МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Российский химико-технологический университет имени Д. И. Менделеева

И. И. Митричев

АДМИНИСТРИРОВАНИЕ ОПЕРАЦИОННЫХ СИСТЕМ

Конспект лекций

Утверждено Редакционным
советом университета в
качестве учебного пособия

Москва
2019

УДК 004.451
ББК 32.973-018.2
М67

Рецензенты:
Доктор технических наук, профессор, заместитель директора по науке
ПАО Научный центр «Малотоннажная химия»
А. М. Бессарабов
Доктор технических наук, профессор, заведующий кафедрой Информатики и компьютерного проектирования Российского химико-технологического университета им. Д. И. Менделеева
Т. Н. Гартман

Митричев И. И.
М67 Администрирование операционных систем. Конспект лекций: учеб. пособие / И. И. Митричев. – М.: РХТУ им. Д. И. Менделеева, 2019. – 156 с.
ISBN 978-5-7237-1653-7

Содержатся такие разделы, как управление файлами и каталогами, файловыми системами и дисковыми разделами, процессами и сигналами, учетными записями пользователей. Изучаются средства и способы настройки и проверки сети в операционной системе Linux. Изложены основы программирования в командной оболочке bash и на языках обработки текстов awk и sed, широко применяемые при настройке серверных компьютеров. Производится сравнение базовых команд администрирования операционных систем Windows и Linux.

Предназначено для студентов, обучающихся по направлениям подготовки бакалавров 09.03.01 Информатика и вычислительная техника (профиль «Системы автоматизированного проектирования химических производств») и 09.03.02 Информационные системы и технологии (профиль «Информационные системы и технологии»).

УДК 004.451
ББК 32.973-018.2

ISBN 978-5-7237-1653-7

ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ 4
Глава 1. ОСНОВЫ АДМИНИСТРИРОВАНИЯ 5
Лекция 1. Введение. Установка ОС Linux 5
Лекция 2. Работа с командной оболочкой 12
Лекция 3. Работа с файлами и каталогами 21
Лекция 4. Сценарии командной оболочки bash 32
Лекция 5. Программирование для bash. Расширенные возможности командной оболочки bash 41
Лекция 6. Управление правами и пользователями 50
Лекция 7. Основы администрирования Windows 59
Глава 2. РАБОТА С ДАННЫМИ И ПРОЦЕССАМИ 73
Лекция 8. Процессы и сигналы 73
Лекция 9. Отложенное и регулярное выполнение заданий. Текстовые файлы и потоки 83
Лекция 10. Текстовые файлы и потоки (часть 2) 91
Лекция 11. Потоковые редакторы 98
Лекция 12. Работа с жесткими дисками и файловыми системами 108
Глава 3. АДМИНИСТРИРОВАНИЕ СЕРВЕРНЫХ СИСТЕМ 119
Лекция 13. Управление программным обеспечением (ПО) 119
Лекция 14. Системные журналы. Процесс загрузки и уровни выполнения 126
Лекция 15. Сетевые службы Linux 136
Лекция 16. Сетевые средства Linux 145
БИБЛИОГРАФИЧЕСКИЙ СПИСОК 155

ПРЕДИСЛОВИЕ

Начало XXI века ознаменовалось повсеместным распространением электронных устройств. Происходит крайне быстрое развитие как их программной, так и их аппаратной составляющих. Операционная система (ОС) – неотъемлемая программная составляющая современного компьютера и ряда других электронных устройств (смартфоны, планшетные компьютеры, ноутбуки, диагностическое медицинское оборудование и т. д.). которая предоставляет средства для взаимодействия других компьютерных программ с аппаратным обеспечением и друг с другом, а также обеспечивает удобство пользователя при работе с компьютером. Поэтому, глубокое и всестороннее изучение современных операционных систем, способов управления ими и их администрирования является важным звеном при подготовке специалистов в области информатики, вычислительной техники и информационных технологий.

XXI век – век цифровой информации, и крупномасштабная обработка и хранение этой информации в настоящее время осуществляется миллионами серверных компьютеров и суперкомпьютеров по всему миру, которые, преимущественно, используют ОС семейства Linux. Поэтому, в настоящем пособии основное внимание уделено свободно распространяемым операционным системам семейства Linux, которые популярны в научной среде и в среде информационных технологий (IT-среде), бесплатны и имеют открытый исходный код.

Глава 1. ОСНОВЫ АДМИНИСТРИРОВАНИЯ

Лекция 1. Введение. Установка ОС Linux

Операционная система – специальный уровень программного обеспечения, который отвечает за управление пользовательскими программами и аппаратными ресурсами. Она позволяет осуществлять легкий ввод-вывод, работу с файлами, программирование, запуск других программ. Операционная система (ОС) работает всегда в режиме супервизора, или ядра, т. е. имеет доступ ко всем аппаратным ресурсам и поддерживает любые допустимые инструкции. Все остальные компьютерные программы работают в пользовательском режиме, то есть могут использовать только ограниченное количество инструкций [1, с. 24–26, 63–74].

Операционная система позволяет пользовательским программам решать конкретные задачи, предоставляя готовые средства для взаимодействия с аппаратным обеспечением. Таким образом, программа, работающая в режиме пользователя, имеет дело с удобными абстракциями. Например, при чтении с диска и записи на диск программы будут работать с абстракциями – файлами, а не с командами, управляющими чтением с дисковых накопителей и записью на них.

Компоненты ОС:

– файловая система – отвечает за хранение и обработку информации;

– интерфейс – отвечает за взаимодействие с пользователем. Включает в себя графический интерфейс (оконную систему) и командную оболочку;

– ядро – обеспечивает взаимодействие остальных компонент.

Администрирование компьютера есть управление компьютером с целью настройки и поддержания его в соответствии с целевым использованием. Включает в себя установку программного обеспечения, настройку параметров (*конфигурирование*) операционной системы и прикладных программ, *мониторинг* (слежение за состоянием программных и аппаратных компонентов), сопровождение, устранение неполадок. Администратор осуществляет мониторинг состояния вычислительных машин, но ремонт аппаратной части не входит в его обязанности. Он имеет доступ к технической информации в отличие от обычного пользователя.

