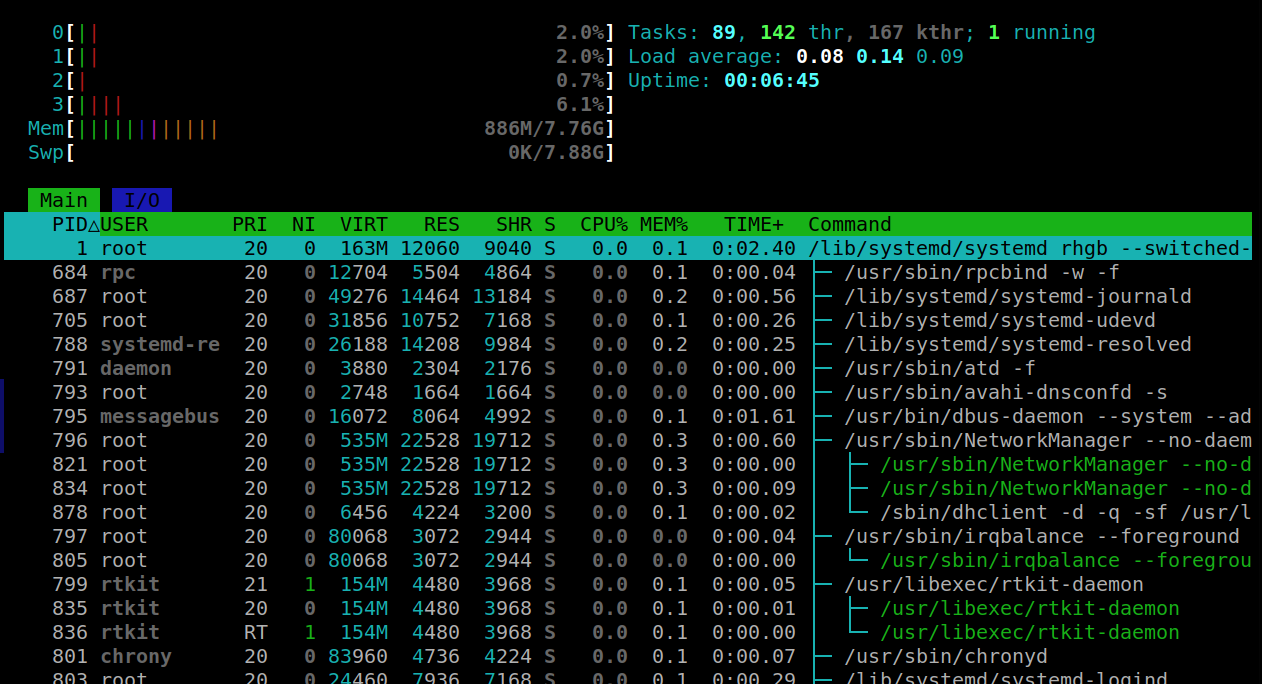


Рисунок 17 - Вывод команды htop -t

* + 1. Команда df

Чтобы посмотреть доступное пространство на всех примонтированных разделах и информацию о них достаточно набрать команду (Рисунок 18):

df

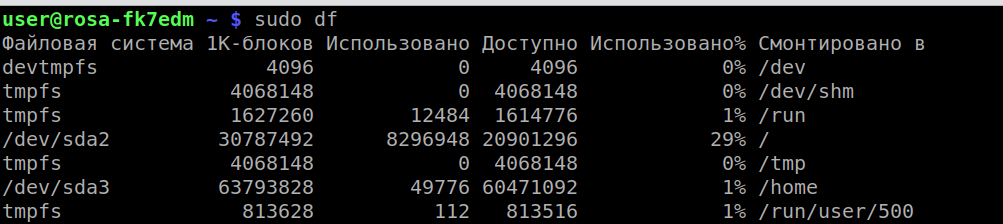


Рисунок 18 - Вывод команды df

Утилита отображает стандартный набор колонок, но понять в ее выводе что-то с первого раза сложно. Все данные выводятся в килобайтах. Теперь можно попросить утилиту выводить данные в более читаемом формате (Рисунок 19):

df -h

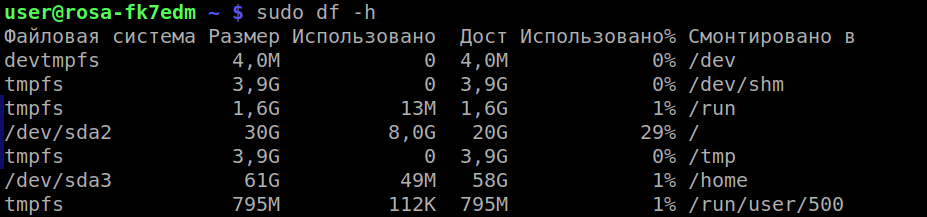


Рисунок 19 - Вывод команды df -h

Это самое популярное сочетание опций, потому что оно дает максимально понятную информацию и уже можно судить о том, что происходит на диске. Например, можно увидеть в корневой директории уже занято 83% места и, возможно, пора что-то удалить. Если задать опцию -a, можно получить информацию обо всех файловых системах известных ядру, которые были смонтированы (Рисунок 20):

df -a

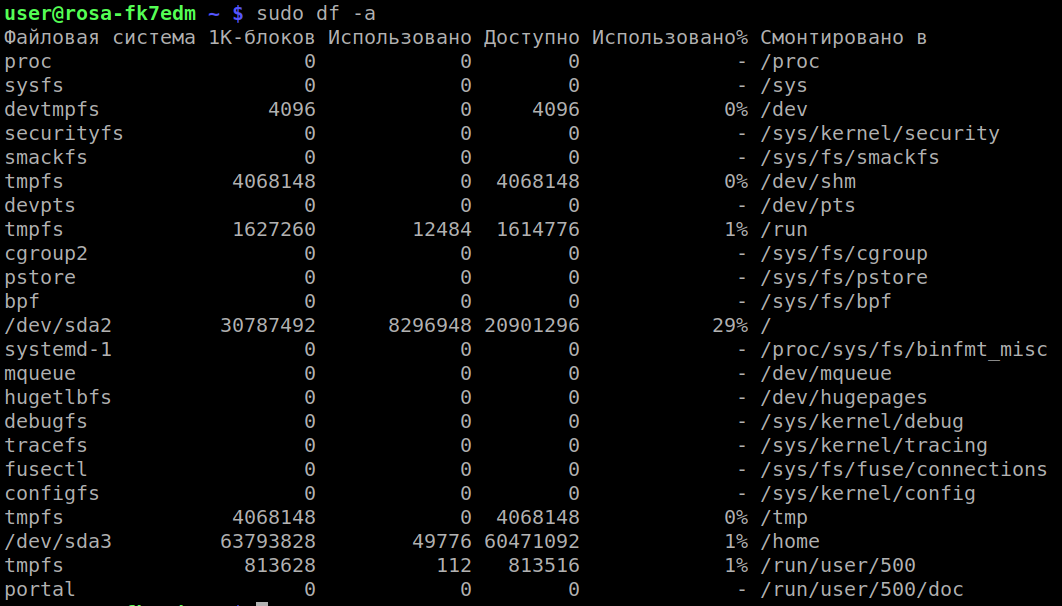


Рисунок 20 - Вывод команды df -a

Здесь отображается огромное количество виртуальных ФС ядра. Если требуется вывести только информацию про реальные ФС на жестком диске, можно использовать опцию -x, чтобы отфильтровать все tmpfs:

df -x tmpfs

Также можно отфильтровать две ФС:

df -x devtmpfs -x tmpfs

или же указать ФС, которую нужно отображать:

df -t ext4

Также можно указать интересующий раздел (Рисунок 21):

df -h /dev/sda2

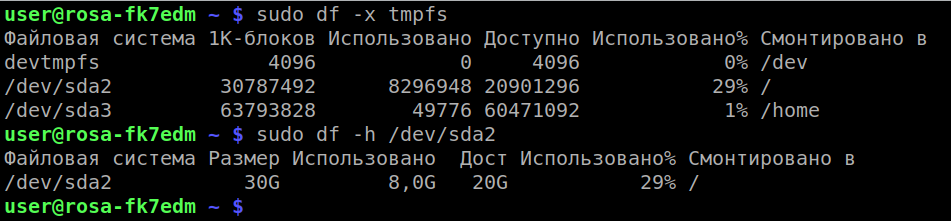


Рисунок 21 - Вывод команды df -h /dev/sda2

С помощью опции -iможно посмотреть информацию про состояние inodeв ФС (Рисунок 22):

df -hi /dev/sda2

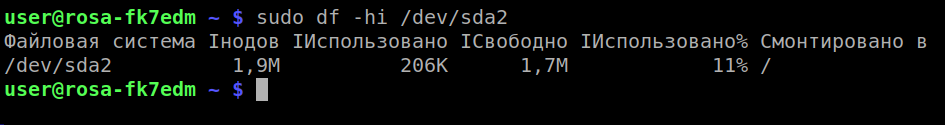


Рисунок 22 - Вывод команды df -hi /dev/sda2

Вывести объем доступного пространства в каталоге:

df -h /home

Команда df– очень простой в освоении, но эффективный инструмент, позволяющий всегда быть в курсе использования дискового пространства в Системах.

* + 1. Команда du

Команда du сообщает приблизительный объем дискового пространства, используемого данными файлами или каталогами. Это практически полезно для поиска файлов и каталогов, занимающих большие объемы дискового пространства.

Общий вид команды:

du [OPTIONS]... FILE...

Если данный FILE является каталогом, duбудет суммировать использование диска для каждого файла и подкаталога в этом каталоге. Если FILE не указан, du сообщит об использовании диска в текущем рабочем каталоге.

При выполнении без какой-либо опции du отображает использование диска для данного файла или каталога и каждого из его подкаталогов в байтах.

Также можно передать команде du несколько файлов и каталогов в качестве аргументов:

du ~/Documents ~/Pictures

Если запустить du для файла или каталога, для которого нет прав, в выводе получится примерно следующее "du: невозможно прочитать каталог". В этой ситуации нужно будет добавить к команде sudo.

Часто используемые параметры для командыdu:

* -a – сообщает об использовании дискового пространства каждым файлом в каталоге:

du -a /tmp/

* -h – отображает информацию о пространстве, занимаемом только данным каталогом, в удобочитаемом формате, например, чтобы получить общий размер /var/lib и всех его подкаталогов, нужно выполнить следующую команду:

sudo du -h /var/lib

Следует использовать sudo, потому что большинство файлов и каталогов внутри каталога /var/lib принадлежат пользователю rootи не доступны для чтения обычным пользователям.

* -s – отображает только общий размер указанного каталога, а не подкаталогов (Рисунок 23):

sudo du -sh /var

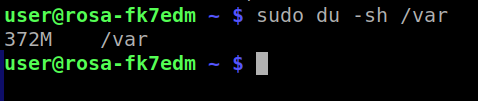


Рисунок 23 - Вывод команды du -sh /var

* -c – указывает duсообщать общую сумму, если требуется получить объединенный размер двух или более каталогов (Рисунок 24):

sudo du -csh /var/log /var/lib

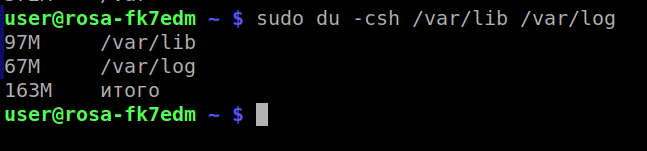


Рисунок 24 - Вывод команды du -csh /var/lib /var/log

* --max-depth – отображает использование диска подкаталогами n-уровня с указанием уровня подкаталогов, например, чтобы получить отчет о каталогах первого уровня, используют (Рисунок 25):

sudo du -h --max-depth=1 /var/log



Рисунок 25 - Вывод команды du -h --max-depth=1 /var/log

* --apparent-size – по умолчанию утилита du повторно использует дисковое пространство, используемое каталогом или файлом; чтобы узнать очевидный размер файла (фактический объем данных в файле), используют переключатель (Рисунок 26):

sudo du -sh --apparent-size /var/lib

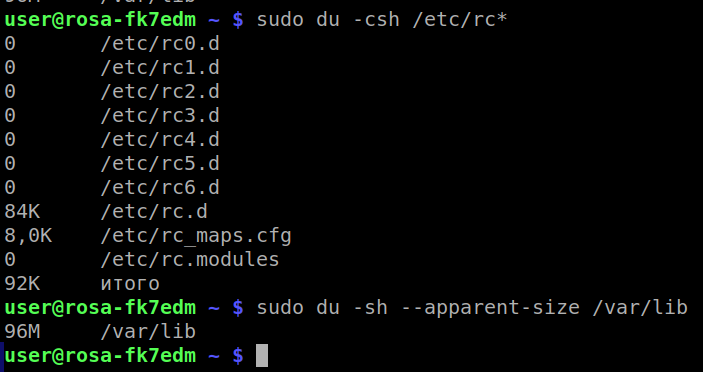


Рисунок 26 - Вывод команды sudo du -sh --apparent-size /var/lib

duтакже позволяет использовать шаблон оболочки. Например, чтобы получить размер всех каталогов, начинающихся с "rc" в каталоге /etc, нужно выполнить:

sudo du -csh /etc/rc\*

* + 1. Комбинации команд

Команду duможно комбинировать с другими командами с конвейерами.

Например, чтобы напечатать 5 самых больших каталогов внутри каталога */var* нужно передать вывод du в команду sort**,** чтобы отсортировать каталоги по их размеру, а затем передать результат в команду head**,** которая распечатает только 5 верхних каталогов:

sudo du -h /var/ | sort -rh | head -5

В отличие от команды df**,** которая выводит информацию об использовании диска смонтированными ФС, команда duдает оценку дискового пространства, используемого данными файлами или каталогами.

1. Каналы
   1. Общие сведения

Канал – это важный механизм в ОС, который позволяет передавать данные от одного процесса к другому без сохранения чего-либо на диске.

В ОС есть два типа каналов: **безымянные каналы** и **именованные каналы (FIFO).**

Каналы используются путем объединения команд, разделенных символом канала "|". Его часто называют конвейером и оболочка определяет его поведение.

Конвейер выполняет каждую команду в отдельном процессе, работающем в фоновом режиме, начиная с крайней левой команды.

Затем стандартный вывод команды с левой стороны подключается к стандартному вводу команды с правой стороны это обеспечивает однонаправленность потока.

Этот механизм действует до тех пор, пока не будут завершены все процессы в конвейере.

Оболочка использует маркер "|&" для ссылки на конвейер, соединяя как стандартный вывод, так и стандартную ошибку команды на левой стороне со стандартным вводом команды на правой стороне.

Например, если требуется использовать командуnetstat, чтобы увидеть, какие процессы запущены с помощью localhost, и отфильтровать с помощью утилитыgrep:

netstat -tlpn | grep 127.0.0.1

(Not all processes could be identified, non-owned process info will not be shown, you would have to be root to see it all.)

tcp 0 0 127.0.0.1:3306 0.0.0.0:\* LISTEN -

В этом примере вывода можно наблюдать как stdout, так и stderr скрипта в netstat независимо от фильтра.

Для объединения stderrв стандартный вывод и передачи его в stdin grep:

netstat -tlpn |& grep 127.0.0.1

tcp 0 0 127.0.0.1:3306 0.0.0.0:\* LISTEN -

* 1. Конвейеры

**Pipe** (конвейер) – это однонаправленный (программный) канал межпроцессного взаимодействия. Конвейеры чаще всего используются в shell-скриптах для связи нескольких команд путем перенаправления вывода одной команды (stdout) на вход (stdin) последующей, используя символ конвейера "|":

cmd1 | cmd2 |.... | cmdN

Например:

$ grep -i “error”./log | wc -l 43

В этом примере grep выполняет регистронезависимый поиск строки "error" в файле log, но результат поиска не выводится на экран, а перенаправляется на вход (stdin) команды wc, которая в свою очередь выполняет подсчет количества строк.

* 1. Логика

Конвейер обеспечивает асинхронное выполнение команд с использованием буферизации ввода/вывода. Таким образом все команды в конвейере работают параллельно, каждая в своем процессе.

Несмотря на то, что конвейер оперирует файловыми дескрипторами потоков ввода/вывода, все операции выполняются в памяти, без нагрузки на диск.

В Bash есть переменная с именем PIPESTATUS, которая содержит список состояний завершения процессов в самом последнем выполненном конвейере:

$ exit 1 | exit 2 | exit 3 | exit 4 | exit 5

$ echo ${PIPESTATUS[@]}

1 2 3 4 5

Возвращаемый статус выполнения всего конвейера будет зависеть от статуса переменной pipefail.

Если эта переменная установлена, возвращаемым статусом конвейера будет статус завершения самой правой команды с ненулевым статусом или будет равен нулю, если все команды завершатся успешно:

$ set -o pipefail

$ exit 1 | exit 2 | exit 3| exit 4 | exit 0

$ echo $?

4

При отключенной опции pipefailвозвращаемым статусом канала будет статус завершения последней команды:

$ set +o pipefail

$ exit 1 | exit 2 | exit 3| exit 4 | exit 0

$ echo $?

0

В Bash также есть опция lastpipe, которая инструктирует оболочку выполнить последнюю команду на переднем плане текущей среды.

* 1. Именованные каналы

**FIFO** (First In, First Out) – способ организации и манипулирования данными относительно времени и приоритетов. Этот способ описывает принцип технической обработки очереди или обслуживания конфликтных требований путём упорядочения процесса по принципу "первым пришёл первым обслужен" (ПППО).

FIFO, также известный как именованный канал, представляет собой специальный файл, похожий на канал, но с именем в ФС. Несколько процессов могут обращаться к этому специальному файлу для чтения и записи, как к любому обычному файлу.

Таким образом, имя работает только как ориентир для процессов, которым необходимо использовать имя в ФС.

FIFO обладает теми же характеристиками, что и любой другой файл. Например, у него есть права, разрешения и метаданные.

Важной особенностью FIFO является то, что он обеспечивает двунаправленную связь.

В ОС можем создать FIFO с помощью команд mknod (используя букву "p" для обозначения типа FIFO) и mkfifo*:*

$ mkfifo pipe1

$ mknod pipe2 p

$ ls -l

prw-r--r-- 1 cuau 0 Oct 7 21:17 pipe1

prw-r--r-- 1 cuau 0 Oct 7 21:17 pipe2

Здесь можно видеть, что тип файла FIFO обозначен буквой "p".

Этот механизм позволяет создавать более сложные приложения, используя оболочку.

Именованные и анонимные каналы можно использовать вместе. В качестве примера можно создать обратную оболочку, объединяющую как FIFO, так и каналы.

Удобно использовать утилиту nc для создания клиент-серверного приложения, в котором "серверная” сторона предоставит свою оболочку, а “клиентская” сторона сможет получить к ней доступ.

Сначала необходимо установитьпакетnc:

$ sudo dnf -y nc

Далее нужно создать FIFO с именем fifo\_reverse*,* набрав:

mkfifo fifo\_reverse

Затем следует войти в Систему с двумя разными пользователями, каждый из которых будет действовать как “клиент“ (например, "пользователь1”) и как “сервер” (например, “пользователь2”). Для запуска этого конвейера в оболочке user2 выполняется:

user2\_prompt$ bash -i < fifo\_reverse |& nc -l 127.0.0.1 1234 > fifo\_reverse

В этом однострочной команде оболочка считывает содержимое FIFO и передает его интерактивной оболочке Bash.

Затем оба стандартных вывода и stderr интерактивной оболочки будут переданы команде nc, которая будет прослушивать порт 1234 с адресом 127.0.0.1.

И, наконец, когда “клиент” успешно установит соединение, nc запишет полученное в FIFO, и интерактивная оболочка сможет выполнить полученное.

Теперь, используя оболочку user1, выполняется:

user1\_prompt$ nc 127.0.0.1 1234

user2\_prompt$

В результате получено приглашение user2, но используя оболочку user1, объединяющую анонимные и именованные каналы.

* + 1. Временные именованные каналы

В некоторых оболочках есть функция подстановки процессов, которая соединяет ввод или вывод списка команд с FIFO. Затем команды будут использовать имя этого FIFO-файла.

Обозначение для этого механизма в Bash и Zsh <(список команд) – для передачи результата списка в стандартный ввод фактической команды или >(список команд) – для передачи стандартного вывода фактической команды в стандартный ввод списка.

Для передачи выходных данных нескольких команд в команду wc:

$ wc -l \

<(find / -mindepth 1 -maxdepth 1 -type d) \

<(find /opt -mindepth 1 -maxdepth 1 -type d)

20 /proc/self/fd/11

2 /proc/self/fd/12

22 total

В этом примере вывода используется команда find, чтобы получить количество каталогов внутри каталогов / и /opt.

* + 1. Условия использования каналов

Использование анонимного канала вместо именованного зависит от характеристик, которые нужно найти. Некоторые из них могут быть постоянством, двусторонней связью, наличием имени файла, созданием фильтра и ограничением прав доступа среди других.