Задачи системного администратора:

– создание, удаление, активация и дезактивация учетных записей пользователей, групп пользователей, выделение ресурсов пользователям (место на диске, аппаратные ресурсы);

– установка, обновление и удаление программного обеспечения;

– подключение аппаратных средств (диски, системы хранения, серверы, сетевое оборудование);

– резервное копирование информации;

– поиск и диагностика неисправностей;

– управление и мониторинг системных журналов, документирование (важных операций, изменений в конфигурации);

– поддержка функционирования и настройка сети, систем безопасности;

– поддержка пользователей.

Свободное программное обеспечение – ПО, на которое распространяются следующие *свободы* (сформулированы впервые Р. Столлманом):

– свобода использования («*нулевая свобода*»);

– свобода изучения логики работы программы («*первая свобода*»). Подразумевает открытость программы, то есть, открытость ее исходных кодов;

– свобода распространения («*вторая свобода*»);

– свобода внесения изменений и улучшений («*третья свобода*»).

UNIX – операционная система, разработанная под руководством Д. Ричи, К. Томпсона и Б. Кернигана в Bell Labs в конце 60-х годов прошлого века. Особые черты: многопользовательская (возможна параллельная работа нескольких пользователей), многозадачная (параллельное выполнение различных программ на одном вычислительном устройстве за счет быстрого переключения между ними, называемого *переключение контекста*). В 80-х годах перестала развиваться как свободное ПО, созданы коммерческие (*проприетарные*) UNIX (Solaris, AIX, HP-UX, IRIX).

GNU (GNU is not UNIX) – проект создания свободного программного обеспечения (ПО) Фонда Свободного Программного Обеспечения, основанного Р. Столманном (1983).

Linux или GNU/Linux – общее название UNIX-подобных операционных систем на основе свободного ядра Linux и собранных для него библиотек и системных программ, разработанных в рамках проекта GNU. Первую ОС Linux написал Л. Торвальдс (1991). Существуют различные версии Linux, распространяемые в форме дистрибутивов (скомпонованы из базовых программ, необходимых для работоспособности системы, и дополнительных программ). Наиболее популярные дистрибутивы – Ubuntu, Slackware, Mint, Fedora, CentOS, Debian (рис. 1).

В ОС семейства Linux по умолчанию используется многопользовательский режим. Каждый пользователь имеет свою учетную запись, характеризуемую своим уникальным идентификатором (UID).

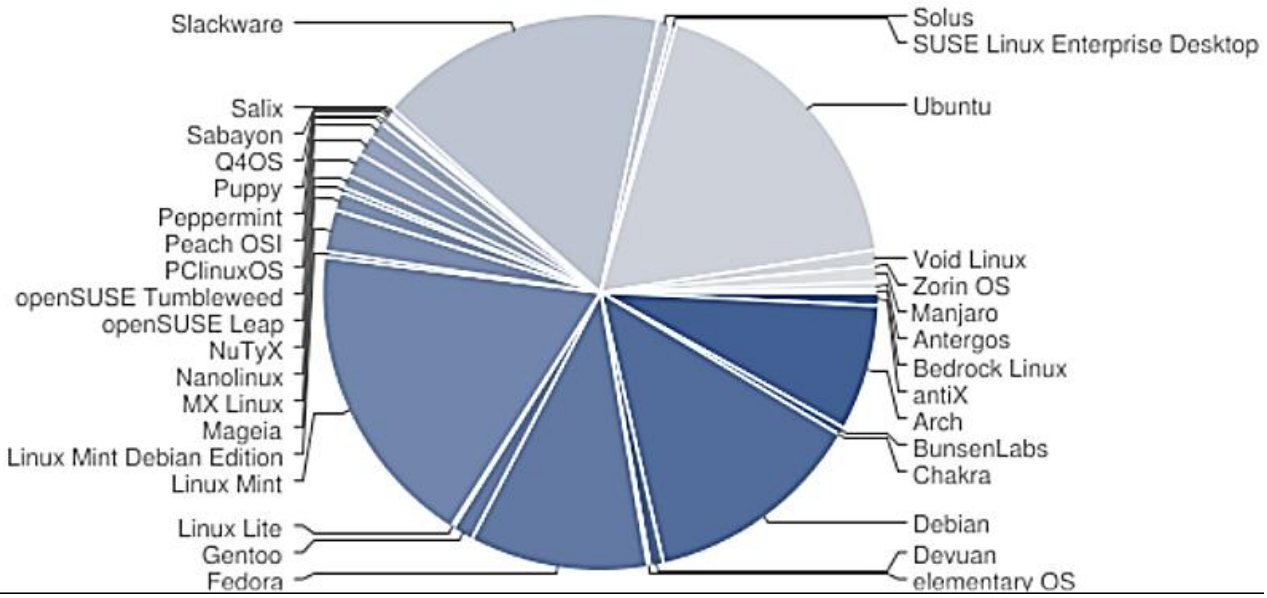


Рис. 1. Наиболее популярные дистрибутивы Ubuntu (2017) [2]

Файловая система построена как дерево, корнем которой является корневой каталог с названием «/» (косая черта, слэш). Слово «корневой» означает, что он соответствует корню воображаемого дерева. Соответственно, каждый каталог может быть родительским по отношению к другим – дочерним каталогам, которые образуют ветви воображаемого дерева. Файлы

пользователя хранятся в домашнем каталоге (пример: /home/student), пользователь для обращения к своему домашнему каталогу может использовать краткое имя «~» (тильда). Каталог (директория, папка – синонимы) /home – дочерний для /, и содержит по умолчанию домашние каталоги всех пользователей.

Командная оболочка может выполнять различные **команды**. Одна из базовых команд, команда **whoami**, показывает имя текущего пользователя ОС. Команда **ls** отобразит содержимое текущей директории.

В Linux символ «обратная косая черта» (backslash) используется для *экранирования*, то есть защиты символов от трактовки их командной оболочкой как специальных, или для соединения нескольких слов в одну строку. Пример: для отображения содержимого поддиректории «мои фото» в текущей директории, нужно выполнить в командной оболочке команду

ls мои\ фото

тем самым заэкранировав символ «пробел» и объединив слова «мои» и «фото» в единую строку. В этом – одно из отличий от ОС Windows, которые нужно запомнить.

В Windows и DOS для путей файлов и директорий используется обратный слеш «\», например, C:\Windows\System32. В Linux используется прямой слеш «/», например, /home/student/Desktop/lecture1.ppt

В Windows регистр имен файлов не играет роли: C:\Windows\System32\smss.exe и C:\WINDOWS\SYSTEM32\SMSS.EXE – одинаковые файлы**.** В Linux регистр имен файлов важен: /home/student/desktop/program.bin и /home/student/Desktop/program.bin – разные файлы.

Имена жестких дисков SCSI/SATA в Linux: /dev/sda, /dev/sdb, и так далее. Каждый диск может содержать несколько разделов (стандартно, с файловой таблицей msdos – 4 первичных, но можно создать расширенный раздел, который будет содержать множество логических разделов). Имена разделов в Linux: /dev/sda1, /dev/sda2 и т. д. Их можно обнаружить непосредственно в файловой системе в каталоге /dev, то есть это файлы особого типа «блочное устройство». Важная особенность Linux – доступ к

устройствам (разделам жестких дисков) может осуществляться как к файлам.

Ограничение для первичных разделов таблицы разделов msdos: размер 2.2 ТВ. Таблица разделов GPT более новая, однако Windows XP не поддерживает ее, а 64-разрядные версии Windows, начиная с Windows 7, поддерживают только в режиме EFI (режим загрузки компьютера с использованием более нового интерфейса начальной инициализации устройств компьютера EFI вместо BIOS). При установке нескольких ОС следует учитывать эти особенности.

Установку и работу с Linux изучим на примере популярной ОС для персональных компьютеров и серверов Ubuntu Linux.

Установка Linux может быть произведена с любого загрузочного устройства (DVD-диска, флэш-накопителя, съемного диска). Установка заключается в выполнении нескольких шагов:

– выбор языка установки;

– выбор имени компьютера (также говорят, имени хоста – hostname);

– разметка жестких дисков;

– выбор пароля и имени пользователя-администратора;

– настройка сети, выбор программного обеспечения для установки;

– настройка часового пояса.

Установку можно произвести внутри существующей ОС (безопасный метод), используя *виртуальную машину* – специальную программу, эмулирующую заданную конфигурацию аппаратных ресурсов и позволяющих устанавливать ОС, которым доступны сэмулированные ресурсы (а не существующие реально аппаратные ресурсы).

Установка Linux (в виртуальной машине)

1. Загрузите с официального сайта VirtualBox для Вашей операционной системы (хост-системы).

2. Создайте новую виртуальную машину. Если Ваша хост-система 64-разрядная, то устанавливайте 64-разрядную ОС в виртуальной машине.

3. Если 64-разрядную ОС выбрать в списке нельзя, то проверьте, включены ли Intel Virtualization Technology и VT-d в BIOS (обычно во

вкладке Advanced или Security). Для хост-системы Windows необходимо отключить встроенную функцию Hyper-V.

4. Выделите объем оперативной памяти под виртуальную машину. На системах с малым количеством RAM (до 8 Гб) использование виртуальной машины не рекомендуется, лучше установить нативную (реально установленную на диск) ОС.

5. Создайте новый виртуальный диск, тип VDI, динамический (размер может увеличиваться по необходимости), выделите не менее 10 Гб места на физическом жестком диске.

6. Вам необходимо скачать образ (iso) гостевой (guest, внутри виртуальной машины) операционной системы (Ubuntu) или использовать установочный диск.

Запустите виртуальную машину. Откройте iso-образ (желтая кнопка) и следуйте инструкциям по установке.

Для компьютеров, использующих EFI, нужно запускать установщик Linux в режиме EFI (происходит автоматически при загрузке с установочного диска).

Также программа установки может предложить вариант установки «Установить рядом с Windows» (удобно для начинающих, обычно безопасно для данных Windows). Рекомендуемый вариант (для системного администратора, требует определенных навыков) – указать разделы вручную. Далее требуется:

1) создать раздел с типом файловой системы ext4 (в свободном месте или уменьшить сначала размер другого раздела на 30 или более ГБ), точка монтирования «/»;

2) создать раздел swap, тип swap, с размером, равным размеру оперативной памяти.

7. Установите дополнения гостевой ОС (позволяет использовать общий буфер обмена между хост-системой и гостевой ОС, а также ряд других дополнительных возможностей). В терминале (вызов терминала Ctrl-Alt-T) введите команду

sudo apt install virtualbox-guest-additions-iso

Остановите виртуальную машину. Далее в настройках виртуальной машины примонтируйте (означает «подключите как файл особого типа к Вашей файловой системе») CD-диск /usr/share/virtualbox/VboxGuestAdditions.iso. Запустите виртуальную машину. В гостевой системе на рабочем столе появится CD-ROM с установщиком дополнений гостевой ОС. Запустите установщик и следуйте его инструкциям до окончания установки.

Работа в ОС Ubuntu

После запуска системы в левом верхнему углу доступна панель управления Dash, для ее запуска щелкните на пиктограмму (рис. 2) или нажмите клавишу «Win» на клавиатуре. Откроется меню, из которого Вы сможете запустить различные приложения, к примеру терминал (приложение, позволяющее использовать командную оболочку), LibreOffice (набор офисных программ для работы с текстом, электронными таблицами и презентациями), gedit (блокнот с подсветкой синтаксиса для программирования).

Работа с окнами в Ubuntu основана на использовании оконной системы XWindow, которая базируется на клиент-серверной структуре. На компьютере запускается оконный сервер XServer, к которому можно установить подключение с помощью клиента XClient; по умолчанию подключение осуществляется с того же компьютера.

Свободные графические среды рабочего стола для UNIX-подобных операционных систем: KDE (от англ. K Desktop Environment), Gnome, Unity. Последняя среда использовалась по умолчанию в Ubuntu 11.10-17.04. Остальные версии Ubuntu используют Gnome.

Правила выбора имени пользователя

– Имя может содержать буквы латинского алфавита, цифры, символ подчеркивания «_», дефис «-», причем начинаться с цифры и дефиса не может;

– Не используются пробел, символы «:», «@», «#» и т. д.;

– Не используются специальные символы (табуляция, конец строки и т. п.).

id – получить информацию об учетной записи (UID, GID).