Например, если требуется отфильтровать выходные данные команды несколько раз, использование анонимного канала кажется наиболее подходящим вариантом. Следует также помнить, что используемая оболочка будет играть центральную роль при работе с анонимными каналами.

С другой стороны, если требуется имя файла и нет необходимости хранить данные на диске, то ищется FIFO, если нужно только имя в качестве ссылки с содержимым, которое поступает непосредственно из другого процесса.

* 1. Программные каналы

Программный канал (pipe) создается с помощью символа "|". При этом вывод первой команды используется как ввод для второй команды. На практике программные каналы часто объединяются командой less, если нужно просматривать длинный ввод в виде постраничной разбивки:

ls -l | less

Приведенная команда позволяет получить "оглавление" текущего каталога и записать его в программный канал. Оттуда команда less, выполняемая параллельно, считывает ввод предыдущей команды и отображает его на экране.

Программные каналы великолепно подходят для комбинирования различных команд. Так, например, следующая команда возвращает отсортированный список всех установленных пакетов RPM:

rpm -qa | sort

Вместо программных каналов для переадресации ввода и вывода могут использоваться так называемые FIFO-файлы. Работать с FIFO-файлами при вводе значительно сложнее, чем с программными каналами, однако они позволяют уяснить, на что же именно влияет символ "|"**.** На практике такие файлы применяются для того, чтобы две программы, работающие независимо друг от друга, могли обмениваться информацией.

mkfifo fifo

ls -l > fifo &

less < fifo

С помощью трех вышеуказанных команд сначала создается FIFO-файл. Кроме того, в виде фонового процесса запускается ls. Она записывает свой вывод в файл. Оттуда less вновь считывает данные и отображает их на экране.

Только такие команды подходят для формулирования программного канала, из которого команды, предназначенные для ввода, будут считывать информацию. Если ситуацией иная, можно достичь сходных эффектов с помощью подстановки команд или команды xargs.

1. Отладка

Довольно часто приходится сталкиваться с тем, что программное обеспечение не работает должным образом. В разделе приведены наиболее простые инструменты и способы поиска ошибок в программах.

* 1. Консольный отладчик GDB

GDB – полноценный консольный отладчик для ОС позволяет останавливать выполнение в произвольных точках, отслеживать состояние переменных, стека, памяти и регистров и т. д. Возможна удалённая отладка.

Запуск:

gdb executable

gdb

После запуска можно указать файл для отладки:

file executable

Либо подключиться к удалённому сеансу отладки:

target remote host:port

Также можно отлаживать уже запущенную программу (нужны права суперпользователя):

attach <pid>

Запустить отлаживаемую программу на выполнение:

run arguments

Программа остановится, если выполнение дойдёт до точки останова, либо, если программе придёт сигнал. Точки останова можно установить командой:

break main.c:15

или

break 15

если отлаживается, например, программа из одного файла. Команды step и next выполняют следующую строку после остановки. Первая входит в функцию, если следующая строка содержит её вызов, вторая игнорирует вызов функции и останавливается уже после её выполнения. Команда continue продолжает выполнение до следующей точки останова, либо до следующего сигнала, либо до завершения программы. Команда quit осуществляет выход из GDB. С помощью команды:

print выражение

можно выполнить любое выражение языка C, содержащее переменные, видимые в текущей точке программы. Команда list выдаёт листинг части программы в окрестности текущей строки.

GDB весьма удобен при отладке, если есть доступ к исходникам и отладочной информации. Для более подробной информации можно обратиться к документации GDB:

info gdb

В частых случаях будет более удобно использование средств, предоставляемых ОС, таких как strace.

* 1. Утилита strace

strace – утилита, позволяющая отслеживать системные вызовы, используемые программой, и получаемые ей сигналы.

Запуск:

strace program

или

strace -p pid

для отладки уже запущенной программы (нужны права суперпользователя).

Можно отлаживать многопроцессные программы, передав ключ -f. В этом случае будут печататься системные вызовы и сигналы не только родителя, но и детей, порождённых после присоединения к процессу. Можно указать файл, в который strace будет писать вывод:

strace -o log.txt program

Для системных вызовов strace печатает имя вызова, переданные ему аргументы и значение, которое он вернул. Можно ограничить количество отслеживаемых системных вызовов:

strace -e trace=open,close,read,write,fork,exec program

То же для сигналов:

strace -e signal=!SIGTERM program

Можно отслеживать данные, читаемые или записываемые в файловые дескрипторы:

strace -e read=0,3 program

strace -e write=1,2,4 program

Кроме того, можно вмешиваться в работу процесса, например, подменяя результаты системных вызовов. Об этом и многом другом можно прочитать в документации strace:

man 1 strace

* 1. Псевдо-файловая система

Различную отладочную информацию можно также почерпнуть из procfs. В ОС эта файловая система монтируется в /proc. Сведения о процессе с id pid можно найти в каталоге /proc/pid. Из такой информации можно выделить:

Файл /proc/pid/exe является символической ссылкой на бинарный файл работающего процесса.

Файл /proc/pid/status, содержащий различные данные о состоянии процесса, такие как:

* имя исполняемого файла;
* umask процесса;
* собственно состояние процесса;
* uid и gid процесса;
* количество потоков процесса.
* маски заблокированных, принимаемых и игнорируемых сигналов; маски представлены в виде 8 байт, среди которых бит с номером "i" отвечает сигналу с номером "i"; биты отсчитываются с единицы, справа налево.

Файл /proc/pid/syscall содержит номера работающих в данный момент системных вызовов, вызванных из программы, и их аргументы.

Файл /proc/pid/maps содержит список виртуальных страниц в адресном пространстве программы, файл /proc/pid/pagemap, их отображение в физические страницы.

Файл /proc/pid/stack содержит трейс стека ядра.

Каталог /proc/pid/fd содержит ссылки на файлы, соответствующие открытым в процессе файловым дескриптором.

Каталог /proc/pid/fdinfo содержит файлы с информацией об этих дескрипторах.

Файл /proc/pid/limits содержит таблицу лимитов для процесса.

Более полное представление о структуре procfs можно получить в документации:

man 5 proc

1. Файловая система

Файловая система в ОС, как и в ОС Windows, представляет собой иерархическую структуру каталогов и файлов (в виде дерева), но при этом имеет ряд кардинальных отличий.

* 1. Каталоги

В ОС файловая система представлена единым корневым каталогом, обозначаемым как **слеш (**/**)**. Соответственно, при данной файловой структуре не диски содержат каталоги, а каталог диски.

* 1. Типы файлов

Все объекты в ОС являются файлами. Существуют следующие типы файлов:

* Обычные файлы **"-";**
* Символьные и двоичные данные (текст, картинки, программы и др.);
* Каталог (директория) "d"**;**
* Список ссылок на файлы или другие каталоги;
* Символьные ссылки "l"**;**
* Ссылки на другие файлы по имени;
* Блочные устройства "b"**,** символьные устройства "c";
* Интерфейсы для взаимодействия с аппаратным обеспечением (диски, терминалы, клавиатуры, принтеры и др.); при обращении к файлу устройства ядро ОС передает запрос драйверу этого устройства;
* Сокеты "s" и каналы "p";
* Интерфейсы для взаимодействия процессов.

Файлы программы разделяются по каталогам в зависимости от типа. Например, исполняемые файлы в /bin, библиотеки в /lib, файлы конфигураций в /etc, логи и кеш в /var.

Если подключается съемный носитель или диск, файл устройства будет виден в каталоге /dev (devices). Чтобы увидеть содержимое этого устройства, его нужно смонтировать в отдельную директорию /mnt. Также ФС позволяет примонтировать его и в любое другое место, например /home.

* 1. Регистр имен файлов

Также стоит отметить чувствительность файловой системы ОС к регистру. Файлы Temp.txt и temp.txt будут интерпретироваться как разные файлы и могут находиться в одной директории, в отличие от ОС Windows, которая не различает регистр имен. Такое же правило действует и на каталоги имена в разных регистрах указывают на разные каталоги.

Назначение каждой директории регламентирует "Стандарт иерархии файловой системы" FHS (Filesystem Hierarchy Standard), основные директории которой приведены на схеме ниже (Рисунок 27):

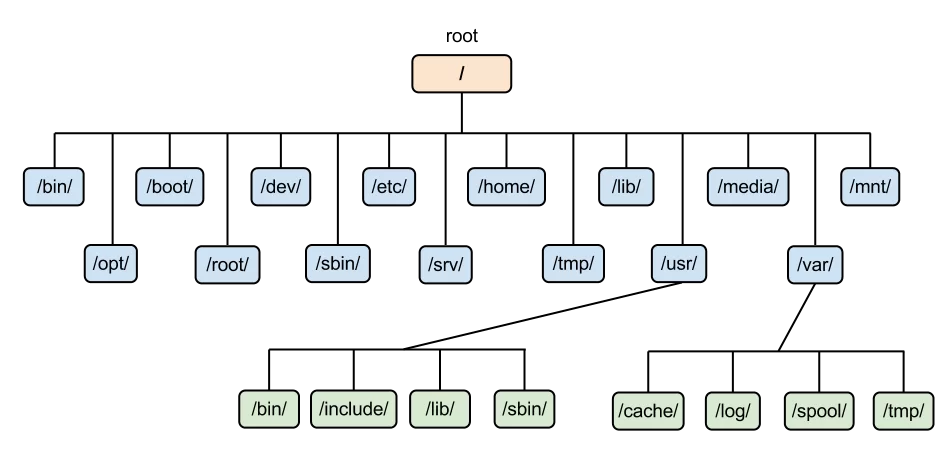


Рисунок 27 - Стандарт FNS

В директориях содержатся следующие данные согласно стандарту FNS:

* **/** – root-каталог; вся иерархия Системы;
* **/bin** – двоичные исполняемые файлы; основные общие команды, хранящиеся отдельно от других программ (например, pwd, ls, cat, ps);
* **/boot** – файлы, используемые для загрузки ОС (образ initrd, ядро vmlinuz);
* **/dev** – файлы устройств (драйверов); с помощью этих файлов можно взаимодействовать с устройствами, например, если это жесткий диск, можно подключить его к ФС, а в файл принтера можно написать напрямую и отправить задание на печать;
* **/etc** – файлы конфигураций программ; позволяют настраивать системы, сервисы, скрипты системных демонов;
* **/home** – каталог, аналогичный каталогу Users в Windows; содержит домашние каталоги учетных записей пользователей (кроме root); при создании нового пользователя здесь создается одноименный каталог с аналогичным именем и хранит личные файлы этого пользователя;
* **/lib** – системные библиотеки, с которыми работают программы и модули ядра;
* **/lost+found** – файлы, восстановленные после сбоя работы Системы; Система проведет проверку после сбоя и найденные файлы можно будет посмотреть в данном каталоге;
* **/media** – точка монтирования внешних носителей; например, при установке диска в дисковод, он будет автоматически смонтирован в директорию /media/cdrom;
* **/mnt** – точка временного монтирования; ФС подключаемых устройств обычно монтируются в этот каталог для временного использования;
* **/opt** – дополнительные (необязательные) приложения; такие программы обычно не подчиняются принятой иерархии и хранят свои файлы в одном подкаталоге (бинарные, библиотеки, конфигурации);
* **/proc** – файлы, хранящие информацию о запущенных процессах и о состоянии ядра ОС;
* **/root** – директория, которая содержит файлы и личные настройки суперпользователя;
* **/run** – файлы состояния приложений, например PID-файлы или UNIX-сокеты;
* **/sbin** – аналогично /bin содержит бинарные файлы; утилиты нужны для настройки и администрирования Системы суперпользователем;
* **/srv** – файлы сервисов, предоставляемых сервером, например FTP или Apache HTTP);
* **/sys** – данные непосредственно о Системе; информация о ядре, драйверах и устройствах;
* **/tmp** – временные файлы; данные файлы доступны всем пользователям на чтение и запись; очищается при перезагрузке;
* **/usr** – содержит пользовательские приложения и утилиты второго уровня, используемые пользователями, а не Системой; содержимое доступно только для чтения (кроме root); каталог имеет вторичную иерархию и похож на корневой;
* **/var** – переменные файлы; имеет подкаталоги, отвечающие за отдельные переменные; например, логи будут храниться в /var/log, кеш в /var/cache, очереди заданий в /var/spool/ и так далее.

1. Система инициализации
   1. Общие сведения

systemd– это система инициализации и системный диспетчер ОС. Из-за сложной адаптивности знакомство с системой systemdоправдано, поскольку это существенно упростит администрирование серверов. Изучение и использование инструментов и демонов, которые включают systemd, поможет лучше оценить предоставляемые возможности и гибкость или, по крайней мере, работать с минимальным количеством проблем.

* 1. Управление службами

Основополагающая цель системы инициализации заключается в инициализации компонентов, которые должны запускаться после загрузки ядра ОС (традиционно называются компоненты пользовательского пространства). Система инициализации также используется для управления службами и демонами для сервера и в любой момент времени работы системы. С учетом этого мы начнем с нескольких базовых операций по управлению службами.

В systemd целью большинства действий являются модули в качестве управляемых ресурсов. Модули распределяются по категориям по типу ресурса, который они представляют, и определяются файлами, известными как файлы модулей. Тип каждого модуля можно вывести из суффикса в конце файла.

Для задач по управлению службами целевым модулем будут модули службы, которые имеют файлы модулей с суффиксом .service**.** Однако для большинства команд по управлению службами можно не использовать суффикс .service, поскольку systemdопределяет необходимость работать со службой при использовании команд по управлению службами.

* + 1. Запуск/остановка

Чтобы запустить службу systemd, используя инструкции в файле модуля службы, используют команду start. Если работа осуществляется в качестве пользователя без прав root, то потребуется использовать sudo, поскольку это влияет на состояние JC:

sudo systemctl start application.service

Так как systemd сама будет искать файлы **\***.service для команд управления службами, то команду можно легко ввести следующим образом:

sudo systemctl start application

Примечание – Хотя можно использовать вышеуказанный формат для общего администрирования, для ясности в документе будет использоваться суффикс .service для остальных команд, чтобы предельно четко выражать цель действий.

Чтобы остановить работающую в данный момент службу, можно использовать команду stop:

sudo systemctl stop application.service

* + 1. Перезапуск

Чтобы перезапустить работающую службу, можно использовать командуrestart**:**

sudo systemctl restart application.service

Если данное приложение может перезагрузить файлы конфигурации (без перезапуска), можно выдать командуreload для инициализации этого процесса:

sudo systemctl reload application.service

Если нет уверенности в наличии у службы функции перезагрузки своей конфигурации, можно использовать командуreload-or-restart. Это перезагрузит необходимую конфигурацию только при наличии. В противном случае будет перезагружена служба для выбора новой конфигурации:

sudo systemctl reload-or-restart application.service

Также можно перечитать конфигурационные файлы systemd, используя следующую команду:

sudo systemctl daemon-reload

* + 1. Включение/выключение

Указанные выше команды полезны для запуска или остановки служб во время текущего сеанса. Чтобы дать команду systemdавтоматически запускать службы при загрузке, их необходимо включить.

Для запуска службы во время загрузки используют команду enable**:**

sudo systemctl enable application.service

При этом будет создана символическая ссылка из системной копии служебного файла (обычно в /lib/systemd/system или /etc/systemd/system) в месте на диске, где systemd ищет файлы для автозапуска (обычно/etc/systemd/system/some\_target.target.wants).

Для того чтобы добавить службу в автозагрузку без перезагрузки Системы, можно использовать флаг--now:

sudo systemctl enable --now application.service

Чтобы отключить автоматический запуск службы, можно ввести следующее:

sudo systemctl disable application.service

При этом будет удалена символическая ссылка, что укажет на то, что служба не должна запускаться автоматически.

Следует учитывать, что включение службы не запустит ее в текущем сеансе. Если требуется запустить службу и включить ее при загрузке, необходимо задать обе команды start и enable.

* + 1. Проверка статуса

Чтобы проверить статус службы в Системе, можно использовать команду status:

systemctl status application.service

При этом будет показан статус службы, иерархия контрольных групп и первые несколько строк журнала.

Также есть методы для проверки определенных статусов. Например, чтобы проверить, активен ли (работает ли) модуль в данный момент, можно использовать командуis-active**:**

systemctl is-active application.service

Это вернет текущий статус модуля, который обычно active или inactive**.** Код выхода будет "0", если он активен, и результат будет проще парсить в скрипты оболочки.

Чтобы увидеть, включен ли модуль, можно использовать команду is-enabled:

systemctl is-enabled application.service

Это выведет информацию о том, что службаenabled или disabled, и снова установит код выхода на "0" или "1" в зависимости от вопроса команды.

Третья проверка заключается в проверке того, находится ли модуль в состоянии сбоя. Это означает, что была проблема, которая запустила данный модуль:

systemctl is-failed application.service

Это вернет active, если он работает должным образом, или failed, если возникла ошибка. Если модуль был намеренно остановлен, может вернуться unknown или inactive. Статус выхода "0" означает, что произошел сбой, а статус выхода "1" указывает на какой-либо другой статус.

* 1. Обзор состояния Системы

Команды до сих пор были полезны для управления отдельными службами, но они не очень подходят для понимания текущего состояния Системы. Существует ряд команд systemctl, предоставляющих эту информацию.

* + 1. Текущие модули

Чтобы увидеть список всех активных модулей, о которых знает systemd, можно использовать команду list-units:

systemctl list-units

Состояние будет, как правило, enabled**,** disabled**,** static или masked. В этом контексте static обозначает, что файл модуля не содержит раздел install, который используется для включения модуля. Эти модули как таковые не могут быть включены. Обычно это означает, что модуль выполняет разовое действие или используется только как зависимость другого модуля и не должен работать самостоятельно.

* + 1. Управление модулями

До сих пор описывалась работа со службами и информация о модулях и файлах модулей, о которых знает systemd. Однако можно узнать более конкретную информацию о модулях, используя некоторые дополнительные команды.

* + - 1. Файлы

Чтобы отобразить файл модуля, который systemd загрузила в Систему, можно использовать команду cat. Например, чтобы увидеть файл модуля демона-планировщика atd, можно ввести следующее:

systemctl cat atd.service

Output[Unit]

Description=ATD daemon

[Service]

Type=forking

ExecStart=/usr/bin/atd

[Install]

WantedBy=multi-user.target

Вывод – это файл модуля, известный выполняемому в настоящее время процессу systemd. Это может быть важно, если ранее был модифицирован файл модуля или переопределены определенные опции во фрагменте файла модуля.

* + - 1. Зависимости

Чтобы увидеть дерево зависимостей модуля, можно использовать команду list-dependencies:

systemctl list-dependencies sshd.service

При этом отобразится иерархическая схема зависимостей, с которой необходимо работать, чтобы запустить интересуемый модуль. Зависимости в этом контексте включают те модули, которые либо требуются, либо желательны для модулей выше.

Outputsshd.service

├─system.slice

└─basic.target

├─microcode.service

├─rhel-autorelabel-mark.service

├─rhel-autorelabel.service

├─rhel-configure.service

├─rhel-dmesg.service

├─rhel-loadmodules.service

├─PATHs.target

├─slices.target

...

Рекурсивные зависимости отображаются только для модулей .target, которые указывают состояние Системы. Чтобы рекурсивно перечислить все зависимости, добавляют флаг --all.

Чтобы отобразить обратные зависимости (модули, зависящие от указанного модуля), можно добавить в команду флаг --reverse. Другие полезные флаги --before и --after могут быть использованы для отображения модулей, которые зависят от указанного модуля перед ними и после соответственно.

* + - 1. Свойства

Чтобы увидеть свойства более низкого уровня модуля, можно использовать команду show. При этом будет выведен список свойств, заданных для указанного модуля с помощью формата key=value:

systemctl show sshd.service

OutputId=sshd.service

Names=sshd.service

Requires=basic.target

Wants=system.slice

WantedBy=multi-user.target

Conflicts=shutdown.target

Before=shutdown.target multi-user.target

After=syslog.target network.target auditd.service systemd-journald.socket basic.target system.slice

Description=OpenSSH server daemon

...

Если требуется отобразить одно свойство, можно передать флаг -p с именем свойства. Например, чтобы увидеть конфликты, которые есть у модуля sshd.service, можно ввести следующее:

systemctl show sshd.service -p Conflicts

OutputConflicts=shutdown.target

* + - 1. Маскировка

systemd также имеет возможность отметить модуль как полностью незапускаемый автоматически или вручную, связав его с /dev/null. Это называется маскировкой модуля, и она возможна с помощью команды mask:

sudo systemctl mask nginx.service

Это не позволит запустить службу Nginx автоматически или вручную, пока она замаскирована.