Правила выбора паролей

– Пароль должен быть сложным, устойчивым к простым криптоатакам: не основывать на имени пользователя, или известных данных (день рождения, город проживания, любимое блюдо), не должен быть словарным словом.

– Следует назначать срок действия пароля (максимальный и минимальный). При смене пароля новый должен отличаться от старого.

– Следует запоминать пароль, а не записывать.

– Следует пресекать попытки пользователей меняться паролями.

Создание сложных паролей возможно командой **pwgen**.

pwgen 10 – сгенерировать пароли из 10 символов.

Дополнительная литература к лекции 1:

Таненбаум Э., Бос Х. Современные операционные системы. Глава 1. Введение /4-е изд. СПб.: Питер, 2015. С. 24–26, 63–74.

Лекция 2. Работа с командной оболочкой

Командная оболочка (англ. shell) – программа, взаимодействующая с пользователем с помощью текстового интерфейса. Командная оболочка предназначена для ввода команд.

Примеры командных оболочек:

Command.com – оболочка MS-DOS;

cmd.exe – Windows NT, XP

PowerShell – Windows XP (последние версии) и более новые версии Windows

sh – Bourne shell, классическая оболочка UNIX;

bash – Bourne Again SHell, самая распространенная оболочка для ОС семейства Linux;

dash – POSIX-совместимая оболочка для UNIX очень компактного размера;

csh – оболочка C, где используется синтаксис языка C;

tcsh – версия оболочки C с возможностью интерактивного редактирования командной строки;

fish – дружественная оболочка, имеющая множество подсказок для освоения;

ksh – оболочка Korn, есть возможность редактирования команд;

zsh – Z-shell, оболочка, имеющая расширенные возможности подстановки команд, автодополнения, и множество настроек.

В операционных системах Linux список установленных оболочек хранится в файле /etc/shells.

Командная оболочка является интерпретатором, то есть, выполняет введенные команды построчно.

Функции командных оболочек:

– Интерпретация ввода командной строки. Доступ к командам и результатам их выполнения;

– Поддержка переменных, специальных символов и зарезервированных слов;

– Обработка файлов, операций стандартного ввода и вывода;

– Реализация специального языка программирования оболочки.

В Linux после запуска shell на экран выводится **приглашение к вводу команд**. Если приглашение начинается с **$**, это означает, что используется учетная запись обычного пользователя, если с # – то используется учетная запись суперпользователя (root). Суперпользователь имеет административные права (*привилегии*), обладает доступом ко всей системе и настройкам. По умолчанию в Ubuntu (и других, основанных на Debian дистрибутивах) суперпользователь root не имеет пароля, так что напрямую войти с его учетной записью в систему невозможно. Однако есть команды, которые повышают уровень прав обычного пользователя до root, т. е. позволяют получить доступ к командной оболочке под учетной записью суперпользователя или выполнить отдельные команды от его имени.

Как перейти в режим root?

1) **su** (заметьте: в приглашении появляется #);

2) **su -** – копирует переменные окружения из сеанса, где вызывается команда, в сеанс суперпользователя;

3) **sudo -i –** повышение прав, пользователь должен иметь на это права (определены в /etc/sudoers).

Задание: выясните, пароль какого пользователя требуется вводить в случаях 1, 2, 3.

Для дистрибутивов, основанных на Debian, используется команда sudo (Вы не сможете открыть командную оболочку пользователя root, используя 1 и 2, как показано выше). В этих дистрибутивах пользователь root не имеет пароля.

Кто может повысить свои права до уровня root?

1) знающий пароль root;

2) указанный в файле /etc/sudoers пользователь или член группы, указанной в этом файле (группа администраторов названа wheel).

Создаваемый по умолчанию при установке ОС Linux единственный пользователь системы входит в число sudoers.

Linux – многозадачная и многопользовательская система. Завершение сеанса одного пользователя не приводит к прекращению функционирования ОС.

Приглашение ввода команд

Приглашение хранится в переменной окружения PS1 (**echo $PS1**) в виде

u@h:w$

Можно определять другой формат, используя следующие спецификаторы: d – текущее время; h – имя хоста; n – символ новой строки; A – текущее время (ЧЧ:ММ); u – имя пользователя; w – (в нижнем регистре) полный путь к рабочему каталогу; W – (в верхнем регистре) только рабочий каталог.

Горячие клавиши Linux

Копирование/вставка – Ctrl+Insert / Shift+Insert (вставка – также Ctrl-Shift-V).

Закрыть командную оболочку - Ctrl+D или **exit**.

Автодополнение вводимых команд, переменных и т. п. – [Tab]; дважды нажмите клавишу [Tab] – увидите список возможных окончаний. Если строка начинается с символа «$» – дополняется имя переменной оболочки, с символа «~» – дополняется имя пользователя, с символа «@» – дополняется имя хоста (*компьютера в сети*).

Переключиться на виртуальный терминал tty1-tty7 – Ctrl+Alt+F1/ Ctrl+Alt+F7.

История выполненных команд: чтобы просмотреть введенные ранее команды, используйте клавиши со стрелками [Вверх] и [Вниз], сочетание клавиш [Ctrl]-[R] позволяет по введенным начальным буквам ранее использовавшейся команды вывести последнюю введенную.

Фоновый режим исполнения команд – **команда &** (чтобы иметь возможность продолжать использовать оболочку командной строки, введите после команды, запускающей приложение, символ «&»). Сочетание клавиш [Ctrl]-[Z] приостанавливает работу приложения, запущенного в терминале.

Остановка выполнения текущей команды (используйте при зависании) – [Ctrl]-[C]

Далее в этом пособии будем рассматривать интерпретатор команд bash, поскольку он установлен по умолчанию в Ubuntu и многих других популярных ОС семейства Linux.

Bash умеет использовать знаки подстановки и регулярные выражения (подробнее о них – далее), например, для поиска файла с названием «example1.txt» введите:

ls exam*.txt

Шаблоны подстановки и перечисление

* – любое количество любых символов, или их отсутствие, кроме имен файлов, начинающихся на «.» (скрытые файлы):

echo *

? – заменяет один символ в имени файла;

[bcd] – один из символов перечисления.

Пример:

echo .[bcd]* – выведет все скрытые файлы, начинающиеся на b, c, d.

Исключение символов производится с помощью знака «!» (**[!bcd]).**

echo .bash{rc,_profile} – перечисление (набор вариантов);

echo {,/usr}/{,s}bin – перечисление (другой пример);

echo {,/usr{,/local}}/{,s}bin – вложенный перебор.

Вы можете поменять цветовую гамму в переменной **PS1**: измените в части строки «[e[31m]» номер 31 на любой другой.

Команда Linux – строка символов из имени команды и аргументов, разделенных пробелами.

Аргументы предоставляют команде дополнительные параметры. Аргументы включают также опции и имена файлов и каталогов.

ls -l -a file1 file2 или **ls -la file1 file2** (команды идентичны) – имя команды ls, две опции l и a, два файла file1 и file2.

Команды, являющиеся частью оболочки, называются встроенными. cd, exec и другие встроенные команды могут отличаться для различных вариантов оболочек.

Кроме встроенных команд, возможен запуск исполняемых файлов и файлов **сценариев командной оболочки** (*скриптов*), содержащих последовательно выполняемые строки с командами оболочки.

Некоторые скрипты могут выполняться процессами Linux, как например, планировщик задач cron. При этом возникает проблема выбора оболочки. Принято, в первой строке указывать используемую оболочку: **#!/bin/bash** – для оболочки bash, **#!/bin/sh** – для оболочки sh.

Строки скрипта, начинающиеся с символа #, при обработке оболочкой пропускаются как комментарии.

Опции передают командам в разном стиле (рис. 2), что сложилось исторически при параллельном развитии нескольких ветвей UNIX. Стиль опций зависит от самой команды, многие команды поддерживают несколько стилей (например, **ps**).

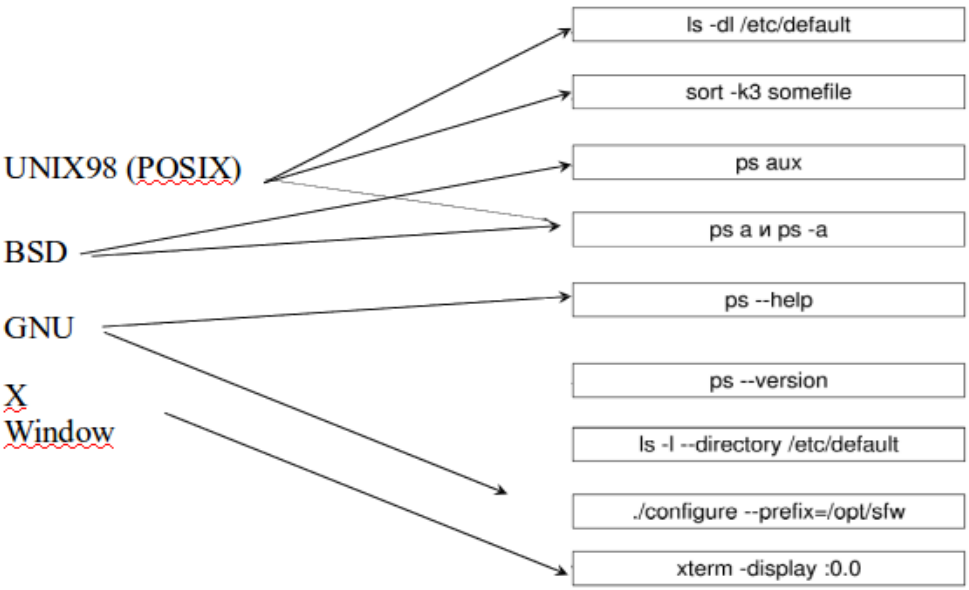


Рис. 2. Стили написания команд в Linux

Команды делят на **встроенные** и **системные**. Встроенные команды выполняются самой оболочкой, системные – это исполняемые файлы. Встроенные команды в разных оболочках могут выполняться по-разному. Встроенные команды выполняются быстрее, чем системные. Для многих встроенных команд есть системные аналоги (**pwd** и **/bin/pwd**).

Места расположения системных команд: /bin, /sbin, /usr/bin, /usr/sbin, /usr/local/bin, /usr/local/sbin.

Для ввода длинной команды используется «\» (перевод строки). Для ввода нескольких команд в одной строке используется точка с запятой «;».

Команда1 && Команда2 – команда 2 выполняется только в случае удачного выполнения команды 1.

Команда1 || Команда2 – команда 2 выполняется только в случае неудачного выполнения команды 1.

Получение помощи

help – встроенная помощь оболочки;

mkdir --help – отобразить справку для команды mkdir (справка команды);

man – система помощи в любой UNIX системе;

info – иерархическая система помощи GNU TexInfo.

Самой популярной является система помощи man (сокращение от manual – руководство по эксплуатации). Man имеет восемь стандартных разделов. Перечень разделов [4]:

1 – Пользовательские команды – это команды, которые могут быть выполнены пользователем из оболочки.

2 – Системные вызовы – функции, которые являются оберточными для функций, выполняемых ядром.

3 – Библиотечные вызовы (большинство функций libc).

4 – Специальные файлы (устройства) – файлы в директории /dev, которые позволяют получить доступ к устройствам через ядро.

5 – Форматы файлов и файлы конфигурации – описывает различные форматы файлов и файлы конфигурации.

6 – Игры.

7 – Обзорные статьи, условные обозначения и соглашения, разное.

8 – Команды управления системой – большую часть команд может выполнять только root (пример – mount (8)).

На некоторых системах также доступен раздел 9 – Подпрограммы ядра.

Синтаксис команды man

man <номер раздела> <имя страницы>

Программа man находит страницу документации (хранятся в виде архивов), расшифровывает и открывает программу просмотра текста less. Горячие клавиши:

PgUp, PgDn – перемещение по тексту;

<пробел> – следующая страница;

</> строка – поиск подстроки вниз;

<?> строка – поиск подстроки вверх;

<!> – вызов командной строки для тестирования команд;

<n> – следующее вхождение искомой строки;

<q> – выход.

Опции команды:

man -a – получить все страницы, а не только первую найденную;

man -k <строка> или **apropos <строка>** – поиск по подстроке в имени страницы;

man -f <строка> или **whatis <строка>** – поиск по полному имени страницы.