Если проверить list-unit-files, то можно убедиться, что служба теперь указана как замаскированная:

systemctl list-unit-files

Output...

kmod-static-nodes.service static

ldconfig.service static

mandb.service static

messagebus.service static

nginx.service masked

quotaon.service static

rc-local.service static

rdisc.service disabled

rescue.service static

...

Если попытаться запустить службу, то будет выдано следующее сообщение:

sudo systemctl start nginx.service

OutputFailed to start nginx.service: Unit nginx.service is masked.

Чтобы снять маскировку модуля и сделать его доступным для использования снова, используют команду unmask:

sudo systemctl unmask nginx.service

Это вернет модуль в его предыдущее состояние, что позволит его запускать или включать.

* + - 1. Редактирование

Хотя конкретный формат файлов модулей выходит за рамки этого руководства, systemctl предоставляет встроенные механизмы для редактирования и модификации файлов модулей при необходимости изменений.

Команда edit по умолчанию откроет фрагмент файла модуля для интересующего модуля:

sudo systemctl edit nginx.service

Это будет пустой файл, который можно использовать для переопределения или добавления директив в определение модуля. Каталог будет создан в каталоге /etc/systemd/system, который содержит название модуля с добавлением .d. Например, для nginx.service будет создан каталог под названием nginx.service.d.

В этом каталоге будет создан фрагмент под названием override.conf. При загрузке модуля systemd в памяти соединит фрагмент переопределения с полным файлом модуля. Директивы фрагмента получат приоритет над найденными в оригинальном файле модуля.

Если требуется редактировать весь файл модуля, а не создавать фрагмент, можно передать флаг --full:

sudo systemctl edit --full nginx.service

Это загрузит текущий файл модуля в редактор, в котором его можно редактировать. После выхода из редактора измененный файл будет записан в /etc/systemd/system, что будет иметь приоритет над определением модуля Системы (обычно находится в /lib/systemd/system).

Чтобы удалить какие-либо сделанные добавления, удаляют либо каталог конфигурации модуля .d или модифицированный служебный файл из /etc/systemd/system. Например, для удаления фрагмента можно ввести следующее:

sudo rm -r /etc/systemd/system/nginx.service.d

Чтобы удалить весь отредактированный файл модуля, добавляют:

sudo rm /etc/systemd/system/nginx.service

После удаления файла или каталога необходимо перезагрузить процесс systemd, чтобы он больше не пытался ссылаться на эти файлы и не возвращался к использованию системных копий. Для этого можно ввести следующую команду:

sudo systemctl daemon-reload

* + 1. Цели
       1. Общие сведения

Целями являются специальные файлы модулей, которые описывают состояние Системы или точку синхронизации. Как и другие модули, файлы, которые определяют цели, могут быть идентифицированы по суффиксу, которым в данном случае является .target. Цели используются для группировки других модулей.

Их можно использовать, чтобы привести Систему в определенные состояния, подобно тому, как другие системы инициализации используют уровни запуска. Они используются в качестве справки, если доступны определенные функции, позволяя указывать желаемое состояние вместо необходимости использования отдельных модулей для получения этого состояния.

Например, swap.target используется для указания того, что переключение готово к использованию. Модули, являющиеся частью этого процесса, могут синхронизироваться с этой целью путем указания в своей конфигурации, что они WantedBy= или RequiredBy= swap.target. Модули, которым требуется возможность переключения, могут указывать это состояние с помощью спецификаций Wants=, Requires= и After= для указания характера их отношений.

* + - 1. Настройка цели по умолчанию

Процесс systemd имеет цель по умолчанию, которую он использует при загрузке Системы. Выполнение каскада зависимостей от этой одной цели приведет Систему в желаемое состояние. Чтобы найти цель по умолчанию для Системы, нужно ввести:

systemctl get-default

Outputmulti-user.target

Если требуется задать другую цель по умолчанию, можно использовать set-default. Например, если установлен графический рабочий стол и требуется загрузить Систему в него по умолчанию, можно изменить цель по умолчанию соответственно:

sudo systemctl set-default graphical.target

* + - 1. Список доступных целей

В ОС можно получить список имеющихся целей, выполнив:

systemctl list-unit-files --type=target

В отличие от уровней запуска, несколько целей могут быть активны одновременно. Активная цель указывает, что система systemd попыталась запустить все модули, привязанные к цели, и не попыталась закрыть их снова. Чтобы увидеть все активные цели, нужно ввести:

systemctl list-units --type=target

* + - 1. Изолирование целей

Можно запустить все модули, связанные с целью, и остановить все модули, не являющиеся частью дерева зависимостей. Команда, необходимая для этого, называется isolate. Она аналогична изменению уровня запуска в других системах инициализации.

Например, при работе в графической среде с активным graphical.target, можно закрыть графический интерфейс и перевести Систему в состояние многопользовательской командной строки путем изоляции multi-user.target. Поскольку graphical.target зависит от multi-user.target, а не наоборот, все графические модули будут остановлены.

Если потребуется посмотреть на зависимости цели, которую изолируют, перед выполнением этой процедуры следует убедиться, что не остановлены важные службы:

systemctl list-dependencies multi-user.target

Если модули полностью устраивают и будут сохранены в активном состоянии, можно изолировать цель, выполнив:

sudo systemctl isolate multi-user.target

systemctl работает главным образом с основным процессом systemd. В экосистеме systemd есть другие компоненты, которые контролируются другими утилитами. Другие возможности, такие как управление журналами и сеансы пользователя, обрабатываются отдельными демонами и утилитами управления (journald/journalctl и logind/loginctl соответственно). Использование этих инструментов и демонов упростит задачу управления.

1. Указатели
   1. Общие сведения

Ссылка на файл в ОС – это указатель на файл. Если проводить аналогию с Windows, то ссылки чем-то похожи на ярлыки. То есть при создании ссылки, которая указывает на какой-либо файл или директорию, можно разместить эту ссылку в другом каталоге. Используя такую ссылку, обращение будет направляться к настоящему файлу или каталогу.

Ссылки в ОС бывают двух типов: **символические** и **жесткие**. Несмотря на то, что оба типа называются ссылками, они имеют существенные отличия друг от друга. Поэтому очень важно понимать, как создавать и использовать тот или иной тип ссылок.

* 1. Символические ссылки

**Символическая ссылка** (**symbolic link**) – это специальный файл, который является ссылкой на другой файл или каталог (их еще называют целевым файлом, целевым каталогом).

Символические ссылки также называют символьными, мягкими ссылками (**soft links**) или сим-ссылками (**sym-link**).

Важно понимать, что символическая ссылка не содержит в себе копию самого файла, на которую она указывает. Она является всего лишь указателем на файл. Несмотря на это, символическая ссылка обладает собственными правами доступа, так как сама является небольшим файлом, который содержит путь до целевого файла.

Возвращаясь к аналогии с ярлыками в Windows, символические ссылки – это своего рода ярлыки на файлы. Можно создавать несколько символических ссылок на один файл и эти ссылки могут иметь разные имена.

Связь между символической ссылкой и файлом, на который она указывает, является "мягкой". Если удалить символическую ссылку, то файл, на который она указывает, не удаляется.

Если удалить файл, на который указывает ссылка, то сама ссылка не обновляется и остается на диске. При этом она указывает на уже несуществующий файл. Аналогично, если переименовать или переместить целевой файл, то ссылка не обновляется автоматически.

При создании символических ссылок можно указывать относительный путь до целевого файла. В таком случае ссылка считает, что относительный путь указан относительно каталога, в котором создана сама ссылка (но не относительно каталога, из которого она была создана).

Схематично отношение между файлом, символической ссылкой и данными, которые хранятся в файле, можно показать следующим образом (Рисунок 28):

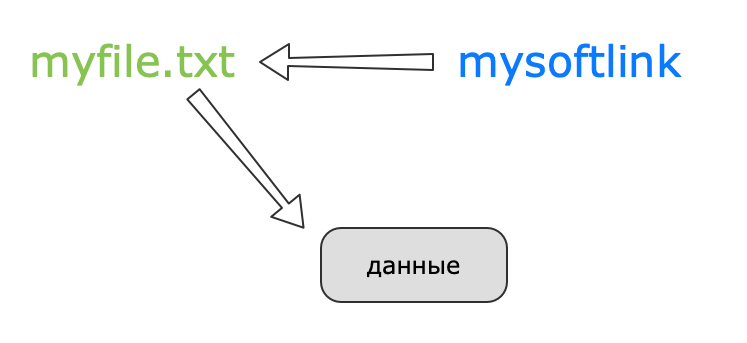


Рисунок 28 - Отношения между файлом и символической ссылкой

* 1. Жесткие ссылки

**Жесткая ссылка** (**hard link**) является своего рода синонимом для существующего файла. При создании жесткой ссылки создается дополнительный указатель на существующий файл, но не копия файла.

Жесткие ссылки выглядят в файловой структуре как еще один файл. Если жесткая ссылка создается в том же каталоге, где находится целевой файл, то они должны иметь разные имена. Жесткая ссылка на файл должна находится в той же ФС, что и другие жесткие ссылки на этот файл.

В ОС каждый файл имеет уникальный идентификатор индексный дескриптор (inode) – это число, которое однозначно идентифицирует файл в ФС. Жесткая ссылка и файл, для которой она создавалась, имеют одинаковые inode. Поэтому жесткая ссылка имеет те же права доступа, владельца и время последней модификации, что и целевой файл. Различаются только имена файлов.

Жесткие ссылки нельзя создавать для директорий.

Жесткая ссылка не может указывать на несуществующий файл.

Жесткие ссылки появились раньше, чем символические, но сейчас уже устаревают. В повседневной работе жесткие ссылки используются редко.

Схематично отношение между исходным файлом, жесткой ссылкой и данными можно показать следующей схемой (Рисунок 29):

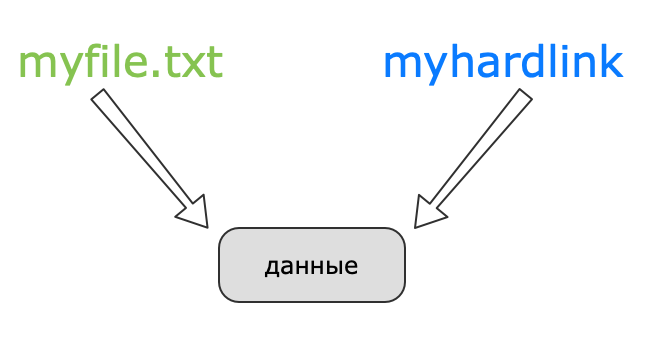


Рисунок 29 - Отношения между файлом и жесткой ссылкой

* 1. Отличия символической и жесткой ссылок

**Символическая ссылка:**

* Указывает на целевой файл или каталог. Фактически является небольшим файлом, содержащим путь до целевого файла.
* Не содержит внутри себя содержимого самого файла. Содержит путь к целевому файлу.
* Имеет собственные права доступа, которые не распространяются на целевой файл.
* Удаление/переименование/перемещение целевого файла не обновляет автоматически ссылку. Ссылка начинает указывать на несуществующий файл, становится неработающей.
* Изменение прав доступа у целевого файла не обновляет права доступа у ссылки.
* Может быть создана для директории.
* Ссылка и целевой файл имеют разные файловые индексы (inode) в файловой системе.
* Может указывать на несуществующий файл.
* Символическая ссылка может использовать относительный путь до целевого файла.

**Жесткая ссылка:**

* Является своего рода еще одним именем на файл.
* Не может указывать на директорию.
* Нельзя создавать жесткие ссылки между файлами разных ФС.
* Не может указывать на несуществующий файл.
* Жесткая ссылка и файл, для которого она создавалась, имеют одинаковые индексы (inode) в ФС.
  1. Создание ссылок

Для создания ссылок в ОС используется команда ln.

Синтаксис команды ln:

ln опции целевой\_файл имя\_ссылки

Обычно используется только одна опция -s. Полный список опций можно получить, выполнив команду:

man ln

Чтобы создать символическую ссылку, нужно выполнить команду ln с опцией -s**:**

ln -s целевой\_файл имя\_символьной\_ссылки

Чтобы создать жесткую ссылку нужно использовать команду ln без опции -s**.**

ln целевой\_файл имя\_жесткой\_ссылки

* 1. Удаление ссылок

Ссылки, как и обычные файлы, можно удалять, используя команду rm:

rm имя\_ссылки

* 1. Файлы-устройства

В файловой системе ОС осуществляется управление не только файлами и каталогами, но и устройствами так называются особым образом обозначаемые файлы, в которых не хранится никакой информации, так как данные файлы обеспечивают связь с ядром ОС.

* 1. Идентификаторы

Устройства обеспечивают доступ ко многим компонентам аппаратного обеспечения ПК, то есть к жестким дискам, дисководам, серийным и параллельным интерфейсам, оперативной памяти (RAM) и т. д. Три параметра старший номер устройства, младший номер устройства и тип доступа (блочный или символьный) определяют характеристику устройства.

Старший номер задает, какой драйвер ядра ОС отвечает за управление устройством.

У многих драйверов младший номер служит тем признаком, с помощью которого различаются отдельные (родственные) устройства, например драйверы различных сегментов одного и того же жесткого диска.

Тип доступа указывает, являются ли устройства буферизованными (все блочные устройства, например жесткие диски) или нет (символьно-ориентированные устройства, например серийные или параллельные интерфейсы).

При просмотре содержимого каталога /dev с помощью команды:

ls -l

вместо размера файла Система показывает номера устройства (старший и младший). Первый символ бита доступа b или c означает блочную либо символьную ориентацию.

ls -l /dev/sda?

brw-rw---- 1 root 8, 1 2010-02-02 10:39 /dev/sda1

brw-rw---- 1 root 8, 2 2010-02-02 10:39 /dev/sda2

...

* 1. Дескрипторы

В ОС в каталоге /dev находятся только так называемые индексные дескрипторы (I-Node) – это мельчайшие управляющие единицы Системы, а не настоящие файлы. Новые файлы устройств можно создавать с помощью команды mknod. Однако на практике это требуется редко, так как система udev делает такое автоматически.

Старший и младший номера устройств представлены в виде 64-битных чисел.

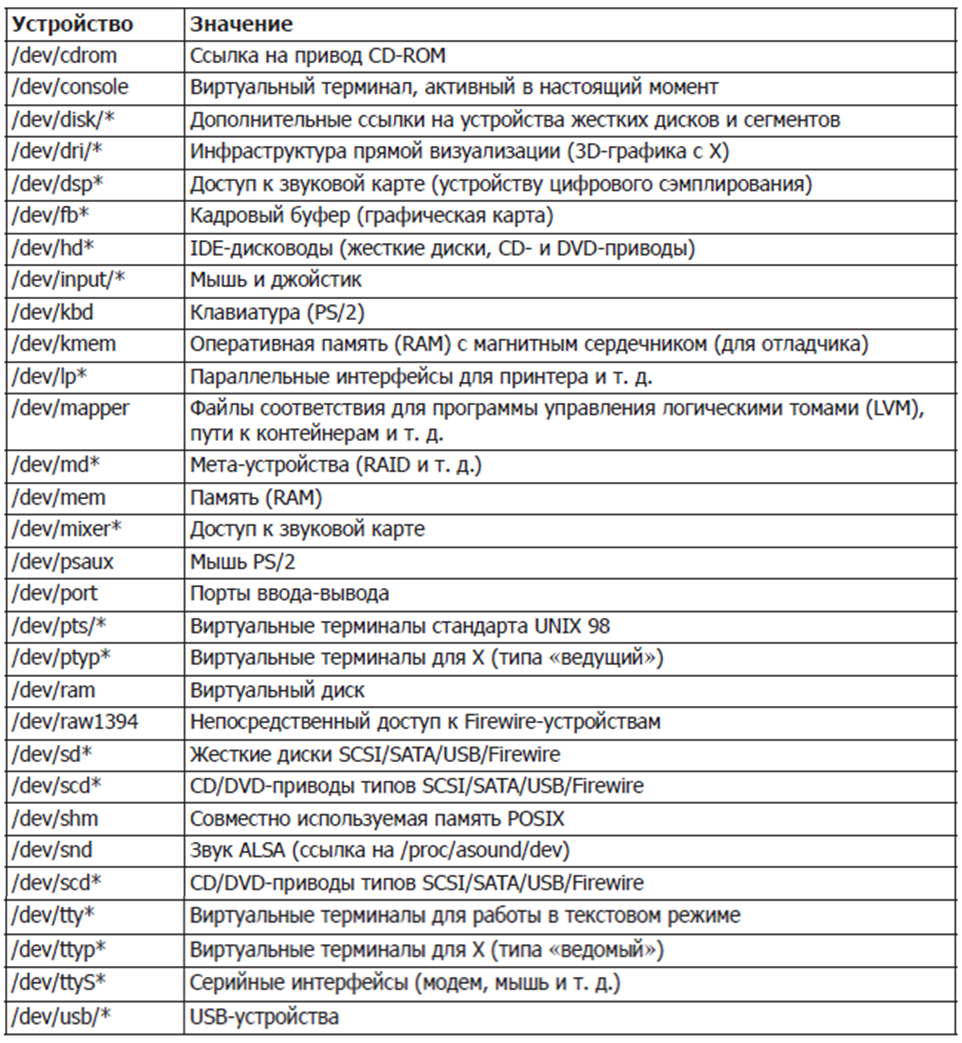
Доступ ко многим устройствам (из соображений безопасности) может получить только администратор либо члены определенной группы. Чтобы предоставить другому пользователю доступ к этим устройствам, добавляют его к этой группе.

Некоторые файлы устройств выполняют специфическую функцию:

* /dev/null – служит "черной дырой" и отправляемые туда данные навсегда исчезают (используется, например, для переадресации командного вывода, если эта информация должна быть скрыта);
* /dev/zero – это неиссякаемый источник нулевых байтов, которые иногда используются для того, чтобы заполнять нулями файлы до определенного (заданного) размера;
* /dev/random и /dev/urandom– возвращают случайные числа

Список важных файлов и устройств приведен в ниже (Таблица 23).

Таблица 23 - Перечень файлов и устройств



* 1. Система udev

Из списка в работе используется всего около сотни файлов, но этот набор неодинаков в зависимости от аппаратного обеспечения ПК, на котором установлена ОС.

В данном случае правильный выбор помогает сделать система udev. Фоновая программа udevdраспознает все аппаратные компоненты, подключенные к ПК, и создает необходимые файлы устройств. Программа udevdзапускается в начале процесса Init-V. Конфигурация осуществляется с помощью файлов, содержащихся в каталоге /etc/udev.

Система udevработает просто отлично и способна обращаться с внешними жесткими дисками, флешками и многими другими аппаратными компонентами, которые могут подключаться к ПК и вновь отсоединяться в ходе работы. Основная проблема системы udevзаключается в том, что создание файлов устройств при запуске ПК длится достаточно долго (несколько секунд). Поскольку часть этих файлов необходима для продолжения запуска (в частности, для доступа к жестким дискам и сетевым интерфейсам), отложить выполнение udevили запустить ее в фоновом режиме очень сложно.

1. Архивирование
   1. Общие сведения

Архивирование в ОС – это то же самое, что и сжатие файлов.

Архивирование – это объединение нескольких небольших файлов в один с целью более удобной последующей передачи, хранения, шифрования или сжатия. Архивация выполняется специальными утилитами.

Самой популярной для ОС утилитой для архивации является tar. Она используется почти всегда для архивации исходников, упаковки пакетов. Для сжатия используются другие утилиты, в зависимости от алгоритма сжатия, например zip, bz, xz, Lzma и т д. Сначала выполняется архивация, затем сжатие отдельными программами. Автоматический запуск некоторых утилит сжатия для только что созданного архива поддерживается в tar и других подобных программах с помощью специальных опций.

Также полезной возможностью архивации является шифрование.

* 1. Утилита Tar

Tar – это стандартная утилита, с помощью которой выполняется архивирование файлов ОС. Постепенно из небольшой программы архивации она превратилась в мощный инструмент, поддерживающий работу со многими вариантами архивов и алгоритмами сжатия. Программа поддерживает большое количество параметров.

Общий вид команды:

$ tar опции файл\_для\_записи /папка\_файлами\_для\_архива

Основные опции:

* A – добавить файл к архиву;
* c – создать архив;
* d – сравнить файлы архива и распакованные файлы в ФС;
* j – сжать архив с помощью Bzip;
* z – сжать архив с помощью Gzip;
* r – добавить файлы в конец архива;
* t – показать содержимое архива;
* u – обновить архив относительно ФС;
* x – извлечь файлы из архива;
* v – показать подробную информацию о процессе работы;
* f – файл для записи архива;
* -C – распаковать в указанную папку;
* --strip-components – отбросить n вложенных папок.