Примеры:

man passwd

man -k clock

Каждая страницы описания команды в man имеет разделы (рис. 3):

NAME – информация для поиска по ключевому слову;

SYNOPSIS – формат вызова, опции и аргументы;

DESCRIPTION – описание объекта (программы, файла, библиотеки);

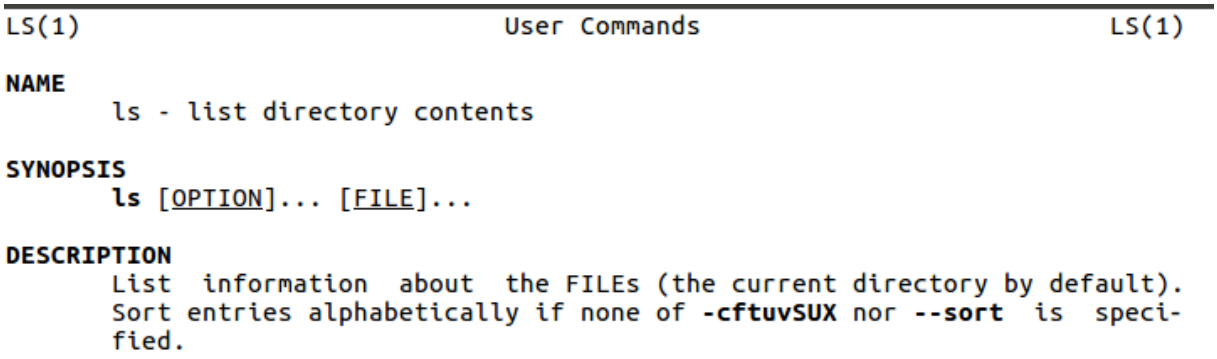
OPTIONS – подробное описание опций;

FILES – файлы, связанные с командой;

AUTHOR – имя автора с указанием электронной почты;

SEEALSO – указатели на другие страницы man;

COPYRIGHT – права собственности, политика распространения.



```
LS(1)                          User Commands                          LS(1)

NAME
       ls - list directory contents

SYNOPSIS
       ls [OPTION]... [FILE]...

DESCRIPTION
       List  information  about  the FILEs (the current directory by default).
       Sort entries alphabetically if none of -cftuvSUX nor --sort  is  speci-
       fied.
```

Рис. 3. Пример страницы man для команды ls

Переменные оболочки и окружения

Bash позволяет временно сохранять данные в переменных оболочки. В названиях переменных допустимы только буквы, цифры, и знак подчеркивания «_».

Переменные оболочки доступны только в той оболочке, в которой были описаны.

Присвоение значения переменной

Var1=Вася

Присвоение значения переменной, когда значение содержит пробелы

Var1='Вася Петров'

Использование значения переменной, присвоение

Var2=$Var1

Список всех переменных оболочки можно получить командой **set**, а команда **unset Var1** удалит переменную Var1.

Подоболочка (subshell) представляет собой отдельный экземпляр командного интерпретатора, являющийся дочерним (или, порожденным) процессом от родительской оболочки.

Окружение – один из способов передачи информации процессов в системе друг другу. Изменение значения *переменной окружения* приводит к изменению поведения программ.

Перевод переменной оболочки в переменную окружения: **export VAR1**

Переменные оболочки не копируются в дочерние процессы и подоболочки, копируется только окружение. Из общего правила есть

исключение: при использовании командной подстановки $( ) или `` запускается дочерняя подоболочка (subshell), в которую копируются все переменные текущей оболочки.

env – список переменных окружения.

Важнейшие переменные окружения:

HOME – путь к домашнему каталогу; LOGNAME и USER – имя пользователя; MAIL – путь к почтовому ящику;

PATH – путь поиска исполняемых файлов; PS1 – вид приглашения оболочки; PWD – имя текущего каталога;

OLDPWD – имя предыдущего каталога; SHELL – имя исполняемого файла оболочки; TERM – тип терминала; HOSTNAME – имя хоста; SHLVL – номер загруженной оболочки.

Чтобы сохранить переменные между сессиями, нужно добавить их в файл настроек переменных окружения. Список таких файлов (точка «.» означает скрытый файл):

/etc/profile – переменные, общие для всех пользователей;

~/.bash_profile, ~/.bash_login, ~/.profile – настройки для пользователя;

~/.bashrc – выполняется при ручном запуске оболочки.

Команда **clear** – очистить экран.

История команд

Также введенные ранее команды можно узнать из истории команд.

~/.bash_history – файл с историей команд по умолчанию. $HISTFILE – имя файла истории (можно изменять). $HISTFILESIZE – количество команд, запоминаемых в истории.

Команда history

!номер команды – выполнить ее снова;

!! – последняя выполненная команда;

!<префикс> – поиск недавно введенной команды, начинающейся на <префикс>;

!?<подстрока> – поиск недавно введенной команды по подстроке;

fc <префикс> – поиск и редактирование недавней команды;

<Ctrl> + <R> – поиск недавней команды по подстроке;

<вверх>, <вниз> – последние команды.

Псевдонимы команд

Псевдонимы служат для ускорения набора длинных часто используемых команд. alias – вывести список псевдонимов;

alias lls='ls ld' – создать новый псевдоним;

unalias lls – удалить псевдоним;

unalias a – удалить все псевдонимы.

Командная подстановка (англ. *command substituion*) – результат выполнения одной команды автоматически передается в качестве аргументов другой команде. Используются два оператора: обратные косые кавычки ``, и $( ):

внешняя_команда `внутренняя_команда`

внешняя_команда $(внутренняя_команда)

Примеры:

ls l `which rpm`

ls l $(cat /etc/shells)

Оператор «$(( ))» используется для вычисления арифметических выражений.

Внимание: bash поддерживает только целочисленную арифметику! Для выполнения операций с числами с плавающей запятой используйте специальные программы, например, калькулятор с интерфейсом командной строки **bc**.

По умолчанию в Linux в качестве разделителя целой и дробной части используется точка «.»

Оператор передачи вывода результатов одной команды другой команде «|»:

echo "5+2.4" | bc

Дополнительная литература к лекции 2:

Береснев А.Л. Администрирование GNU/Linux с нуля. Гл. 2. Работа в оболочке bash /2-е изд., перераб. и доп. СПб.: БХВ-Петербург, 2010. С. 25–50.