Чтобы создать архив, используют такую команду:

tar -cvf archive.tar.gz /PATH/to/files

Чтобы распаковать архив tar:

tar -xvf archive.tar.gz

Сжатый архив создается точно так же, только с опцией -z, с шифрованием gizp; если нужно bzip, то применяется опция -j:

tar -zcvf archive.tar.gz /PATH/to/files

$ tar -zxvf archive.tar.gz

Например, чтобы заархивировать папку в ОС:

tar -zcvf home.tar.gz ~/

По-другому тот же архив можно получить, если сначала создать обычный архив с помощью tar, а потом сжать его утилитой для сжатия, при этом остается больше контроля над процессом сжатия:

Gzip archive.tar

Также можно убрать сжатие:

gunzip archive.tar.gz

Чтобы добавить файл в архив, используют:

tar -rvf archive.tar file.txt

Для извлечения одного файла синтаксис тот же:

tar -xvf archive.tar file.txt

Можно извлечь несколько файлов по шаблону соответствия с помощью параметра wildcard, например, извлечь все php-файлы:

tar -xvf archive.tar --wildcards '\*.php'

По умолчанию распаковать архив tar можно в текущую папку с именем архива, а чтобы распаковать в нужную папку используют ключ -C:

tar -xvf archive.tar -C /PATH/to/dir

Далее рассматриваются альтернативы утилиты tar.

* 1. Утилита Shar

Утилита Shar позволяет создавать самораспаковывающиеся архивы. По сути, это скрипт оболочки, и для распаковки ему необходима оболочка Bash или другая совместимая с Bourne Shell. У Shar есть несколько преимуществ, но также она является потенциально небезопасной, так как архив представляет собой исполняемый файл.

Опции Shar:

* -o – сохранять архив в файл вместо стандартного вывода;
* -l – ограничить размер выходного файла;
* -L – ограничить размер выходного файла и разбить его на части;
* -n – имя архива не будет включено в заголовок;
* -a – разрешить автоматическую генерацию заголовков.

Примеры использования Shar для того, чтобы заархивировать папку:

Создать Shar–архив:

Shar file\_name.extension > filename.Shar

Распаковывать Shar–архив:

./filename.Shar

* 1. Утилита Ar

Ar – утилита для создания и управления архивами. В основном используется для архивации статических библиотек, но может быть использована для создания любых архивов. Раньше использовалась довольно часто, но была вытеснена утилитой tar. Сейчас используется только для создания и обновления файлов статических библиотек.

Опции Ar:

* -d – удалить модули из архива;
* -m – перемещение членов в архиве;
* -p – напечатать специфические члены архива;
* -q – быстрое добавление;
* -r – добавить члена к архиву;
* -s – создать индекс архива;
* -a – добавить новый файл к существующему архиву.

Создать статическую библиотеку libmath.a из объектных файлов substraction.o и division.o:

ar cr libmath.a substraction.o division.o

Далее необходимо извлечь файлы из архива:

ar x libmath.a

Таким образом можно распаковать любую статическую библиотеку.

* 1. Утилита Cpio

Улитита cpio (copy in and out – скопировать ввод и вывод) – это еще один стандартный архиватор для ОС. Активно используется в менеджере пакетов Red Hat, а также для создания initramfs. Архивация в ОС для обычных файлов с помощью этой программы не применяется.

Опции утилиты:

* -a – сбросить время обращения к файлам после их копирования;
* -A – добавить файл;
* -d – создать каталоги при необходимости.

Создать cpio-архив:

ls

file1.o file2.o file3.o

ls | cpio -ov > /PATH/to/output\_folder/obj.cpio

Чтобы распаковать архив:

cpio -idv < /PATH/to folder/obj.cpio

1. Сжатие
   1. Утилита Gzip

В качестве стандартной утилиты сжатия в ОС чаще всего применяется утилита Gzip. Для декомпрессии используются команды gunzipили Gzip -d.

Общий вид утилиты:

$ Gzip опции файл

$ gunzip опции файл

Опции:

* -c – выводить архив в стандартный вывод;
* -в– распаковать;
* -f – принудительно распаковывать или сжимать;
* -l – показать информацию об архиве;
* -r – рекурсивно перебирать каталоги;
* -0 – минимальный уровень сжатия;
* -9 – максимальный уровень сжатия.

Например, выполнить сжатие файла:

Gzip -c файл > архив.gz

А теперь распаковать:

gunzip -c архив.gz

Но чтобы сжать папку в ОС, придется сначала заархивировать ее с помощью tar, а уже потом сжать файл архива с помощью Gzip.

* 1. Утилита Bzip

bzip2– еще одна альтернативная утилита сжатия для ОС. Она более эффективная чем Gzip, но работает медленнее. Для распаковки используют утилиту bunzip2**.**

Опции утилиты bzip2 аналогичны опциям Gzip, которые рассмотрены в п.21.1.

Чтобы создать архив, используют:

bzip2 file

В текущем каталоге будет создан файл file.bz2.

* 1. Утилита Lzma

Lzma – новый и высокоэффективный алгоритм сжатия. Синтаксис и опции тоже похожи на Gzip. Для распаковки используют команду unLzma.

* 1. Утилита Xz

Xz – еще один высокоэффективный алгоритм сжатия, обратно совместимый с Lzma. Параметры вызова тоже похожи на Gzip.

* 1. Утилита Zip

Zip – кроссплатформенная утилита для создания сжатых архивов формата zip, совместимая с Windows-реализациями этого алгоритма. Zip-архивы очень часто используются для обмена файлами в сети Интернет. С помощью этой утилиты можно сжимать как файлы, так и папки.

Синтаксис утилиты:

$ zip опции файлы

$ unzip опции архив

Опции утилиты:

* -d – удалить файл из архива;
* -r – рекурсивно обходить каталоги;
* -0 – только архивировать, без сжатия;
* -9 – наилучшая степень сжатия;
* -F – исправить zip-файл;
* -e – шифровать файлы.

Чтобы создать Zip-архив, используют:

zip -r /PATH/to/files/\*

Для распаковки:

unzip archive.zip

* 1. Утилита RAR

Для работы с RAR-архивами используют утилиты rar для создания архивов и unrar для распаковки. Установить архиватор можно средствами стандартного пакетного менеджера ОС, но для установки последней актуальной версии используют другой способ:

1. скачать с официального сайта проекта www.rarlab.com архив с бинарными файлами и извлечь из него файлы rar и unrar; скопировать их, например, в /usr/bin/ либо произвести установку:

wget

tar xzf rar linux-x64-612.tar.gz

cd rar

make install

1. команда make install выполнит следующие действия согласно содержимому файла makefile:

mkdir -p /usr/local/bin

mkdir -p /usr/local/lib

cp rar unrar /usr/local/bin

cp rar files.lst /etc

cp default.sfx /usr/local/lib

Далее приводятся основные команды rar.

Создание архива файлов file01.sql, file02, file03:

rar a archive.rar file01.sql file02 file03

Создать архив директории dir01 рекурсивно:

rar -r a archive.rar dir01

Добавить файлы в архив можно с помощью ключа u, а удалить из архива ключом d**:**

rar u archive.rar file02

rar d archive.rar file02

Также можно указать степень сжатия архива; параметр сжатия принимает значения от 0 до 5, при этом 0 – без использования сжатия, по умолчанию используется степень 3:

rar a -m5 archive.rar file02

Для распаковки архива также можно использовать rar с опцией x:

rar x archive.rar

Для распаковки RAR-архивов используется утилита unrar. Распаковать архив:

unrar x archive.rar

Распаковать архив в определенную директорию:

unrar x archive.rar ./dir

Проверить, что архив целый:

unrar t archive.rar

Просмотреть список файлов в архиве:

unrar l archive.rar

1. Создание разделов диска
   1. Общие сведения

Раздел диска – это часть долговременной памяти накопителя данных, логически выделенная для удобства работы и состоящая из смежных блоков. Информация о разделах хранится в таблице разделов. Эта таблица включает информацию о первом и последнем секторах раздела и его типе, а также дополнительные сведения о каждом разделе.

Обычно каждый раздел рассматривается ОС как отдельный логический диск, даже если все они находятся на одном физическом носителе. В системах Windows им присваиваются такие буквы, как C: (исторически основной диск), D: и так далее. В ОС каждый раздел назначается каталогу /dev, например /dev/sda1 или /dev/sda2.

Существует два основных способа хранения информации о разделах на жестких дисках. Первый MBR (основная загрузочная запись), а второй GPT (таблица разделов GUID).

* 1. MBR

MBR – это пережиток первых дней MS-DOS, который на протяжении десятилетий был стандартной схемой разбиения на ПК. Таблица разделов хранится в первом секторе диска, называемом загрузочным сектором, вместе с загрузчиком, которым в ОС является загрузчик GRUB. Но MBR имеет ряд ограничений, которые препятствуют ее использованию: невозможность адресации дисков размером более 2 ТБ и возможность создать только 4 первичных раздела на диске.

* + 1. Управление разделами MBR

Стандартной утилитой для управления разделами MBR в ОС является fdisk – это интерактивная утилита с меню. В качестве параметра нужно задать имя устройства, соответствующее диску, который требуется отредактировать. Например, команда:

# fdisk /dev/sda

отредактирует таблицу разделов первого устройства, подключенного по SATA (sda) в ОС. Следует иметь в виду, что нужно указать устройство, соответствующее физическому диску, а не один из его разделов (например, /dev/sda1).

Все операции с дисками в данном контексте необходимо выполнять от имени пользователя root (системного администратора) или с привилегиями root, используя sudo.

При вызове fdisk покажет приветствие, затем предупреждение и будет ждать ваших команд.

# fdisk /dev/sda

Welcome to fdisk (util-linux 2.33.1).

Changes will remain in memory only, until you decide to write them.

Be careful before using the write command.

​Command (m for help):

Следует обратить внимание, что можно создавать, редактировать или удалять разделы по своему усмотрению и ничего не будет записано на диск, пока не будет использована команда записи w(write). Таким образом, можно "практиковаться" без риска потери данных. Чтобы выйти из fdisk без сохранения изменений, используют команду q(quit).

При этом не следует практиковаться на важном диске, так как всегда есть риск. Вместо этого используют запасной внешний диск или флеш-накопитель USB.

* + 1. Печать текущей таблицы разделов

Команда p(print)используется для печати текущей таблицы разделов. Вывод на экран выглядит примерно следующим образом:

Command (m for help): p

Disk /dev/sda: 111.8 GiB, 120034123776 bytes, 234441648 sectors

Disk model: CT120BX500SSD1

Units: sectors of 1 \* 512 = 512 bytes

Sector size (logical/physical): 512 bytes / 512 bytes

I/O size (minimum/optimal): 512 bytes / 512 bytes

Disklabel type: dos

Disk identifier: 0x97f8fef5

​

Device Boot start End Sectors Size Id Type

/dev/sda1 4096 226048942 226044847 107.8G 83 Linux

/dev/sda2 226048944 234437550 8388607 4G 82 Linux swap / Solaris

Значения столбцов:

* Device – устройство, назначенное разделу;
* Boot – показывает, является ли раздел "загрузочным" или нет;
* Start – сектор, с которого начинается раздел;
* End – сектор, где заканчивается раздел;
* Sectors – общее количество секторов в разделе. Умножают его на размер сектора, чтобы получить размер раздела в байтах;
* Size – размер раздела в "удобочитаемом" формате. В приведенном выше примере значения указаны в гигабайтах;
* Id – числовое значение, представляющее тип раздела;
* Type – описание типа раздела.

Для получения аналогичной информации можно выполнить в консоли команду fdisk с опцией -l:

fdisk -l

​

Disk /dev/sda: 40 GiB, 42949672960 bytes, 83886080 sectors

Disk model: VMware Virtual S

Units: sectors of 1 \* 512 = 512 bytes

Sector size (logical/physical): 512 bytes / 512 bytes

I/O size (minimum/optimal): 512 bytes / 512 bytes

Disklabel type: dos

Disk identifier: 0x00000000

* + 1. Первичные и расширенные разделы

На MBR-диске может быть 2 основных типа разделов: **основной первичный (primary)** и **расширенный (extended***)*. Если требуется сделать диск "загрузочным", первый раздел должен быть основным. Следует учесть, что в таблице MBR может быть только 4 основных раздела на диске.

Один из способов обойти это ограничение – создать расширенный раздел, который действует как контейнер для логических разделов. Например, могут быть: основной раздел, дополнительный раздел, занимающий оставшуюся часть дискового пространства, и пять логических разделов внутри него.

В ОС первичный и расширенный разделы обрабатываются одинаково, поэтому нет никаких преимуществ использования одного над другим.

* + 1. Создание раздела

Чтобы создать раздел, используют команду n(new). По умолчанию разделы будут создаваться в начале нераспределенного пространства на диске. В процессе будет предложено указать тип раздела (основной или расширенный), первый сектор и последний сектор.

Обычно для первого сектора используют значение по умолчанию, предложенное fdisk, если только не нужно, чтобы раздел начинался с определенного сектора. Вместо указания последнего сектора можно указать размер, за которым следуют буквы K, M, G, T или P (Kilo, Mega, Giga, Tera или Peta). Например, если нужно создать раздел размером 1 ГБ, то вместо последнего сектора указывают +1G. Ниже пример создания основного раздела:

Command (m for help): n

Partition type

p primary (0 primary, 0 extended, 4 free)

e extended (container for logical partitions)

Select (default p): p

Partition number (1-4, default 1): 1

First sector (2048-3903577, default 2048): 2048

Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-3903577, default 3903577): +1G

* + 1. Проверка нераспределенного пространства диска

Если неизвестно сколько свободного места осталось на диске, то можно использовать команду F(free), чтобы показать нераспределенное пространство, например:

Command (m for help): F

Unpartitioned space /dev/sdd: 881 MiB, 923841536 bytes, 1804378 sectors

Units: sectors of 1 \* 512 = 512 bytes

Sector size (logical/physical): 512 bytes / 512 bytes

​

start End Sectors Size

2099200 3903577 1804378 881M

* + 1. Удаление разделов

Чтобы удалить раздел, используют команду d(delete). fdisk запросит номер раздела, который требуется удалить, если количество разделов больше одного. Если на диске всего один раздел, то он будет удален сразу.

Следует иметь в виду, что в случае удаления расширенного раздела, все логические разделы внутри него также будут удалены.

Также следует иметь в виду, что при создании нового раздела с помощью fdisk максимальный размер будет ограничен максимальным объемомнепрерывного свободного пространства на диске.

Например, на диске имеются следующие разделы:

Device Boot start End Sectors Size Id Type

/dev/sdd1 2048 1050623 1048576 512M 83 Linux

/dev/sdd2 1050624 2099199 1048576 512M 83 Linux

/dev/sdd3 2099200 3147775 1048576 512M 83 Linux

Затем удаляются раздел 2 (/dev/sdd2) и проверяется наличие свободного места:

Command (m for help): F

Unpartitioned space /dev/sdd: 881 MiB, 923841536 bytes, 1804378 sectors

Units: sectors of 1 \* 512 = 512 bytes

Sector size (logical/physical): 512 bytes / 512 bytes

​

start End Sectors Size

1050624 2099199 1048576 512M

3147776 3903577 755802 369M

Суммарно доступно 881 МБ нераспределенного пространства.

Но при создании раздела размером 700 МБ происходит следующее:

Command (m for help): n

Partition type

p primary (2 primary, 0 extended, 2 free)

e extended (container for logical partitions)

Select (default p): p

Partition number (2,4, default 2): 2

First sector (1050624-3903577, default 1050624):

Last sector, +/-sectors or +/-size{K,M,G,T,P} (1050624-2099199, default 2099199): +700M

Value out of range.

Параметр "Value out of range" говорит о том, что допустимое значение размера раздела было превышено. Это происходит потому, что самым большим непрерывным нераспределенным пространством на диске является блок размером 512 МБ, который принадлежал разделу 2, а следующее нераспределенное пространство находится после раздела 3 (/dev/sdd3). Fdisk не может объединить нераспределенные участки. разделенные разделами.

* + 1. Изменение типа раздела

Иногда может понадобиться изменить тип раздела, особенно при работе с дисками, которые будут использоваться в других ОС и платформах. Сделать это можно с помощью команды t(type), за которой следует номер раздела, подлежащий изменению.

Тип раздела должен быть указан соответствующим шестнадцатеричным кодом. Посмотреть список всех допустимых кодов можно используя команду l(list).

Не следует путать тип раздела с используемой на нем ФС. В изначальном замысле между ними существовала непосредственная зависимость, но в настоящее время это не совсем так. Например, раздел ОС может содержать любую собственную файловую систему ОС, такую как ext4 или ReiserFS.

Разделы ОС имеют тип 83(Linux). Разделы подкачки бывают типа 82(Linux Swap).

* 1. GPT

GPT – система разделов, которая устраняет многие ограничения MBR. Практических ограничений на размер диска нет, а максимальное количество разделов ограничено только самой ОС. Чаще встречается на более современных ПК, использующих UEFI вместо старого BIOS.

При выполнении задач системного администрирования весьма вероятно, что будут встречаться обе схемы хранения данных, поэтому важно знать, как использовать инструменты для создания, удаления или изменения разделов применительно к каждой из них.

* + 1. Управление разделами GPT

Утилита gdisk является эквивалентом fdisk при работе с дисками с разделами GPT.

Интерфейс смоделирован аналогично fdisk с интерактивной командной строкой и теми же (или очень похожими) командами.

* + 1. Таблица разделов

Команда p используется для печати текущей таблицы разделов. Результат примерно такой:

Command (? for help): p

Disk /dev/sdb: 3903578 sectors, 1.9 GiB

Model: DataTraveler 2.0

Sector size (logical/physical): 512/512 bytes

Disk identifier (GUID): AB41B5AA-A217-4D1E-8200-E062C54285BE

Partition table holds up to 128 entries

Main partition table begins at sector 2 and ends at sector 33

First usable sector is 34, last usable sector is 3903544

Partitions will be aligned on 2048-sector boundaries

Total free space is 1282071 sectors (626.0 MiB)

​

Number start (sector) End (sector) Size Code Name

1 2048 2099199 1024.0 MiB 8300 Linux filesystem

2 2623488 3147775 256.0 MiB 8300 Linux filesystem

И сразу же можно заметить некоторые особенности:

* Каждый диск имеет уникальный идентификатор диска (GUID). Это 128-битное шестнадцатеричное число, назначаемое случайным образом при создании таблицы разделов. Поскольку существует (3.4×1038) возможных значений этого числа, вероятность того, что 2 случайных диска имеют один и тот же GUID, довольно мала. GUID можно использовать для определения того, какие ФС монтировать во время загрузки и в каком месте. Таким образом устраняется необходимость использования для этого пути устройства (например, /dev/sdb).
* См. фразу "Partition table holds up to 128 entries". Действительно на GPT-диске может быть до 128 разделов. Из-за этого нет необходимости в первичных и расширенных разделах.
* Свободное место указано в последней строке, поэтому нет необходимости в эквиваленте команды F из fdisk.
  + 1. Создание раздела

Для создания раздела используют команду n, как и в fdisk. Основное отличие заключается в том, что во время создания также можно указать тип раздела, помимо номера раздела, первого и последнего сектора (или размера). Разделы GPT поддерживают гораздо больше типов, чем MBR. Можно проверить список всех поддерживаемых типов с помощью команды l.

* + 1. Удаление раздела

Чтобы удалить раздел, вводят команду d и номер раздела. В отличие от fdisk первый раздел не будет удален автоматически, если он является единственным на диске.

На GPT-дисках разделы можно легко переупорядочить, чтобы избежать пробелов в последовательной нумерации. Для этого используют команду s. Например, на диске имеется следующая таблица разделов:

Number start (sector) End (sector) Size Code Name

1 2048 2099199 1024.0 MiB 8300 Linux filesystem

2 2099200 2361343 128.0 MiB 8300 Linux filesystem

3 2361344 2623487 128.0 MiB 8300 Linux filesystem

Если удалить второй раздел, таблица изменится:

Number start (sector) End (sector) Size Code Name

1 2048 2099199 1024.0 MiB 8300 Linux filesystem

3 2361344 2623487 128.0 MiB 8300 Linux filesystem

Если использовать команду s, таблица будет выглядеть следующим образом:

Number start (sector) End (sector) Size Code Name

1 2048 2099199 1024.0 MiB 8300 Linux filesystem

2 2361344 2623487 128.0 MiB 8300 Linux filesystem

Следует обратить внимание, что третий раздел стал вторым.

Примечание – В отличие от MBR-дисков, при создании раздела на GPT-дисках можно задавать размер, используя все доступное нераспределенное пространство, независимо от месторасположения на диске.

* + 1. Варианты восстановления

На GPT-дисках хранятся резервные копии заголовка GPT и таблицы разделов, что упрощает восстановление дисков в случае повреждения этих данных. В gdisk для доступа к функциям восстановления диска используют команду r(recovery).

Возможно восстановить поврежденный главный заголовок GPT или таблицу разделов с помощью команд b и c соответственно. Кроме того, используют главный заголовок и таблицу разделов для восстановления резервной копии с помощью команд d и e.

Также можно преобразовать MBR в GPT с помощью f и наоборот – g.

Чтобы получить список всех доступных команд восстановления и описания их действий, используют ? в режиме восстановления.

* + 1. Работа с дисками

Установленного в системе жесткого диска может быть недостаточно и иногда встает вопрос о добавлении дополнительного носителя. Далее рассматривается монтирование диска на физическую или виртуальную машину.

* + - 1. Подготовка

Перед началом работы следует убедиться существует ли диск в Системе. Иногда наличие устройства в BIOS может быть недостаточным. Доступные накопители проверяем командой:

sudo fdisk -l

* + - 1. Разметка

Для разметки диска запускают утилиту fdisk с указанием пути до диска:

fdisk /dev/sdb

При нажатии "m" и подтверждении ввода клавишей Enter, программа предоставит страницу доступных команд:

Справка:

DOS (MBR)

a переключить флаг загрузочного раздела

b редактировать вложенную BSD метку диска

c переключить флаг совместимости с DOS

Общие

d удалить раздел

F показать свободное неразмеченное пространство

l список известных типов разделов

n добавить новый раздел

p показать таблицу разделов

t изменить тип раздела

v проверить таблицу разделов

i вывести информацию о разделе

Разное

m показать это меню

u изменить единицы отображения/ввода

x дополнительные функции (только для экспертов)

Script

I загрузить разметку диска из файла сценария sfdisk

O сохранить разметку диска в файл сценария sfdisk

Сохранить и выйти

w сохранить таблицу на диск и выйти

q выйти без сохранения изменений

Создать новую метку

g создать новую пустую таблицу разделов GPT

G создать новую пустую таблицу разделов SGI (IRIX)

o создать новую пустую таблицу разделов DOS

s создать новую пустую таблицу разделов Sun

Так как предполагается создавать простой раздел (не загрузочный) и используется все дисковое пространство, следует нажать клавишу "n" и Enter.

Система запросит тип раздела выбратьp(primary) – первичный.

Номер раздела – 1**.**

На вопрос о первом и последнем секторе нажать клавишу Enter**.**

**Следует обратить внимание**, что если требуется создать несколько логических дисков, то в ответе на последний сектор следует указать размер в kilo-, mega-, giga-, tera-, petabytes. Соответственно, если раздел нужен размером в 2 гигабайта, то нужно указать 2G.

Далее нужно сохранить изменения нажатием "w" и подтвердить выбор с помощью клавиши Enter.

* + - 1. Форматирование

После выполнения операции, описанной выше, в Системе будет создано устройство /dev/sdb1 по своей сути это и есть раздел на диске.

ОС предлагает на выбор несколько вариантов ФС. Создание ФС происходит выполнением команды mkfs с указанием ключей либо выполнением одной из программ:

mkfs.bfs

mkfs.btrfs

mkfs.cramfs

mkfs.ext2

mkfs.ext3

mkfs.ext4

mkfs.ext4dev

mkfs.fat

mkfs.minix

mkfs.msdos

mkfs.ntfs

mkfs.vfat

mkfs.xfs

Форматирование выполняется командой:

sudo mkfs.ext4 /dev/sdb1

Диск готов к работе. Остается только смонтировать его.

* + - 1. Монтирование

В ОС существует возможность монтировать диск (либо иное блочное устройство) в любую из директорий, которая должна быть пустой:

1. создать каталог в директории /mnt:

sudo mkdir /mnt/1

1. изменить права доступа к каталогу; всем разрешаем всё:

sudo chmod -R 0777 /mnt/1

1. монтировать:

sudo mount /dev/sdb1 /mnt/1

Для монтирования диска автоматически при загрузке Системы необходимо редактировать файл /etc/fstab с помощью любого текстового редактора, например nano:

sudo nano /etc/fstab

Далее в самый конец файла вставить строку и сохранить файл:

/dev/sdb1 /mnt/1 ext4 defaults 0 0

* + 1. Образы дисков
       1. Создание

Чтобы записать информацию на чистый диск, нужно сначала создать образ записываемого диска. Есть несколько способов сделать это.

**Первый способ** применим в случае, если требуется скопировать существующий диск; для этого нужно поместить диск в привод, смонтировать диск, перейти в каталог размещения образа и выполнить команду, например:

$ dd if=/dev/cdrom of=mycd.iso

где:

* dd – стандартная программа, которая копирует содержимое одного файла в другой; можно использовать для создания образов любых дисков от дискет до DVD и записи на соответствующие носители;
* if – входной файл (input file);
* of – выходной файл (output file);
* /dev/cdrom – стандартная ссылка на устройство CD/DVD-привода; вместо нее можно подставить и настоящее имя устройства (например, /dev/hdd);
* mycd.iso – имя файла образа, который будет создан из помещенного в привод диска.

Примечание – Расширение iso дань традиции. По умолчанию dd читает с устройства блоками по 512 байт. Это значение можно изменить параметром bs (например, bs=1024). Для принудительного копирования даже после ошибок чтения можно добавить ключ noerror. Создание копии при помощи dd может не сработать при использовании двухслойных DVD. В остальных случаях обычно работает.

В результате на выходе получается точная копия диска. Но таким образом не получится скопировать ни аудиодиск, ни какой-нибудь другой защищенный диск с игрой (последний перепишется, но распознаваться как "правильный", лицензионный не будет). Кроме того, dd не показывает данные о процессе выполняемого копирования для этого можно лишь наблюдать за работой привода или проверять изменения в размере записываемого файла-образа.

**Второй способ** создания образа диска актуален для случаев, когда нужно записать на диск каталог с файлами или множество таких каталогов. Для этого используется программа genisoimage, которая умеет делать образ диска из директории таким образом:

$ genisoimage -f -v -J -o mycd.iso /mnt/somedisk/temp

В этом примере в текущем каталоге создается образ каталога /mnt/somedisk/temp и записывается в файл mycd.iso. Параметр -vуказывает, что genisoimage должна информировать о ходе процесса в консоль, -J задает формат файловой системы (Joliet), а -f указывает genisoimage следовать по символическим ссылкам, помещенным в копируемый каталог.

Итак, чтобы сделать "виртуальный проект" диска, нужно создать некий временный каталог и поместить в него символические ссылки на директории или файлы, которые требуются в проекте. В консоли это делать неудобно, поэтому можно воспользоваться для создания этих ссылок консольным файловым менеджером Midnight Commander:

1. запустить mc;
2. на одной панели перейти в каталог для создания диска, а на другой панели в каталог расположения файла или папки, символическую ссылку на который требуется добавить в проект;
3. поместить курсор на этот файл или каталог, нажать Ctrl+X;
4. нажать клавишу "S"; появится диалоговое окно с уже заполненными полями (на что делать ссылку и куда ее помещать);
5. подтвердить нажатием Enter;
6. таким же образом добавить в проект другие файлы и каталоги;
7. передать каталог проекта утилите genisoimage и создать образ.

Примечание – ОС может плохо считывать с носителя большие видеофайлы, если на диске нет файловой системы UDF (Universal Disc Format). На съемном носителе могут соседствовать разные файловые системы: по умолчанию ISO-9660, также Joliet надстройка над ISO-9660, разрешающая длинные имена файлов, причем даже в юникоде; Linux и Windows хорошо понимают Joliet.

Поддержку UDF надо обязательно включать, если записываетcя стандартный DVD с видео, т.е. с четкой структурой каталогов VIDEO\_TS (с фильмом, состоящем из VOB-файлов) и иногда AUDIO\_TS. Есть два способа включить UDF для образа диска: первый – использовать ключ -udf, а второй – это "ориентированный" на DVD Video параметр -dvd-video. В последнем случае нужно, чтобы имена файлов и каталогов (VIDEO\_TS и прочих) были в верхнем регистре. При создании образа файлы (их содержимое) будут отсортированы и расположены в образе диска. Сортировка не произойдет, если все названия фигурируют не в верхнем регистре.

Другое расширение к ISO-9660 – это Rock Ridge. Оно по умолчанию включено в k3b и дает возможность использовать более длинные, чем в Joliet, имена файлов, а также б*о*льшую глубину вложенности каталогов и атрибуты файлов UNIX (следует обратить на это особое внимание). Rock Ridge включается параметром -r.

* + - 1. Запись

После подготовки образа его нужно записать на диск. В этом поможет программа wodim.

В простейшем случае ее запуск может выглядеть так:

# wodim dev=/dev/hdc -eject -v mycd.iso

Вместо /dev/hdc можно использовать другое устройство, а mycd.iso – это название файла с образом. Параметр -eject выдвигает каретку привода после окончания записи. У wodim есть ряд других интересных параметров:

* -dummy – симуляция прожига с выключенным лазером; полезно, чтобы определить время запись;
* -multi – многосессионный диск;
* -fix – фиксация диска (без записи);
* speed=значение – скорость записи;
* blank=режим – очистка содержимого CD-RW; доступные режимы: all (полная очистка, долго), fast (быстро), session (очистить последнюю сессию);
* -format – форматирование болванки CD/DVD-RW;
* -overburn – включение возможности записи больше стандартного объема диска; в этом случае, как правило, необходимо также использовать режим записи SAO (-sao).

1. Написание bash-скриптов
   1. Команды

Командный арсенал оболочки bash является средством автоматизации и позволяет успешно писать простые сценарии, иначе системным администраторам пришлось бы вручную вводить команды в командную строку. Опыт использования командных строк необходим для создания bash-сценариев (и наоборот). При превышении объема в сотню строк в bash-сценарии или необходимости применения средств, которыми bash не обладает, рекомендуется переходить к языкам программирования Perl или Python.

В среде bash комментарии начинаются со знака "#" и продолжаются до конца строки. Как и в командной строке, можно разбить одну логическую строку на несколько физических строк, обозначив переход на новую строку символом обратной косой черты "\". И, наоборот, можно поместить на одной строке несколько операторов, разделив их точкой с запятой.

Сценарий для оболочки bash может состоять только из последовательности командных строк. Например, следующий сценарий helloworld просто выполняет команду echo.

#!/bin/bash

echo "Hello, world!"

Первая строка содержит описательный оператор, который сообщает, что данный текстовый файл является сценарием, предназначенным для интерпретации командной оболочкой /bin/bash. При принятии решения о том, как выполнить этот файл, ядро отыщет соответствующий синтаксис. С точки зрения оболочки, выполняющей этот сценарий, первая строка представляет собой простой комментарий. Если бы оболочка находилась в другом каталоге, пришлось бы изменить эту строку.

Для того чтобы подготовить файл сценария к выполнению, достаточно установить его атрибут, отвечающий за выполнение:

$ chmod +х helloworld

$./helloworld

Hello, world!

Если оболочка понимает команду helloworld без префикса ./, это означает, что в пути поиска указан текущий каталог ".". И это небезопасный вариант, поскольку дает другим пользователям возможность устраивать “ловушки” при попытке выполнения определенных команд при переходе к каталогу, в котором они имеют доступ для записи.

Можно также непосредственно запустить (активизировать) оболочку в качестве интерпретатора сценария:

$ bash helloworld

Hello, world!

$ source helloworld

Hello, world!

Первая команда выполнит сценарий helloworld в новом экземпляре оболочки bash, а вторая заставит начальную оболочку прочитать содержимое файла, а затем выполнить его. Второй вариант полезен в случае, если сценарий устанавливает переменные среды или выполняет другие операции настройки, применимые только к текущей оболочке. Обычно этот вариант используется при написании сценариев, включающих содержимое файла конфигурации, написанного в виде ряда присваиваний значений bash-переменным.

По опыту работы с Windows привычнее использовать понятие расширения файла, по которому можно судить о типе файла, а также о том, можно ли его выполнить. В ОС признак того, может ли файл быть выполнен и кем именно, содержится в специальных битах полномочий. При желании можно наделить bash-сценарии суффиксом .sh, который бы будет сообщать о типе, но при выполнении соответствующей команды потребует вводить суффикс .sh, так как в ОС расширения не интерпретируются специальным образом.

* 1. Сценарии

Перед переходом к особенностям написания bash-сценариев следует изучить методику этого процесса. Многие администраторы пишут bash-сценарии так же, как Perl- или Python-сценарии, т.е. используя какой-либо текстовый редактор, но рекомендуется для удобства рассматривать в качестве интерактивной среды разработки сценариев режим с приглашением на ввод команды.

В качестве примера можно рассмотреть наличие журналов регистрации, именованных с суффиксами .log и .LOG и размещенных по всей иерархической структуре каталогов, необходимость изменения этих суффиксы, приведя их наименования к прописному виду.

Сначала нужно отыскать все эти файлы:

$ find. -name '\*log'

.do-not-touch/important.log

admin.com-log/

foo.log

genius/spew.log

leather\_flog

Далее необходимо включить в шаблон точку и игнорировать при поиске каталоги, для чего нажать комбинацию клавиш Ctrl+P, чтобы вернуть команду в командную строку, а затем модифицировать ее.

$ find.-type f -name '\*.log'

.do-not-touch/important.log

foo.log

genius/spew.log

С внесением таких изменений вывод команды становится более понятным. При необходимости можно избавиться от каталога .do-not-touch следующим образом:

$ find. -type f -name '\*.log ' | grep -v.do-not-touch

foo.log

genius/spew.log

Теперь получен абсолютно точный список файлов, которые должны быть переименованы.

Далее можно попробовать сгенерировать несколько новых имен.

$ find. -type f -name '\*.log ' I grep -v.do-not-touch | while read fname

> do

> echo mv $ fname $ {fname/. log/. LOG/}

> done

mv foo.log foo.LOG

mv genius/spew.log genius/spew.LOG

Эти команды позволяют переименовать нужные файлы.

В реальности можно снова вызвать уже выполненную команду и отредактировать команду echo, чтобы заставить оболочку bash выполнять команды mv, а не просто выводить их, так как передача команд в отдельную копию оболочки bash более надежный вариант работы, который к тому же требует меньшего объема редактирования.

При нажатии комбинации клавиш Ctrl+P оболочка bash сворачивает мини-сценарий данного примера в одну. К этой уплотненной командной строке нужно добавить канал, передающий выходные данные команде bash -х.

$ find. -type f -name '\*.log ' I grep -v.do-not-touch | while read fname;

do echo mv $fname $ {fname/. log/. LOG/}; done | bash -x

+ mv foo.log foo.LOG

+ mv genius/spew.log genius/spew.LOG

Ключ -x команды bash обеспечивает вывод каждой команды перед ее выполнением.

Теперь переименование файлов завершено, но полезно сохранить этот сценарий, чтобы воспользоваться им вновь. Встроенная в bash команда fc по своему действию во многом аналогична нажатию комбинации клавиш Ctrl+P, но вместо возврата последней команды в командную строку она передает команду в заданный редактор. Следует добавить в нужный файл строку идентификационного комментария, поместить сохраненный файл в приемлемый каталог (например, ~/bin или /usr/local/bin), сделать файл исполняемым, чтобы в итоге получить настоящий сценарий.

Рекомендации по использованию сценариев:

* Разработка сценария (или его части) в виде конвейера команд, причем пошагово и в режиме выполнения командных строк.
* Пересылка результата в стандартный выходной поток, проверяя правильность работы используемых команд.
* Использование на каждом этапе буфера ранее выполненных команд для их появления в командной строке и инструментов редактирования для их модификации.
* Сохранение всех команд сценария до тех пор, пока они команды не заработают должным образом.
* Выполнение команды на реальном примере при получении правильного результата, чтоб убедиться в достоверности.
* Использование команды fс для фиксации работы в редакторе, оформления ее соответствующим образом и сохранения.

В приведенном выше примере были выведены командные строки, а затем направлены в подоболочку для выполнения. Этот метод не является универсально применимым, но часто оказывается полезным. В качестве альтернативного варианта можно фиксировать результаты, перенаправляя их в файл.

Рекомендуется добиваться при разработке сценариев получения нужных результатов, и до этих пор не выполнять никаких потенциально деструктивных действий.

* 1. Ввод и вывод данных

Команда echo не отличается интеллектуальностью, но зато проста в применении. Для получения большего контроля над выводом данных используют команду printf, которая менее удобна, поскольку предполагает указание в нужных местах символов перехода на новую строку "\n", но при этом позволяет использовать символ табуляции и другие средства форматирования результата.

Для сравнения результаты выполнения следующих двух команд:

$ echo "\taa\tbb\tcc\n"

\taa\tbb\tcc\n

$ printf "\taa\tbb\tcc\n"

аа bb сс

Для того чтобы сформировать пользователю приглашение ввести данные, можно использовать команду read:

#!/bin/bash

echo -n "Ввести свое имя: "

read user\_name

if [ -n "$user\_name" ]; then

echo "Привет $user\_name!"

exit 0

else

echo "Вы не назвали свое имя!"

exit 1

fi

* 1. Функции и аргументы

Аргументы командной строки служат для сценария переменными с числовыми именами: $1 первый аргумент командной строки, $2 второй и т.д. Аргумент $0 содержит имя, по которому был вызван сценарий, например ./bin/example.sh, т.е. это не фиксированное значение.

Переменная $# содержит количество переданных аргументов командной строки, а переменная $\* все эти аргументы. Ни одна из этих переменных не учитывает аргумент $0.

При вызове сценария с некорректными аргументами или вообще без них должно быть выведено короткое сообщение с напоминанием о том, как следует использовать данный сценарий. Приведенный ниже сценарий, принимающий два аргумента, проверяет, являются ли оба эти аргумента каталогами, и, убедившись в этом, отображает их. Если аргументы оказываются недопустимыми, сценарий выводит сообщение о его правильном использовании и завершается с ненулевым значением кода возврата. Если программа, вызвавшая этот сценарий, проверит код возврата, она зафиксирует, что сценарий не удалось выполнить корректно.

#!/bin/bash

function show\_usage {

echo "Использование: $0 source\_dir dest\_dir"

exit 1

}

# Основная программа начинается здесь

if [ $# -ne 2 ]; then

show\_usage

else # Существуют два аргумента

if [ -d $1 ]; then

source\_dir=$l

else

echo 'Недопустимый каталог-источник'

show\_usage

fi

if [ -d $2 ]; then

dest\_dir=$2

else

echo 'Недопустимый каталог-приемник'

show\_usage

fi

fi

printf "Каталог-источник: ${source\_dir}\n"

printf "Каталог-приемник: ${dest\_dir}\n"

Для вывода сообщения о правильном использовании данного сценария была создана отдельная функция show\_usage. Если бы позже этот сценарий был модифицирован и стал бы принимать дополнительные аргументы, это сообщение нужно было бы изменить только в одном месте.

$ mkdir ааа bbb

$ sh showusage ааа bbb

Каталог-источник: ааа

Каталог-приемник: bbb

$ sh showusage foo bar

Недопустимый каталог-источник

Использование: showusage source\_\_dir dest\_dir

Аргументы bash-функций обрабатываются практически так же, как и аргументы командной строки: $1 первый аргумент командной строки, $2 второй и т.д. Как видно из приведенного выше примера, аргумент $0 содержит имя сценария.

Для того чтобы сделать предыдущий пример более устойчивым к ошибкам, можно передавать функции show\_usage в качестве аргумента код ошибки. Это позволило бы конкретизировать код возврата для каждого типа отказа. Реализация этой идеи показана в следующем фрагменте программы:

function show\_usage {

echo "Использование: $0 source\_dir dest\_dir"

if [ $# -eq 0 ]; then

exit 99 # Выход с любым ненулевым кодом возврата

else

exit $1

fi

}

В этой версии функции добавляется анализ наличия аргументов. Внутри функции переменная $# сообщает, сколько аргументов было ей передано. Сценарий завершится с кодом возврата 99, если при его вызове не было передано никаких аргументов. Но при передаче конкретного значения, например "show\_usage 5" сценарий завершится с этим кодом после вывода сообщения об использовании сценария (переменная $? содержит статус завершения последней выполненной команды, независимо от факта использования в сценарии или в командной строке).

Между функциями и командами в оболочке bash существует строгая аналогия. Можно определить полезные функции в файле ~/.bash\_profile, а затем использовать их в командной строке как команды. Например, если узел стандартизировал в сети порт 7988 для протокола SSH, можно определить в файле ~/.bash\_profile функцию ssh, чтобы она всегда запускалась (как команда) с ключом -р 7988.

function ssh {

/usr/bin/ssh -р 7988 $\*

}

Подобно многим командным оболочкам, bash обладает механизмом псевдонимов (alias), который может репродуцировать этот ограниченный пример в еще более лаконичном виде, тем не менее функции остаются более универсальным и более мощным средством.

* 1. Области видимости и потоки

Переменные в рамках сценария имеют глобальный статус, но функции могут создавать собственные локальные переменные с помощью объявления local.