Лекция 3. Работа с файлами и каталогами

Приведем основные команды для работы с файлами и каталогами.

pwd – выводит список файлов и каталогов.

ls -la – список файлов и каталогов с подробной информацией (-l), включая скрытые файлы (-a). Пример результаты выполнения данной команды разобран на рис. 4.

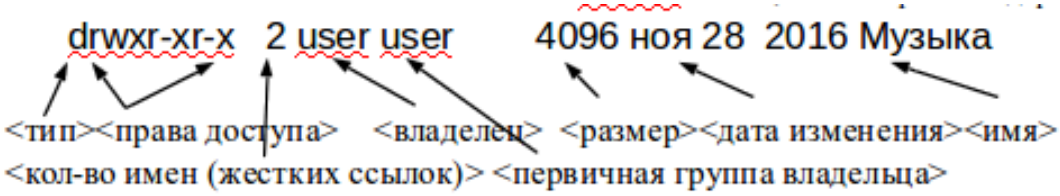


Рис. 4. Результат выполнения команды ls -la

Опция -d позволяет не выводить содержимое каталогов, а трактовать их как обычные файлы. Информацию **только** о каталогах можно получить, добавив к концу команды «*/»:

ls -ld */

Это связано с тем, что bash раскрывает шаблоны подстановки (*) до выполнения команд.

В первом столбце вывода команды **ls -l** мы видим тип файла. Директория, напомним, также является файлом особого типа в Linux (обозначение - d). В табл. 1 приведены возможные типы файлов.

Таблица 1

Типы файлов в Linux

Обозначение	Пояснение
–	обычный файл
d	директория
l	символическая ссылка (указатель на другой файл)
b	файл блочного устройства (например, /dev/sda)
c	файл символьных устройств (терминала, мыши)
p	именованный канал (для организации взаимодействия между процессами)
s	сокет (для организации межпроцессного взаимодействия по сети)

Перемещение по директориям осуществляется командой cd.

cd ~ – переместиться в домашнюю директорию,

cd .. – переместиться на уровень выше в дереве директорий,

cd - – переместиться в предыдущую директорию, который был открыт до этого в текущей командной оболочке,

cd <директория> – переместиться в заданную директорию.

Создание файлов

> filename – создать пустой файл (путем переадресации стандартного потока вывода, в который ничего не подано, в файл);

>> filename – создать файл, но не перезаписывать, если файл существует.

touch filename1 filename2 ... – создание файла или нескольких файлов, или изменение даты для существующего файла.

Пример:

echo "333" > alive

ls -l a*

sleep 70

touch alive

ls -l a*

Разберем пример:

echo "333" – создает файл «alive», содержащий «333»;

ls -l a* – выводит подробную информацию о файлах, начинающихся на букву «a»;

sleep 70 – ожидание в течение 70 секунд («сон»);

touch alive – изменяет дату модификации файла «alive»;

В выводе команды **ls -l a*** во второй раз даты модификации файла «alive» будет отличаться от выведенных при первом запуске команды.

Перечисление в Linux

ls f{1,2,3}

Для **удаления файлов и каталогов** используется команда

rm <filename(s)>

Опции:

f – удаления без подтверждения (опасная опция);

i – спрашивать подтверждение на удаление каждого файла;

r – рекурсивно удалить каталог и его содержимое.

Обязательно выполняйте проверку имен файлов (отобразите их с помощью ls) перед удалением по шаблону! Пример удаления трех файлов:

ls f*[abc]

fa fb fc

rm f*[abc]

rm -rf – удаление рекурсивное и без подтверждения. Служит для удаления директорий, содержащих множество поддиректорий. **Крайне опасная команда**, не выполняйте ее в режиме суперпользователя без крайней необходимости!

Опция -i – удаление с подтверждением для каждого объекта.

Создание и удаление каталогов – команды mkdir и rmdir

mkdir <dir> – создание каталога;

mkdir -p dir1/dir2/dir3 – создание дерева директорий (опция -p – создать систему вложенных каталогов);

rmdir <dir> – удаление пустого каталога;

rmdir -p dir1 – удаление дерева пустых каталогов, начиная с dir1.

Копирование файлов

cp <source> <destination> – скопировать файл source в файл/расположение destination. Если destination – имя существующей директории, копирование произойдет в нее. Чтобы при просмотре команд не возникало двусмысленности, используйте / на конце для destination, в случае, если destination – это директория.

cp <source> <destination>/ – копирование в директорию. Если директории не существует, будет выдана ошибка (в отличие от предыдущей команды, где будет создан/перезаписан файл destination).

cp <file1> <file2> ... <filen> <dir> – копирование файлов в каталог dir;

cp dir/f{1,2} – копирование с подстановкой вариантов (плохой стиль), эквивалентно команде cp dir/f1 dir/f2, создается копия файла f1 с именем f2 в том же каталоге dir, где находится исходный файл.

Для рекурсивного копирования директорий предназначены следующие опции:

cp -r <dir1> <dir2>

Опция -R команды cp работает аналогично опции -r.

Перемещение и переименование файлов выполняется командой mv:

mv <source> <destination>

Можно перемещать несколько файлов:

mv <file1> <file2> ... <fileN> <dir>

Команда mv работает как с файлами, так и с директориями.

mv dir1/ dir2/ – перемещение директории. Во избежание ошибок, при использовании mv и cp на конце названий директорий ставьте слэш.

Поиск файлов

find <where> <what> <how>

Аргументы команды **find**:

where – директории, начиная с которых будет осуществлен поиск (поиск также производится и в поддиректориях);

what – атрибуты файла (имя файла, размер, владелец файла, тип, даты доступа и изменения и пр.

how – как искать и выводить найденную информацию.

Поиск ведется в названиях файлов/директорий. Для поиска по файлам используйте **grep** (см. далее).