Следующий код используется для примера:

#!/bin/bash

​

function localizer {

echo "==> В функции localizer начальное значение а равно '$а' "

local а

echo "==> После объявления local значение а стало равным '$а' "

a="localizer version"

echo "==> При выходе из функции localizer значение а равно '$а' "

}

a="test"

echo "До вызова функции localizer значение а равно '$а' "

localizer

echo "После вызова функции localizer значение а равно '$а' "

По приведенным ниже результатам выполнения сценария scopetest.sh видно, что локальная версия переменной $а внутри функции localizer перезаписывает глобальную переменную $а. Глобальная переменная $а видна в функции localizer до тех пор, пока не встретится объявление local, т.е. объявление local работает как команда, которая в момент выполнения создает локальную переменную.

При запуске скрипта:

$ sh scopetest.sh

до вызова функции localizer значение а равно "test":

==> В функции localizer начальное значение а равно "test"

==> После объявления local значение а стало равным ""

==> При выходе из функции localizer значение а равно "localizer version".

После вызова функции localizer значение а равно "test".

* 1. Операторы

Терминатором (признаком конца) для условного оператора if служит оператор fi. Для образования цепочки if-операторов можно использовать ключевое слово elif, означающее "else if".

if [ $base -eq 1 ] && [ $dm -eq 1 ]; then

installDMBase

elif [ $base -ne 1 ] && [ $dm -eq 1 ]; then

installBase

elif [ $base -eq 1 ] && [ $dm -ne 1 ]; then

installDM

else

echo '==> Installing nothing'

fi

Как и специальный [] -синтаксис для выполнения операций сравнения, так и ключеподобные имена операторов целочисленного сравнения (например -eq) происходят от использования утилиты /bin/test старых оболочек. В действительности квадратные скобки – это не что иное, как условное обозначение вызова утилиты test; они не являются частью оператора if.

Ниже (Таблица 24) собраны bash-операторы сравнения для чисел и строк. В отличие от Perl, в bash используются текстовые операторы для чисел и символические операторы для строк.

Таблица 24 - Операторы сравнения чисел и строк

|  |  |  |
| --- | --- | --- |
| **Строки** | **Числа** | **Истина, если** |
| x = y | x -eq y | x равно y |
| x != y | x -ne y | x не равно y |
| x < y | x -lt y | x меньше y |
| x <= y | x – le y | x меньше или равно y |
| x > y | x – gt y | x больше y |
| x >=y | x -ge y | x больше или равно y |
| -n x | - | x не равно значению null |
| -n x | - | x равно значению null |

В оболочке bash предусмотрены возможности оценки свойств файлов. Некоторые из операторов тестирования и сравнения файлов приведены ниже (Таблица 25).

Таблица 25 - Операторы оценки файлов

|  |  |
| --- | --- |
| **Оператор** | **Истина, если** |
| -d файл | Файл файл существует и является каталогом |
| -e файл | Файл файл существует |
| -f файл | Файл файл существует и является обычным |
| -r файл | У вас есть право доступа для чтения файла |
| -s файл | Файл файл существует, и он не пустой |
| -w файл | У вас есть право доступа для записи в файл |
| файл1 -nt файл2 | Файл файл1 новее, чем файл2 |
| файл1 -ot файл2 | Файл файл1 старше, чем файл2 |

Несмотря на всю полезность формы elif, зачастую лучше (с точки зрения ясности программного кода) использовать case-структуру выбора варианта. Ниже показан ее синтаксис на примере функции, которая централизирует процесс регистрации сообщений для сценария. Конкретные варианты описываются закрывающими скобками после каждого условия и двумя точками с запятой, завершающими блок операторов, который должен быть выполнен при реализации заданного условия. Оператор case завершается ключевым словом esac.

# Уровень протоколирования устанавливается в глобальной

# переменной LOG\_LEVEL. Возможные варианты перечислены в порядке

# от самого строгого до наименее строгого: Error, Warning, Info и Debug.

function logMsg {

message\_level-$l

message\_itself=$2

if [ $message\_level -le $LOG\_LEVEL ]; then

case $message\_level in

0) message\_level\_text="Error" ;;

1) message\_level\_text="Warning" ;;

2) message\_level\_text="Info" ;;

3) message\_level\_text="Debug" ;;

\*) message\_level\_text="Other"

esac

echo "${message\_level\_text}: $message\_itself"

fi

}

Эта функция иллюстрирует общепринятую парадигму уровня регистрации (log level), используемую многими приложениями административного характера. Код этого сценария позволяет генерировать сообщения на различных уровнях детализации, но действительно регистрируются или отрабатываются только те из них, которые проходят глобально устанавливаемый порог $LOG\_LEVEL. Чтобы прояснить важность каждого сообщения, его текст предваряется меткой, описывающей соответствующий уровень регистрации.

* 1. Циклы и конструкции

Конструкция for...in предназначена для упрощения выполнения некоторых действий для группы значений или файлов, особенно при универсализации файловых имен, т.е. замене реальных символов в имени и расширении универсальными (например, "\*" и "?") с целью формирования целых списков имен файлов. Шаблон \*.sh в приведенном ниже цикле forпозволяет обработать целый список совпадающих с ним (шаблоном) имен файлов из текущего каталога. Оператор for, проходя по этому списку, по очереди присваивает имя каждого файла переменной $file.

#!/bin/bash

suffix=BACKUP--`date +%Y%m%d-%H%M`

for script in \*.sh; do

newname="$script.$suffix"

echo "Copying $script to $newname..."

cp $script $newname

done

Результат выполнения этого сценария таков:

$ sh forexample

Copying rhel.sh to rhel.sh.BACKUP--20091210-1708...

Copying sles.sh to sles.sh.BACKUP--20091210-1708...

В раскрытии имени файла все работает в точном соответствии с тем, как написано в командной строке. Другими словами, сначала имя файла раскрывается (т.е. шаблон заменяется существующим именем), а затем уже обрабатывается интерпретатором в развернутом виде.

Аналогично можно ввести имена файла статически, как показано в командной строке:

for script in rosa.sh postinst.sh; do

В действительности любой список имен, содержащих пробельные символы (включая содержимое переменной), обрабатывается как объект циклической конструкции for...in.

Оболочке bash также присуща близость к циклу for из традиционных языков программирования, в которых задается стартовое выражение, инкрементация и условие окончания цикла.

for (( i=0 ; i < $CPU\_COUNT ; i++ )) ; do

CPU\_LIST="$CPU\_LIST $i"

done

На примере следующего сценария иллюстрируется bash-цикл while, который часто применяется для обработки аргументов командной строки и чтения строк файла:

#!/bin/bash

ехес 0<$1

counter=l

while read line; do

echo "$counter: $line"

$((counter++))

done

Результат выполнения этого сценария:

rosa$ sh whileexainple /etc/passwd

1: root:х:0:0:Superuser:/root:/bin/bash

2: bin:x:l:l:bin:/bin:/bin/bash

3: daemon:x:2:2:Daemon:/sbin:/bin/bash

В этом сценарии есть интересные особенности. Оператор ехеспереопределяет стандартный выходной поток сценария так, чтобы входные данные считывались из файлов, имена которых должны определяться в соответствии с первым аргументом командной строки. Если окажется, что такой файл не существует, данный сценарий сгенерирует ошибку.

Оператор readв конструкции while на самом деле является встроенным в оболочку, но действует подобно внешней команде. Внешние команды можно также помещать в конструкцию while. Цикл while в такой форме завершится при условии, если внешняя команда возвратит ненулевое значение кода завершения.

Конструкция $((counter++)) обозначает вычисление выражения. Кроме того, оно делает необязательным использование символа "$" для обозначения имен переменных. Удвоенный знак "++" – это оператор постинкремента, знакомый, например, по языку С. Он возвращает значение переменной, с которой он связан, но также увеличивает на 1 это значение.

Выражения $((...)) работают в контексте двойных кавычек, поэтому тело рассмотренного выше цикла можно свернуть до одной строки.

while read line; do

echo "$((counter++)) : $line"

done

* 1. Арифметические операции и массивы

Сложные структуры данных и вычисления нельзя отнести к сильной стороне оболочки bash, но применяются работа с массивами и арифметическими операциями.

Все bash-переменные представляют собой строковые значения, поэтому оболочка bash не делает различия в присваиваниях между числом 1 и символьной строкой "1". Различие лежит в использовании переменных. Следующий код иллюстрирует это различие:

#!/bin/bash

​

а=1

b=$((2))

​

с=$а+$b

d=$(($a+$b))

​

echo "$а + $b = $с \t(знак плюс как строковый литерал)"

echo "$а + $b = $d \t(знак плюс как арифметическое сложение)"

При выполнении этого сценария получится такой результат:

1 + 2 = 1+2 (знак плюс как строковый литерал)

1 + 2 = 3 (знак плюс как арифметическое сложение)

Следует обратить внимание на то, что знак "плюс" в присваивании переменной $с не работает как оператор конкатенации для строк. Он остается всего лишь литеральным символом, и посему эта строка эквивалентна следующей:

с="$а+$b"

Для того чтобы добиться вычисления, необходимо заключить выражение в двойные скобки $((...)), как показано выше в присваивании переменной $d. Но даже эта мера предосторожности не позволяет получить в переменной $d числового значения; это значение по-прежнему хранится в виде строки "3".

В оболочке bash реализован обычный ассортимент операторов: арифметических, логических и отношений (подробнее см. соответствующие man-страницы).

Массивы в командной оболочке bash могут показаться немного странными объектами и используются не очень часто, тем не менее при необходимости их можно применять. Литеральные массивы ограничиваются круглыми скобками, а отдельные элементы разделяются пробельными символами. Для включения литеральных пробелов в элемент можно использовать кавычки:

example=(aa 'bb сс' dd)

Для доступа к отдельным элементам массива используют выражение ${имя\_ массива [индекс]}. Индексация начинается с нуля. Такие индексы, как "\*" и "@", относятся к массиву в целом, а специальные конструкции ${#имя\_массива[ \* ] } и ${#имя\_массива[ @ ] } возвращают количество элементов в массиве. Не следует путать эти выражения с конструкцией ${#имя\_массива} хотя эта форма кажется более логичной, но в действительности она содержит указатель на первый элемент массива (эквивалент для ${#имя\_массива[0]}).

Можно подумать, что выражение $example[1] должно служить однозначной ссылкой на второй элемент массива, но оболочка bash интерпретирует эту строку так: $example (обозначение ссылки на $example[0]) плюс литеральная строка [1]. Отсюда вывод: ссылаясь на переменные массива, всегда используют фигурные скобки (без каких-либо исключений).

В качестве примера можно рассмотреть сценарий, который иллюстрирует некоторые особенности bash-мacсивов и подводные камни, на которые можно наткнуться при управлении ими.

#!/bin/bash

​

example=(aa 'bb сс' dd)

example[3]=ее

​

echo "example[@] = ${example[0]}"

echo "Массив example содержит ${#example[@]} элементов"

​

for elt in "${example[8] }"; do

echo " Элемент = $elt"

done

Вот как выглядит результат выполнения этого сценария.

$ sh arrays

example [@] = аа bb сс dd ее

Массив example содержит 4 элемента

Элемент = аа

Элемент = bb сс

Элемент = dd

Элемент = ee

Данный пример правильно реализован и по этому выглядит не сложным к выполнению. Однако, сценарий с заменой for-строки вариантом:

for elt in ${example[@]}; do

(т.е. без кавычек, в которых заключено выражение массива) также работает, но вместо четырех элементов он выведет пять: аа, bb, сс, dd и ее.

Важно помнить, что все bash-переменные все равно остаются строками, поэтому работа с массивами в некотором роде иллюзия.

Приложение А. Список доступных консольных команд

| Команда | Описание |
| --- | --- |
| A |  |
| **a2p** | Конвертирует awk-скрипт в Perl-скрипт. |
| **a2ps** | Форматирует текстовый файл для печати на Postscript-принтере. |
| **access** | Проверяет, имеет ли вызывающая программа доступ к указанному файлу. |
| **accton** | Используется администратором для включения/выключения ведения журнала действий пользователей (лог-файла). Регистрация активности пользователей сохраняется в заданном текстовом файле. |
| **aclocal** | Автоматическое создание файлов aclocal.m4 на основе содержимого файлов configure.in. |
| **acpi\_available** | Проверяет, доступна ли подсистема ACPI или нет. |
| **acpid** | Обеспечивает интеллектуальное управление питанием и используется для уведомления программ пользовательского пространства о событиях ACPI. |
| **acpi** | Отображение состояния батареи и другой информации подсистемы ACPI (сокр. от "Advanced Configuration and Power Interface"). |
| **ac** | Отображает статистику о времени подключения пользователей (в часах) на основе входов/выходов из Системы. |
| **addgroup** | Добавляет группу в Систему. |
| **addr2line** | Конвертирует адреса в имена файлов и номера строк. |
| **adduser** | Добавляет нового пользователя в Систему. |
| **agetty** | Linux-версия getty, которая представляет собой Unix-программу, работающую на главном компьютере и управляющую физическими или виртуальными терминалами для обеспечения многопользовательского доступа. |
| **agrep** | Ищет в файле записи, содержащие строки, которые точно или приблизительно соответствуют заданному шаблону. |
| **alias** | Создание или удаление псевдонима для команды или серии команд. |
| **alsactl** | Управление звуковым драйвером ALSA. |
| **amixer** | Консольный микшер звуковой карты, работающей под управлением ALSA (сокр. от "Advanced Linux Sound Architecture" – набор драйверов и утилит для поддержки звука в ОС). |
| **amp** | Проигрывает заданные MPEG аудио файлы, или (если вызывается с опцией -convert) распаковывает файл в wave-формат. |
| **aplay** | Консольный аудиоплеер для звуковых карт, работающих под управлением ALSA. |
| **apm** | Запрос к подсистеме расширенного управления питанием (Advanced Power Management APM) BIOS. |
| **apropos** | Помогает пользователю найти команду по ключевому слову, связанному с ней. Показывает краткое описание команд, в которых присутствует искомое слово. |
| **arch** | Отображает информацию об архитектуре компьютера. |
| **aria2c** | Сверхбыстрая утилита загрузки. |
| **arp** | Управление системным ARP-кешем (сокр. от "Address Resolution Protocol"). Также позволяет создать его полный дамп. |
| **ar** | Используется для создания, изменения и извлечения файлов из архивов. |
| **aspell** | Используется в качестве средства проверки орфографии. |
| **as** | Ассемблер проекта GNU; используется компилятором GCC. |
| **atd** | Демон планировщика заданий, запускающий задания, поставленные в очередь с помощью команды at. |
| **atq** | Отображает список отложенных заданий, запланированных пользователем. |
| **atrm** | Удаление указанных заданий, добавленных через команду at. Чтобы удалить выбранное задание, необходимо передать команде его номер. |
| **attr** | Расширенные атрибуты объектов xfs. |
| **at** | Выполняет команды в заданное время. |
| **autoconf** | Создание конфигурационных скриптов для пакетов с исходным кодом. |
| **autoheader** | Создание файла шаблона операторов #define или любого другого заголовка шаблона. |
| **automake** | Автоматическое создание makefile-ов (файлов, соответствующих стандартам кодирования GNU). |
| **autoreconf** | Создание автоматически собираемого исходного кода. |
| **autoupdate** | Обновление файла configure.in до более новой версии Autoconf. |
| **awk** | Язык поиска и обработки шаблонов. |
| B |  |
| **badblocks** | Поиск на диске сбойных блоков. |
| **banner** | Вывод большими буквами строки ASCII-символов в стандартный вывод. |
| **base64** | Base64 – кодирование/декодирование данных и стандартный вывод. |
| **basename** | Удаляет информацию о каталоге и суффиксы из имен файлов, т.е. Выводит имя файла с удалением всех компонентов каталога. |
| **bash** | Запуск командной оболочки bash (сокр. от "bourne again shell"). Данная оболочка является стандартным интерпретатором в большинстве Linux-дистрибутивов. |
| **batch** | Используется для считывания команд из стандартного ввода или указанного файла и их выполнения при разрешенном уровне нагрузки Системы, т.е. когда средняя нагрузка падает ниже 1.5. |
| **bc** | Консольный калькулятор. |
| **bdiff** | Как и команда diff, применяется для поиска отличий, но в больших файлах. |
| **bg** | Используется для восстановления на передний план задачи, выполняемой в фоновом режиме. |
| **biff** | Система почтовых уведомлений, которая оповещает пользователя в командной строке о появлении новых писем. |
| **bind** | Встроенная команда оболочки bash, которая используется для установки привязок клавиш и переменных Readline. |
| **bison** | Генератор парсера, схожий с yacc. |
| **break** | Используется для завершения выполнения циклов for, while и until. |
| **builtin** | Выполняет встроенную команду оболочки. Используется при замене встроенной команды оболочки некоторой функцией, но при этом нужна функциональность встроенной команды внутри самой функции. |
| **bye** | Аналог команды exit, используется для завершения сессии или выхода из терминала. |
| **bzcat** | Декомпрессия файлов в stdout. |
| **bzcmp** | Вызов утилиты cmp для сжатых файлов формата .bzip. |
| **bzdiff** | Сравнение сжатых файлов формата .bzip2. |
| **bzgrep** | Поиск шаблона или выражения, но внутри сжатого файла формата .bzip2. |
| **bzip2, bunzip2** | Сжатие и распаковка файлов. |
| **bzip2recover** | Восстановление поврежденных bzip2-файлов. |
| **bzless** | Похожая на команду bzmore, но имеет гораздо больше функций. Bzless не нужно читать весь входной файл перед запуском, поэтому с большим файлом она запускается быстрее, чем текстовые редакторы (например, vi). |
| **bzmore** | Это фильтр, который позволяет поэкранно просматривать как сжатые (.bzip2), так и простые текстовые файлы. |
| C |  |
| **calendar** | Служба напоминаний. |
| **cal** | Просмотр календаря определенного месяца или целого года. По умолчанию показывает календарь текущего месяца. |
| **cancel** | Остановка вывода информации о работе задачи. |
| **case** | Альтернатива нескольким операторам if/elif (когда используется одна переменная). |
| **cat** | Считывает данные из файла и выдает их содержимое в качестве выходных данных. Также способен объединять данные из нескольких файлов. |
| **ccrypt** | Инструмент командной строки для шифрования и дешифрования данных. |
| **cc** | Используется для компиляции кода языка С и создания исполняемых файлов. |
| **cd** | Изменение текущего рабочего каталога. |
| **cdrecord** | Запись компакт дисков (в том числе и музыкальных) из образа диска. |
| **c**fdisk | Утилита для просмотра и управления таблицей разделов диска. |
| **chage** | Утилита для просмотра и изменения информации об истечении срока действия пароля пользователя. |
| **chattr** | Изменение атрибутов файла в каталоге. |
| **chcon** | Изменение контекста безопасности файла. |
| **chdir** | Смена рабочей директории (аналог команды cd). |
| **checkeq** | Процессор языка программирования для описания уравнений и проведения сравнений. |
| **checknr** | Проверка nroff- и troff-файлов на ошибки. |
| **chfn** | Позволяет легко изменить имя пользователя и другие детали. |
| **chgrp** | Изменение группы, владеющей файлом или каталогом. |
| **chkconfig** | Перечисление текущей информации о запуске служб или какой-либо конкретной службы, а также обновления настроек уровня запуска службы. |
| **chmod** | Используется для управления разрешениями на заданный файл/каталог. |
| **chown** | Изменение владельца файла или группы. |
| **chpasswd** | Смена пароля одновременно для нескольких пользователей. |
| **ch**root | Изменение корневого каталога. |
| **chrt** | Управление атрибутами реального времени процесса. |
| **chsh** | Изменение оболочки входа пользователя (текущего shell). |
| **chvt** | Переключение между различными доступными терминалами TTY (сокр. от "Teletypewriter"). |
| **cksum** | Отображение и вычисление значения контрольной суммы файла или CRC (сокр. от "Cyclic Redundancy Check"), его размера в байтах и имени в стандартном выводе. |
| **clear** | Очистка экрана терминала. |
| **cmp** | Побайтовое сравнение двух файлов. Помогает выяснить, идентичны ли два сравниваемых файла или нет. |
| **colcrt** | Форматирование вывода текстового процессора таким образом, чтобы его можно было просматривать на дисплеях с ЭЛТ (сокр. от "Электронно-Лучевая Трубка"). |
| **colrm** | Удаляет выбранные столбцы из строк файла. Столбец определяется как один символ в строке. Входные данные считывается со стандартного входа. Результат записывается в стандартный вывод. |
| **column** | Форматирование отображения содержимого файла в виде столбцов. |
| **col** | Фильтрует переводы строк из входного потока. |
| **comm** | Сравнивает два отсортированных файла построчно и записывает в стандартный вывод: строки, которые являются общими, и строки, которые являются уникальными. |
| **compress** | Уменьшение размера файла. После сжатия файл приобретает расширение .z. |
| **continue** | Пропуск текущей итерации в циклах for, while и until. |
| **cpio** | Обработка архивных файлов, таких как \*.cpio или \*.tar. Данная команда может копировать файлы в архивы и из архивов. |
| **cpp** | Автоматически используется компилятором C для преобразования программы перед компиляцией. |
| **cp** | Копирование файлов и каталогов. |
| **crontab** | Список команд, которые требуется выполнять по регулярному расписанию, а также имя команды, используемой для управления этим списком. |
| **cron** | Утилита, автоматизирующая выполнение запланированной задачи в заданное время. |
| **csh** | Командный интерпретатор C Shell. |
| **csplit** | Разделение любого файла на множество частей. |
| **ctags** | Позволяет быстро получить доступ к файлам (например, быстро увидеть определение функции). |
| **cupsd** | Планировщик подсистемы вывода CUPS. |
| **curl** | Утилита для передачи данных на сервер или с сервера с использованием любого из поддерживаемых протоколов. |
| **cut** | Утилита для вывода заданной части строк каждого заданного файла. |
| **cu** | Отправка сигнала через терминал другой Системе. |
| **cvs** | Хранение истории изменений файла. При повреждении файла или других проблемах команда помогает вернуться к предыдущей версии и восстановить файл. |
| **D** |  |
| **date** | Отображение и установка системной даты и времени. |
| **dc** | Вычисление арифметических выражений. |
| **dd** | Утилита командной строки, основной целью которой является конвертирование и копирование файлов. |
| **declare** | Объявление переменных и функций, установка атрибутов и отображение их значений. |
| **delgroup** | Удаление группы из Системы. |
| **deluser** | Удаление пользователя из Системы. |
| **depmod** | Формирование списка зависимостей модулей ядра и генерация соответствующих \*.map-файлов. |
| **deroff** | Удаляет из файлов nroff- и troff-конструкции. |
| **df** | Отображение информации о доступном и использованном дисковом пространстве. |
| **dhclient** | Утилита для работы с DHCP-протоколом (получение динамического IP-адреса, настройка сетевых интерфейсов и пр.). |
| **diff3** | Построчное сравнения трех файлов. |
| **diff** | Отображение различий в файлах путем их построчного сравнения. |
| **dig** | Отображение информации о DNS. |
| **dircmp** | Сравнение содержимого двух каталогов. |
| **dircolors** | Настройка цветов для ls. |
| **dirname** | Удаление замыкающих слешей (/) из пути к каталогу и вывод оставшейся части. |
| **dirs** | Отображение списка сохраненных каталогов. |
| **dir** | Перечисление содержимого каталога. |
| **disable** | Деактивирует принтеры, отключая их от запросов на печать, отправляемых командой lp. |
| **dnsdomainname** | Вывести или установить системное имя хоста. |
| **dnsquery** | Послать запрос на сервер доменных имен через резолвер. |
| **dmesg** | Выводит сообщения ядра во время начальной загрузки ОС или настраивает их буфер. |
| **dmidecode** | Позволяет получить информацию об аппаратных компонентах Системы, а также другую полезную информацию: характеристики процессора, оперативной памяти (DIMM), детали BIOS и т.д. |
| **domainname** | Позволяет вывести или установить NIS/YP-доменное имя. |
| **dosfsck** | Диагностирует ФС MS-DOS на наличие проблем и пытается их устранить. |
| **dpost** | Конвертирование файлов из формата troff в postscript. |
| **dstat** | Обычно используется системными администраторами для получения информации о сетевых соединениях, устройствах ввода-вывода, процессоре и т.д. |
| **dumpe2fs** | Дамп информации ФС ext2/ext3. |
| **dumpkeys** | Выводит информацию о текущей раскладке клавиатуры. |
| **dump** | Резервное копирование ФС на какое-либо запоминающее устройство. |
| **du** | Отслеживание файлов и каталогов, которые занимают чрезмерное количество места на жестком диске. |
| **djview** | Полноценная программа для просмотра файлов djvu. |
| **E** |  |
| **e2fsck** | Проверка ФС ext2/ext3/ext4. |
| **echo** | Отображение текста/строки на экране, которые передаются в качестве аргумента. |
| **edit** | Текстовый редактор (разновидность редактора для простых пользователей). |
| **ed** | Запуск построчного текстового редактора с минималистичным интерфейсом. |
| **egrep** | Обрабатывает шаблон как расширенное регулярное выражение и выводит строки, соответствующие шаблону. |
| **eject** | Позволяет извлекать съемный носитель (обычно CD-ROM, дискету, ленту, JAZ- или ZIP-диск) с помощью ПО. |
| **elm** | Интерактивная почтовая система. |
| **emacs** | Редактор с простым пользовательским интерфейсом, в котором нет режима вставки. У него есть только режим редактирования. |
| **enable** | Включение/выключение lp-принтеров. |
| **env** | Вывод информации о переменных окружения. Также используется для запуска утилиты или команды в пользовательском окружении. |
| **eqn** | Используется для описания сравнений. |
| **ethtool** | Отображает или позволяет изменить настройки сетевой карты. |
| **eval** | Команда воспринимает переданные ей аргументы как директивы оболочки. |
| **evim** | Простой Vim, позволяет редактировать файл с помощью Vim в незаданном режиме. |
| **exec** | Используется для выполнения команды из самого bash. |
| **exiftool** | Чтение и запись метаданных файлов. |
| **exit** | Закрывает командный интерпретатор с состоянием N. Если N не указано, то состоянием выхода будет состояние последней выполненной команды. |
| **expand** | Позволяет конвертировать табуляции в пробелы в файле, а когда файл не указан, то данные считываются со стандартного ввода. |
| **expect** | Команда, работающая со скриптами, которые ожидают ввода данных от пользователя. Автоматизирует задачу, предоставляя входные данные. |
| **expiry** | Проверяет и изменяет пароль согласно политике устаревания. |
| **export** | Помечает переменные окружения, которые экспортируются в дочерние процессы. |
| **expr** | Вычисляет заданное выражение и отображает результат. |
| **ex** | Текстовый редактор в ОС, который также называется линейным режимом редактора vi. |
| **F** |  |
| **false** | Ничего не делает; сообщает о безуспешном выполнении. |
| **factor** | Вывод простых множителей заданных чисел (задаваемых как через командную строку, так и через стандартный ввод). |
| **fc-cache** | Сканирует каталоги шрифтов (и создает их кеш), которые используют fontconfig для обработки шрифтов. |
| **fc-list** | Используется для перечисления доступных шрифтов и стилей шрифтов. Список всех шрифтов можно отфильтровать и отсортировать, применив соответствующую опцию форматирования. |
| **fc** | Используется для перечисления, редактирования или повторного выполнения команд, ранее введенных в интерактивную оболочку. |
| **fdisk** | Диалоговая команда в ОС, используемая для создания и управления таблицей разделов диска. |
| **fetchmail** | Прием почты с серверов POP, IMAP, ETRN, или ODMR. |
| **ffmpeg** | Видеоконвертер. |
| **f**grep | Поиск строк в файле. |
| **fg** | Перемещение фоновой задачи на передний план. |
| **file** | Вывод типа файла. |
| **findsmb** | Список всех машин, доступных по SMB-протоколу (сокр. от "Server Message Block"). |
| **find** | Поиск файлов и каталогов. |
| **finger** | Подробная информация обо всех пользователях, вошедших в Систему. |
| **fmt** | Утилита простого форматирования текста. |
| **fold** | Оборачивает каждую строку во входной файл, чтобы соответствовать заданной ширине, и выводит её на стандартный вывод. |
| **foreach** | Выполняет набор команд для каждого из элементов заданного массива. |
| **for** | Используется для многократного выполнения набора команд для каждого элемента, присутствующего в списке. |
| **free** | Отображение объема свободной и используемой памяти в Системе. |
| **fsck.fat** | Проверка и восстановление ФС MS-DOS. |
| **fsck** | Проверка и восстановление ФС Linux. |
| **ftp** | Интерактивная утилита для доступа к FTP (сокр. от "File Transfer Protocol"). |
| **function** | Создание функций или методов. |
| **fun** | Рисование в терминале узоров различного типа. |
| **fusermount** | Монтирование и демонтирование в ФС FUSE. |
| **fuser** | Определение процессов, использующих файлы или сокеты. |
| **G** |  |
| **gawk** | GNU-версия awk. |
| **gcc** | Используется для компиляции программ, написанных на языках C, C++, Objective-C и Objective-C++. |
| **gdb** | Мощный отладчик для программ, написанных на C, C++, Ada, Fortran и пр. |
| **ginstall** | Копирование файлов и установка атрибутов. |
| **gendiff** | Утилита, помогающая создать diff-файл без ошибок. |
| **getent** | Получение элементов из базы данных. |
| **getfacl** | Получение списков контроля доступа к файлам. |
| **getfattr** | Получить расширенные атрибуты объектов ФС. |
| **git** | Система управления версиями. |
| **glxinfo** | Показать информацию о реализации GLX. |
| **gpasswd** | Администрирование файлов /etc/group и /etc/shadow. |
| **gpg** | Инструмент для шифрования и цифровой подписи. |
| **gpgv** | Программа проверки подписей. |
| **gprof** | Отображение данных по профилированию программы. |
| **grep** | Поиск в файле определенного шаблона символов и вывод всех строк, содержащих этот шаблон. |
| **grodvi** | Преобразует вывод groff в форматы tex и dvi. |
| **groff** | Frontend-приложение для системы форматирования документов groff. |
| **groupadd** | Создание новой группы. |
| **groupdel** | Удаление существующей группы. |
| **groupmod** | Модификация или смена существующей группы. |
| **groups** | Отображение списка групп, в которых состоит пользователь. |
| **grpck** | Проверка целостности информации о группах, а именно: все записи в /etc/group и /etc/gshadow имеют правильный формат и содержат допустимые данные. |
| **grpconv** | Преобразует пароли пользователей и групп в/из защищенную форму. |
| **gs** | Команда вызывает Ghostscript-интерпретатор языка Adobe Systems postscript и формата PDF (сокр. от "Portable Document Format"). |
| **gunzip** | Сжатие или распаковка файлов. |
| **gview** | Версия редактора vi с графическим интерфейсом. Запускается в новом окне. |
| **gvim** | Синоним для команды gview. |
| **gzexe** | Сжатие исполняемых файлов, а также их автоматическое распаковывание в момент выполнения. |
| **Gzip** | Сжатие файлов. |
| **g++** | Используется для предварительной обработки, компиляции, сборки и линкинга исходного кода при создании исполняемого файла. |
| **H** |  |
| **halt** | Указание аппаратному обеспечению компьютера остановить все исполняющиеся в данный момент процессы. Основное применение перезагрузка или выключение Системы. |
| **hash** | Доступ к хеш-таблице недавно выполненных программ. |
| **hdparm** | Получение информации о жестком диске, изменении интервалов записи, настроек DMA (сокр. от "Direct Memory Access"). |
| **head** | Выводит N первых строк файла. |
| **help** | Отображает информацию о встроенных командах оболочки. |
| **hexdump** | Фильтр, который отображает указанные файлы или стандартный ввод, если файлы не указаны, в заданном пользователем формате. |
| **history** | Отображает историю команд, которые были введены с момента начала сессии. |
| **hostid** | Отображает числовой идентификатор текущего хоста. |
| **hostnamectl** | Может использоваться для запроса и изменения имени хоста Системы и связанных с ним параметров. |
| **hostname** | Выдает или устанавливает имя компьютера. |
| **host** | Утилита для работы с DNS-запросами. |
| **htop** | Консольная утилита, которая позволяет пользователю интерактивно (в режиме реального времени) отслеживать список запущенных процессов. |
| **hwclock** | Запрос и установка аппаратных часов (сокр. "rtc" от "real-time clock"). |
| **I** |  |
| **iconv** | Преобразование некоторого текста из одной кодировки в другую. |
| **id** | Выводит информацию о текущем пользователе и группе. |
| **ifconfig** | Конфигурирование сетевого интерфейса. |
| **ifdown** | Отключает сетевой интерфейс. |
| **ifquery** | Позволяет выбрать информацию о сетевом интерфейсе. |
| **iftop** | Инструмент анализа сети, используемый системными администраторами для просмотра статистики, связанной с пропускной способностью канала передачи данных. |
| **ifup** | Задействует (поднимает) сетевой интерфейс, позволяя ему передавать и получать данные. |
| **imagemagick** | Набор программ для чтения и редактирования файлов множества графических форматов. |
| **import** | Создание скриншота экрана (всего экрана или только его части) с последующим сохранением в файл. |
| **info** | Позволяет читать документацию в формате info. |
| **insmod** | Программа для активации модулей ядра. |
| **install** | Копирует файлы и устанавливает атрибуты. |
| **inxi** | Показывает информацию о Системе. |
| **iostat** | Мониторинг статистики I/O-операций для устройств и разделов. |
| **iotop** | Отображение статистики работы процессов с дисками. |
| **ipcrm** | Удаление IPC-ресурсов (сокр. от "Inter-Process Communication") и связанных с ними данных. |
| **ipcs** | Вывод информации об IPC-ресурсах. |
| **iptables-save** | Сохраняет текущие правила iptables в указанном файле. |
| **iptables** | Настройка правил брандмауэра, входящего в состав ядра ОС. |
| **ip** | Отображение и управление сетевыми подключениями, маршрутизацией и пр. |
| **iwconfig** | Отображение параметров и статистики беспроводной связи, которые извлекаются из /proc/net/wireless. |
| **iwlist** | Используется для отображения дополнительной информации от беспроводного сетевого интерфейса, не показываемой iwconfig. |
| **J** |  |
| **jobs** | Отображение состояния заданий в текущей сессии. |
| **join** | Утилита для соединения строк двух файлов на основе ключевого поля, присутствующего в обоих файлах. |
| **journalctl** | Используется для просмотра логов, собранных systemd. Systemd собирает логи в бинарном формате. |
| **K** |  |
| **kernelversion** | Выводит основную версию ядра. |
| **killall** | Посылает сигнал kill всем активным процессам. |
| **kill** | Посылает процессу сигнал завершения работы. |
| **ksh** | Командный интерпретатор (оболочка) Korn Shell. |
| **L** |  |
| **last** | Отображение списка всех пользователей, вошедших и вышедших из Системы с момента создания файла /var/log/wtmp. |
| **lastb** | Отображение списка последних вошедших в Систему пользователей. |
| **ldd** | Отображение зависимостей разделяемых библиотек. |
| **ld** | Редактор ссылок на библиотеки для объектов. |
| **less** | Поэкранное чтение содержимого текстового файла. |
| **let** | Вычисление арифметических выражений для переменных оболочки. |
| **lexgrog** | Анализирует заголовочную информацию справочных страниц. |
| **link** | Создание жесткой ссылки на файл. |
| **ln** | Создание символьных ссылок между файлами. |
| **locate** | Поиск файлов по имени. |
| **login** | Вход в Систему. |
| **logname** | Отображает логин пользователя. |
| **logout** | Аналог lo. |
| **look** | Выводит строки, начинающиеся с заданной подстроки. |
| **losetup** | Настройка и управление виртуальными loop-устройствами. |
| **lo** | Завершает работу с командной оболочкой. |
| **locale** | Выдаёт информацию о локали. |
| **localedef** | Компилирует файлы определения локали. |
| **lpq** | Просмотр очереди печати. |
| **lpr** | Локальная печать. |
| **lprm** | Команда удаления заданий из очереди. |
| **lsblk** | Отображение сведений о блочных устройствах. |
| **lsb\_release -a** | Отображение информации о дистрибутиве. |
| **lshw** | Генерация подробной информации об аппаратной конфигурации Системы на основе файлов из каталога /proc. |
| **lsmod** | Отображение модулей ядра ОС, которые в данный момент загружены. |
| **lsof** | Отображает информацию о том, какие файлы используются тем или иным процессом. |
| **lspci** | Отображает список всех устройств на шине PCI. |
| **lsusb** | Отображение информации о USB-шинах и устройствах, подключенных к ним. |
| **ls** | Отображение списка содержимого каталога. |
| **lsattr** | Выводит атрибуты файла второй расширенной ФС Linux. |
| **lynx** | Браузер. |
| **lzcat** | Посмотреть содержимое файла, сжатого LZMA (сокр. от "Lempel-Ziv-Markov chain-Algorithm"). |
| **Lzma** | Сжать или распаковать файл по алгоритму LZMA. |
| **lzop** | Сжимает или разжимает файлы. |
| **M** |  |
| **mach** | Отображение информации о типе процессора. |
| **mailq** | Сводка почтовых сообщений, поставленных в очередь для дальнейшей доставки. |
| **mailx** | Интерактивная система обработки сообщений электронной почты. |
| **make-ssh-known-hosts** | Создание файла ssh\_known\_hosts из данных DNS. |
| **make** | Автоматическое определение частей большой программы, которые должны быть перекомпилированы, и выдача команды для их перекомпиляции. |
| **manconv** | Изменяет кодировку справочной страницы. |
| **manpath** | Определяет путь поиска справочных страниц. |
| **man-recode** | Изменяет кодировку справочных страниц. |
| **man** | Отображение руководства пользователя по любой команде, которую можно запустить в терминале. |
| **mc** | Файловый менеджер. |
| **md5sum** | Проверка целостности данных с помощью алгоритма хеширования MD5. |
| **memusagestat** | Генерирует график по данным профилирования памяти. |
| **memusage** | Исследует использование памяти программы. |
| **merge** | Слияние содержимого трех файлов. |
| **mesg** | Отправка сообщений в другой терминал. |
| **mkdir** | Создание одного или сразу нескольких каталогов. |
| **mkfs** | Создание ФС в выбранном разделе. |
| **mknod** | Создает специальные блочные или символьные файлы. |
| **mkswap** | Создание файла (или раздела) подкачки (swap). |
| **mkvextract** | Извлечение дорожки из файлов Matroska(TM) в другие файлы. |
| **mkvinfo** | Запись информации об элементах в файлы Matroska(TM). |
| **mkvmerge** | Упаковка потоков мультимедиа в файл Matroska(TM). |
| **mkvpropedit** | Изменяет свойства существующих файлов Matroska(TM) без сложного ремультиплексирования. |
| **mkvtoolnix-gui** | Графический интерфейс для mkvmerge(1), включающий редакторы глав и заголовков. |
| **modeprobe** | Программа для загрузки и выгрузки модулей из ядра ОС. |
| **modinfo** | Отображение информации о модуле ядра ОС. |
| **more** | Поэкранный просмотр текстовых файлов в командной строке. |
| **mount** | Примонтировать ФС в директорию. |
| **mpstat** | Создание статистического отчета о работе процессора (или по каждому процессору, если их несколько). |
| **mtrace** | Трактует журнал трассировки malloc. |
| **mt** | Управление работой накопителя с магнитной лентой. |
| **mv** | Перемещение внутри файловой системы одного или нескольких файлов, или каталогов из одного места в другое. |
| **mysqldump** | Утилита для создания резервной копии базы данных mysql. |
| **mysql** | Программа для управления базой данных mysql. |
| **N** |  |
| **nc** | Мощная утилита, которая используется для решения различных задач, связанных с сетевыми TCP- и UDP-подключениями. |
| **nethogs** | Группирует пропускную способность для каждого процесса. |
| **netstat** | Отображает различную информацию, связанную с сетью (например, сетевые соединения, таблицы маршрутизации, статистика интерфейса и пр.). |
| **networkctl** | Запрос состояния сетевых ссылок. |
| **newgrp** | Предоставляет пользователю права новой группы на время. |
| **newsbeuter** | Программа чтения RSS/Atom для текстовых терминалов. |
| **nice** | Запуск программы с измененным приоритетом. |
| **niscat** | Отображение таблиц и объектов NIS+. |
| **nischmod** | Изменение прав доступа к объекту NIS+. |
| **nischown** | Изменение владельца объекта NIS+. |
| **nischttl** | Изменение значения времени жизни объекта NIS+. |
| **nisdefaults** | Отображение заданных по умолчанию значений NIS+. |
| **nistbladm** | Команда администрирования таблицы NIS+. |
| **nisdomainname** | Вывести или установить системное имя хоста. |
| **nl** | Печатает каждый ФАЙЛ на стандартный вывод, добавляя номера строк. |
| **nmap** | Инструмент исследования сети и сканер безопасности/портов. |
| **nmcli** | Управление networkmanager. Также может применяться для отображения состояния сетевого устройства, создания, редактирования, активации/деактивации и удаления сетевых подключений. |
| **nohup** | Продолжить выполнение команды, когда сессия терминала будет завершена. |
| **nroff** | Приложение для форматирования документов. |
| **nslookup** | Инструмент сетевого администрирования, связанного с DNS. |
| **O** |  |
| **od** | Преобразование содержимого входных данных в восьмеричный и другие форматы. |
| **onintr** | Отображение информации об аппаратных прерываниях. |
| **on** | Выполнение команды на удаленной системе, но с локальным окружением. |
| **osec** | Выполняет контроль целостности Системы. |
| **P** |  |
| **pack** | Сжатие файлов по алгоритму Хаффмана. |
| **pact** | Вывод содержимого сжатого текстового файла. |
| **pagesize** | Отображение размера страницы в памяти. |
| **parted** | Программа для разметки диска. |
| **partprobe** | Информирование ОС об изменениях в таблице разделов. |
| **passwd** | Изменение паролей учетных записей пользователей. |
| **paste** | Соединение файлов (параллельным слиянием) путем вывода результирующих строк, состоящих из строк каждого указанного файла, разделенных табуляцией в качестве разделителя, в стандартный вывод. |
| **pathchk** | Проверить правильность и портируемость имени файла. |
| **pax** | Чтение и запись файловых архивов и копирование иерархий каталогов. |
| **pdftk** | Простой редактор pdf-файлов. |
| **perl** | Интерпретатор Perl-скриптов. |
| **pgrep** | Позволяет находить идентификаторы процессов работающей программы на основе заданных критериев. |
| **pg** | Фильтр для постраничного просмотра содержимого текстовых файлов. |
| **pico** | Простой текстовый редактор в стиле Pine Composer. |
| **pidof** | Определение идентификаторов процессов конкретной запущенной программы. |
| **pine** | Программа для Интернет-новостей и электронной почты. |
| **ping** | Проверка сетевого подключения между хостом и сервером/другим хостом. |
| **pinky** | Команда поиска информации о пользователе, которая дает подробную информацию обо всех пользователях, вошедших в Систему. В отличие от finger можно обрезать интересующую вас информацию. |
| **pkexec** | Выполнить команду от имени другого пользователя. |
| **pkill** | Завершить процесс по его имени. |
| **pmap** | Отображение информации об адресном пространстве процесса. |
| **postconf** | Конфигурационный инструментарий Postfix. |
| **postdrop** | Агент отправки почты Postfix. |
| **postfix** | Программа управления почтовой системой Postfix. |
| **postkick** | Подстегнуть службу Postfix. |
| **postlock** | Блокирует почтовую папку и выполняет команду. |
| **postlog** | Postfix-совместимый инструментарий журналирования. |
| **postsuper** | Суперинтендант Postfix. |
| **poweroff** | Выключение Системы. |
| **preparetips5** | Извлечение текста из файлов с полезными советами. |
| **printenv** | Отображение переменных окружения. |
| **printf** | Отображение отформатированных данных (числа, строки и пр.). |
| **procmail** | Автономный обработчик почты. |
| **prtstat** | Выводит статистику по процессу. |
| **pr** | Преобразование текстовых файлов для вывода. |
| **pstree** | Отображение дерева процессов. |
| **ps** | Вывод информации об активных процессах. |
| **ptx** | Составляет оглавление файла. |
| **pvs** | Производит форматированный вывод информации о физических томах. |
| **pwck** | Проверяет целостность файлов паролей. |
| **pwd** | Отображает полный путь к текущему рабочему каталогу. |
| **Q** |  |
| **qrencode** | Создает QR-коды и сохраняет в виде картинки на диск. |
| **quit** | Завершение сеанса командной оболочки. |
| **R** |  |
| **ranlib** | Генерирует индекс к содержимому архива и сохраняет его в архиве. |
| **rb** | Получение файлов по протоколам XMODEM, YMODEM, ZMODEM. |
| **rcp** | Копирование файлов с одного компьютера на другой. |
| **readelf** | Получение информации о файлах формата ELF (сокр. от "Executable and Linkable Format"). |
| **readlink** | Отображение значения символьной ссылки. |
| **read** | Считывание строки со стандартного ввода. |
| **reboot** | Перезагрузка Системы. |
| **redshift** | Изменяет цветовой профиль экрана в зависимости от времени суток. |
| **red** | Запуск редактора ed в режиме прокрутки текста. |
| **rename** | Переименование файлов. |
| **repeat** | Повторять выполнение команды нужное количество раз. |
| **replace** | Утилита замены строк в файлах. |
| **reset** | Инициализация терминала. Полезно в тех ситуациях, когда завершившаяся программа оставила терминал в "ненормальном" состоянии. |
| **restore** | Восстановление файлов из резервной копии, созданной с помощью dump. |
| **rev** | Реверсивное изменение строк файла. |
| **rlogin** | Удаленный вход в Систему. |
| **rmdir** | Удаление пустых каталогов. |
| **rmmod** | Удаление модуля из ядра. |
| **rm** | Удаление файлов, каталогов, символьных ссылок и пр. |
| **route** | Используется для работы с IP-адресами и таблицей маршрутизации. |
| **rpcgen** | Компилятор протокола RPC. |
| **rpcinfo** | Отображает информацию об RPC (сокр. от "Remote Procedure Call"). |
| **rpm** | Менеджер пакетов RPM. |
| **rsh** | Удаленный командный интерпретатор. |
| **rsync** | Позволяет синхронизировать файлы и каталоги между двумя местоположениями. Ведет себя почти так же, как rcp, но имеет гораздо больше опций и использует протокол удаленного обновления для значительного ускорения передачи файлов при обновлении целевого файла. |
| **runcon** | Запуск команды с указанным контекстом безопасности. |
| **rview** | Улучшенный Vi, текстовый редактор для программистов. |
| **rvim** | Улучшенный Vi, текстовый редактор для программистов. |
| **rx** | Получение файлов по протоколам XMODEM, YMODEM, ZMODEM. |
| **rz** | Получение файлов по протоколам XMODEM, YMODEM, ZMODEM. |
| **S** |  |
| **s2p** | Конвертер sed-скриптов в Perl. |
| **sar** | Мониторинг ресурсов ОС, таких как загрузка процессора, использование памяти, нагрузка на устройства ввода-вывода и т.д. |
| **scp** | Безопасное копирование файлов между серверами. |
| **screen** | Обеспечивает возможность запуска и использования нескольких сеансов оболочки из одной ssh-сессии. |
| **scriptreplay** | Воспроизведение активности терминала, сохраненной с помощью команды script. |
| **script** | Используется для записи всего, что происходит в терминале. |
| **sdiff** | Находит различия между двумя файлами и объединяет их в интерактивном режиме. |
| **sed** | Потоковый редактор для фильтрации и преобразования текста. |
| **select, pselect** | Создание нумерованного меню, из которого пользователь может выбрать нужный параметр. |
| **sendmail** | Отправка электронного письма. |
| **sensors** | Мониторинг температуры и напряжения. |
| **seq** | Генерирует последовательность чисел с заданным шагом. |
| **service** | Утилита для запуска и управления службами. |
| **sestatus** | Проверка того, включен ли selinux (модуль безопасности ОС). |
| **setfacl** | Настройка списков контроля доступа к файлам. |
| **setsid** | Запуск программы в новом сеансе. |
| **set** | Показывает весь набор переменных в окружении. |
| **s**fdisk | Программа для разметки дисков. |
| **sftp** | Клиент для передачи файлов по протоколу SFTP. |
| **shift** | Сдвигает/перемещает аргументы командной строки на одну позицию влево. |
| **showkey** | Отображает код каждой кнопки, нажимаемой на клавиатуре. |
| **shred** | Переписывает несколько раз указанные файлы для того, чтобы сделать более трудоёмким процесс восстановления данных даже в случае использования специального оборудования для восстановления. |
| **shuf** | Перемешивание строк в файле. |
| **shutdown** | Безопасное выключение Системы. |
| **sh** | Командная оболочка Bourne Shell. |
| **sleep** | Задержка на заданное количество времени. |
| **slogin** | Псевдоним ssh-клиента, применяемый для удаленного вход в Систему через ssh. |
| **smartctl** | Утилита управления и мониторинга для SMART-дисков. |
| **smbclient** | Ftp-подобный клиент для доступа к ресурсам SMB/CIFS. |
| **sort** | Сортировка строк в текстовых файлах. |
| **source** | Чтение и выполнение содержимого файла (обычно набора команд), передаваемого в качестве аргумента в текущем скрипте оболочки. |
| **spell** | Проверка орфографии. |
| **split** | Разделение больших файлов на более мелкие. |
| **sha1sum** | Вычисляет и сверяет криптографические контрольные суммы по алгоритму SHA1 (устарел). |
| **sshfs** | Позволяет смонтировать файловую систему удалённого сервера с помощью протокола ssh. |
| **ssh** | Протокол, используемый для безопасного подключения к удаленному серверу/Системе. |
| **ssh-add** | Добавляет RSA- или DSA-идентификацию в аутентификационного агента. |
| **ssh-agent** | Агент аутентификации. |
| **ssh-copy-id** | Инсталлирует ваши файлы публичной идентификации на удаленной машине. |
| **ssh-keygen** | Создание, обслуживание и преобразование аутентификационных ключей. |
| **ssh-keyscan** | Собирает публичные ключи ssh. |
| **ss** | Просмотр информации о сетевых подключениях. |
| **startx** | Инициализация сессии X-сервера. |
| **stat** | Отображение статистики файла или ФС. |
| **stop** | Остановка фоновой задачи. |
| **strace** | Один из самых мощных инструментов мониторинга и диагностики, позволяющий проводить трассировку системных вызовов и сигналов. |
| **strip** | Позволяет удалить отладочную информацию из исполняемых файлов. |
| **stty** | Отображает и устанавливает параметры терминала. |
| **sudo** | Используется в качестве префикса некоторой команды, которую разрешено запускать только суперпользователю. |
| **sum** | Вычисление контрольной суммы и подсчета блоков в файле. |
| **su** | Авторизация от имени другого пользователя. |
| **swapoff** | Отключает файл (или раздел) подкачки. |
| **swapon** | Задействует файл (или раздел) подкачки. |
| **sync** | Синхронизация данных на диске с данными в памяти. |
| **systemctl** | Используется для проверки и контроля состояния systemd и управления службами. |
| **sx** | Передача файлов по протоколам XMODEM, YMODEM, ZMODEM. |
| **sz** | Передача файлов по протоколам XMODEM, YMODEM, ZMODEM. |
| **T** |  |
| **tabs** | Останавливает работу вкладок в терминале. |
| **tac** | Объединение и вывод файлов в обратном порядке. |
| **tail** | Выводит N последних строк файлов. |
| **talk** | Программа визуальной коммуникации, которая копирует строки с терминала в терминал другого пользователя. |
| **tar** | Используется для создания архива и извлечения архивированных файлов. |
| **tcopy** | Копирование магнитной ленты. |
| **tcpdump** | Консольный анализатор сетевого траффика. |
| **tcsh** | Командный интерпретатор TENEX C Shell. |
| **tee** | Считывает данные из стандартного устройства ввода и записывает их на стандартное устройство вывода или в файл. |
| **telnet** | Используется для связи с другим хостом по протоколу TELNET (сокр. от "Teletype Network"). |
| **terminfo** | База данных о возможностях терминала. |
| **test** | Проверяет типы файлов. |
| **timex** | Время выполнения команды (с отображением более подробной информации). |
| **time** | Выполняет команду и по её завершению отображает статистику о затраченных временных ресурсах на её исполнение. |
| **todos** | Конвертирование текстовых файлов Unix в формат DOS. |
| **top** | Отображение и обновление информации о топе процессов CPU. |
| **touch** | Используется для создания, изменения и модификации временных меток файла. |
| **tput, reset** | Инициализирует терминал или запрашивает базу данных terminfo. |
| **tracepath** | Отображает маршрут по сетевым узлам с MTU (сокр. от "Maximum Transmission Unit"). |
| **traceroute** | Отображает маршрут, по которому пакет достигает хоста. |
| **tree** | Список содержимого каталогов в древовидном формате. |
| **truncate** | Удаляет все содержимое файла. Она не удаляет файл, а оставляет его как файл с нулевым байтом на диске. Если воспользоваться командой усечения, права доступа к файлу и право собственности на него сохранятся. |
| **tr** | Преобразование или удаление символов. |
| **true** | Ничего не делает, всегда завершается успешно. |
| **tsort** | Выполняет топологическую сортировку. |
| **tty** | Отображает информацию, связанную с терминалом (имя файла терминала, подключенного к стандартному входу). |
| **type** | Используется для описания того, как будет интерпретироваться каждый аргумент, если он будет использоваться в качестве имени команды. |
| **U** |  |
| **umask** | Устанавливает маску прав для режима создания файла. |
| **umount** | Позволяет размонтировать ФС. |
| **unalias** | Удаление псевдонима. |
| **uname** | Отображает информацию о Системе. |
| **uncompress** | Распаковка сжатого файла. |
| **unexpand** | Преобразует каждый пробел в табуляцию, записывая результат в стандартный вывод. |
| **uniq** | Отображает или пропускает повторяющиеся строки. |
| **unix2dos** | Преобразует текстовый файл Unix в формат DOS. |
| **unlink** | Удаление ссылки на файл. |
| **unlzma** | Извлечение файлов из архива, сжатого командой Lzma. |
| **unpack** | Извлечение файлов из архива, сжатого командой pack. |
| **unxz** | Распаковка .xz-архива. |
| **unzip** | Распаковка .zip-архива. |
| **uptime** | Выводит информацию о времени работы Системы. |
| **useradd** | Добавление учетных записей пользователей в Систему. |
| **userdel** | Удаление учетной записи пользователя и связанных с ней файлов. |
| **usermod** | Изменяет учётную запись пользователя. |
| **username** | Получение имени пользователя. |
| **users** | Отображение имен пользователей, вошедших в Систему на данный момент. |
| **V** |  |
| **vacation** | Автоответчик для электронной почты. |
| **vdir** | Отображение списка содержимого каталога. |
| **vi** | Текстовый редактор. |
| **vim** | Свободный текстовый редактор, созданный на основе более старого vi. |
| **vimdiff** | Позволяет редактировать две или три версии файла с помощью Vim с отображением различий. |
| **vimtutor** | Учебник по Vim. |
| **vmstat** | Команда мониторинга производительности Системы, предоставляющая информацию о процессах, памяти, файле подкачке, активности процессора и т.д. |
| **vlock** | Программа для блокирования виртуальных консолей. |
| **vnstat** | Используется системными администраторами для мониторинга параметров сети: загруженность каналов, входящий/исходящий трафик и пр. |
| **W** |  |
| **wall** | Отображает в терминале сообщение для всех пользователей, вошедших в Систему. |
| **watch** | Используется для запуска и слежки за программой, отображая её вывод в терминал. |
| **wc** | Определение количества строк, слов, байтов и символов в файлах. |
| **wget** | Утилита для загрузки файлов из Интернета. |
| **whatis** | Получение однострочного описания искомой команды. |
| **whereis** | Отображает полный путь к исполняемому файлу программы (или к исходным файлам, если они присутствуют в Системе). |
| **which** | Поиск исполняемого файла, связанного с заданной командой, путем нахождения его в переменной окружения PATH. |
| **whoami** | Отображает имя текущего пользователя. |
| **whois** | Отображение доступной информации об Интернет-ресурсе (например, о сайте). |
| **who** | Сообщает о всех зарегистрировавшихся в Системе пользователях. |
| **wipe** | Безопасное стирание информации с магнитных носителей |
| **write** | Позволяет пользователю коммуницировать с другими пользователями, копируя строки с одного терминала в другие. |
| **w** | Отображает пользователей, вошедших в Систему и их процессы. |
| **X** |  |
| **xargs** | Преобразует входные данные, полученные от стандартного ввода, в аргументы команды. |
| **xdg-open** | Используется для открытия файла или URL-адреса в предпочтительном приложении пользователя. |
| **xfd** | Отображение всех символов шрифта X-сервера. |
| **xinit** | Программа для инициализации системы X Windows. |
| **xhost** | Настройка прав доступа к X-серверу. |
| **xkill** | Отключить клиента от его X-ресурса. |
| **xlsfonts** | Отображение всех шрифтов X-сервера. |
| **xmls**tar**let** | Командная строка инструментария XML/XSLT. |
| **xmodmap** | Утилита для изменения раскладки клавиш клавиатуры и мыши в Xorg. |
| **xrandr** | Настройка режимов работы с дисплеями (мониторами). |
| **xrdb** | Управление базой данных ресурсов X-сервера. |
| **xset** | Изменение значения переменной X-сервера. |
| **xzcat** | Просмотр содержимого текстового файла, сжатого командой xz. |
| **xz** | Сжатие файла в .xz-формат. |
| **xxd** | Создаёт представление файла в виде шестнадцатеричных кодов или выполняет обратное преобразование. |
| **Y** |  |
| **yacc** | Утилита, служащая стандартным генератором синтаксических анализаторов (парсеров) в Unix-системах. |
| **yes** | Выводит строку несколько раз до тех пор, пока команде не будет послан сигнал kill. |
| **ypdomainname** | Вывести или установить системное имя хоста. |
| **yppasswd** | Изменение пароля базы данных NIS. |
| **Z** |  |
| **zcat** | Отображение содержимого файла, сжатого командой zip. |
| **zdiff** | Вызывает diff для файлов, сжатых командой Gzip. |
| **zdump** | Получение информации о часовом поясе. |
| **z**for**ce** | Присвоить расширение .gz всем архивам Gzip. |
| **z**grep | Поиск выражений в заданном файле, даже если он сжат. |
| **zipcloack** | Зашифровать .zip-файл. |
| **zipinfo** | Вывести информацию о .zip-файле. |
| **zipnote** | Просмотр и изменение комментариев к .zip-файлам. |
| **zipsplit** | Объединение нескольких .zip-файлов. |
| **zip** | Сжать файлы в архив. |
| **znew** | Перепаковать .Z-файлы в .gz-файлы. |