Пример: в домашнем каталоге найти файлы f1 и f2.

find ~ -name f[1,2]

Результат выполнения команды:

./d1/f1

./d1/f2

Пример: найти все файлы, начинающиеся на f, в домашней директории (кавычки обязательны)

find ~ -name "f*"

find ~ -name 'f*'

Экранирование (поиск файла с названием «f*»)

find -name 'f*'

Критерии поиска команды find

name – поиск по имени/шаблону;

iname – поиск по имени без учета регистра;

type – поиск по типу;

size – поиск по размеру;

empty – поиск только пустых файлов;

mtime – поиск по дате модификации;

atime –поиск по дате обращения;

ctime – поиск по дате создания;

perm – поиск по правам доступа;

user и group – поиск по принадлежности пользователю и группе.

Пример: найти все файлы в ~/orca, которые разрешено считывать, изменять и исполнять всем пользователям и группам

find ~/orca -perm 777

. – поиск в текущем каталоге;

/ – искать, начиная с корневого каталога системы

Типы файлов для поиска (значения критерия type): d – каталог; f – обычный файл; l – символическая ссылка.

Пример: поиск символических ссылок в ~/orca:

find ~/orca -type l

Пример: найти все файлы/директории в /tmp/t, доступ к которым последний раз осуществлялся 1) 17 минут назад, 2) меньше 18 минут назад.

1) **find /tmp/t -amin 17**

2) **find /tmp/t -amin -18**

Соединение поисковых критериев через логическое «ИЛИ» дает опция -o:

find ~ -name "f*" -o -empty

Опция exec позволяет выполнять команды с найденными файлами:

**find ~ -name "*lost*" -exec mv {} ~ **

С помощью данной команды будут выполнены поиск и удаление всех файлов, имена которых содержат в себе строку «lost».

{} \; – специальный синтаксис опции -exec. Вместо фигурных скобок будут подставлены имена всех найденных файлов, и эти имена будут переданы в качестве аргументов командам, вызванным опцией -exec. Конструкция «\;» применяется для указания конца опции -exec.

Поиск файлов командой locate по индексной базе

locate <подстрока имени файла> – поиск по индексу (базе данных). Не воспринимает файловые шаблоны поиска, а только строки. Имеются

версии команды, работающие с регулярными выражениями.

Задание: создайте новый файл, попытайтесь найти его с помощью locate. Получилось ли это у Вас?

updatedb – команда для обновления индекса поиска для **locate**. Стандартно – выполняется в автоматическом режиме (через планировщика заданий cron). В Ubuntu стандартно выполняется ночью, чтобы не нагружать компьютер в «часы пик» (locate полностью загружает одно ядро процессора, а время работы зависит от размера подключенных файловых систем). Выполняется только от суперпользователя.

Автоматическое выполнение настраивается в Ubuntu в файлах, которые можно вывести на экран командами

cat /etc/crontab

cat /etc/cron.daily/mlocate

Подробно о настройке файла **/etc/crontab** – в последующих лекциях.

which <чтоищем> – выводит путь к заданной команде;

whereis <чтоищем> – команда UNIX-подобных ОС для получения информации о расположении файлов документации и исходных файлов системных программ.

Команда file определяет тип содержимого файла при помощи базы данных сигнатур файлов (*магических чисел*, англ. *magic numbers*).

/usr/share/file/magic.mgc – база сигнатур файлов. Список возможных сигнатур с пояснениями – на сайте [4].

Между файлами могут существовать **жесткие связи**, или *жесткие ссылки* (вспомним, что директория – это особый тип файла). Жесткая связь – это наличие у файла другого имени. Оба файла, указывают на те же самые **метаданные** (*индексные дескрипторы, или, inode*). Количество имен у файла отражается в третьем столбце в выводе команды **ls -l** (рис. 5).

Вопрос: почему у директории всегда имеется хотя бы два имени? Подсказка: посмотрите вывод команды **ls <название_любой_пустой_директории>**. Ответ с объяснением представлен в [3, с. 62–86].

Метаданные хранятся в таблицах inodes по всему диску (в каждой группе блоков – см. лекцию 12). Номер inode для каждого файла можно видеть в первом столбце вывода команды **ls -i**.

df --inodes – сколько свободных inodes осталось на в разделах.

Для создания жесткой связи используйте ln:

ln f1 f1000 – установить жесткую связь от файла f1 к файлу f1000.

```
root@01-PK:/tmp# ls -li
итого 24
27132465 -rw-r--r-- 1 root    root       0 ноя  1 02:40 a
27132466 -rw-r--r-- 1 root    root       0 ноя  1 02:40 b
```

Рис. 5. Получение информации о файлах с указанием inode

Внимание: при удалении файла по любому из имен (f1 или f1000) файл будет полностью удален! В этом заключается жесткость связи.

Индексные дескрипторы (inode), или метаданные, есть данные о самих файлах. Они содержат (по стандарту POSIX)

– размер файла в байтах;

– идентификатор владельца и группы владельцев файла;

– права доступа к файлу;

– идентификатор устройства, на котором хранится файл;

– счетчик имен (жестких ссылок) для файла;

– timestamps – даты доступа к файлу, его создания и изменения;

– указатели на блоки с данными файла.

Обычно, около 1% файловой системы зарезервировано для хранения метаданных.

Также существуют **мягкие связи** между файлами. Один файл ссылается на другой, указывает на него. Связь в этом случае – мягкая, поскольку удаление ссылки не приводит к удалению самого файла. Мягкие связи организуют с помощью **символических ссылок** (**ln -s**):

ln -s f1 f2

После выполнения этой команды файл с именем «f2» является указателем на файл с именем «f1».

Удалить ссылку можно:

– с помощью rm, как обычный файл;

– командой unlink (unlink имя_ссылки).

Определение свободного места на блочных устройствах

df -h – определить свободное место.

Команда df показывает наличие свободного места на смонтированных файловых системах (см. файлы /etc/mtab, /etc/fstab).

df /dev/sda – информация о свободном месте на заданном устройстве.

df -h – отображение в единицах, удобных для чтения (human readable format) –мегабайтах, гигабайтах, а не в 512-байтных блоках.

du -sh <имя_директории> – определение размера директории.

du -shc ~/* – вывод информации по поддиректориям с общим итогом: -s – суммарный результат для каждая поддиректории, -c – общая сумма.

Команда **who** выводит пользователей, работающих в системе в настоящий момент. Эта информация хранится в двоичном файле /var/run/utmp.

last – данные о прошлых входах пользователей в сеанс (log in). Указанная информация хранится в двоичном файле /var/log/wtmp