Перечень терминов и сокращений

| Сокращение | Определение |
| --- | --- |
| АПМДЗ | Аппаратно-программный модуль доверенной загрузки |
| БД | База данных |
| ВМ | Виртуальная машина |
| ЗПС | Замкнутая программная среда |
| ИТ | Информационные технологии |
| ОС | Операционная система |
| ПК | Персональный компьютер |
| ПО | Программное обеспечение |
| СДЗ | Средство доверенной загрузки |
| УЗ | Учетная запись |
| ФС | Файловая система |
| ЦС | Центр сертификации |
| ACL | Access Control List – список контроля доступа |
| AIDE | Advanced Intrusion Detection Environment – усовершенствованная система обнаружения вторжений |
| API | Application programming interface – программный интерфейс приложения |
| **BIOS** | **Basic Input/Output System** – базовая подсистема ввода-вывода |
| CA | Certification authority – центр сертификации (удостоверяющий центр) |
| CIFS | Common Internet File System – протокол, являющийся частью протокола SMB, для удаленного подключения нескольких платформ Windows, Linux или macOS, для идентификации и чтения файлов системы NTFS в Windows и HFS в macOS, чтобы совместно использовать эти файлы и принтеры между машинами с Windows и Linux |
| CSV | Comma separated values – текстовый формат, предназначенный для представления табличных данных |
| DHCP | Dynamic host configuration protocol – протокол динамической настройки сетевой конфигурации хоста |
| DNS | Domain name system – система доменных имен |
| EA | Extended Attributes – расширенные атрибуты |
| ENC | External node classifier – классификатор внешних хостов |
| ESR | Extended support release – релиз с расширенной (долговременной) поддержкой |
| HTTP | Hypertext transfer protocol – протокол передачи гипертекста |
| HTTPS | Hypertext transfer protocol secure – защищенная версия протокола передачи гипертекста |
| IP | Internet protocol – протокол межсетевого взаимодействия |
| IPA | Identity, policy and audit – система идентификации, политик и аудита (СИПА) |
| IPC | Interprocess Communication – средство межпроцессного взаимодействия |
| LDAP | Lightweight directory access protocol – протокол доступа к каталогам |
| LDAPS | Lightweight directory access protocol secure – защищенная версия протокола доступа к каталогам |
| LUKS | Linux Unified Key Setup – система шифрования дисков, которая хранит данные в зашифрованном физическом разделе |
| MAC | Media access control – уникальный идентификатор сетевого оборудования |
| MBR | Мaster boot record – главная загрузочная запись |
| MTA | Mail transfer agent – агент передачи почтовых сообщений |
| NTP | Network time protocol – протокол сетевого времени |
| SMB | Server Message Block – протокол соединения с устройствами со старыми версиями ОС Microsoft для подключения к серверам и обмена файлами |
| SMTP | Simple mail transfer protocol – протокол передачи почтовых сообщений |
| SSH | Secure shell – защищенная оболочка |
| SSL | Secure sockets layer – уровень защищенных сокетов |
| TCP | Transmission control protocol – протокол управления передачей данных |
| TFTP | Trivial file transfer protocol – протокол передачи файлов |
| TTY | Teletype – телетайп |
| UDP | User datagram protocol – протокол пользовательских датаграмм |
| URL | Uniform resource locator – сетевой адрес ресурса |
| YAML | Yet another markup language – язык разметки |
|  |  |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Лист регистрации изменений | | | | | | | | | |
| Номера листов (страниц) | | | | | Всего листов (страниц) в докум. | № докумен-та | Входящий № сопроводит. докум. и дата | Подп. | Дата |
| Изм. | изменен-ных | заменен-ных | новых | аннули-рованных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |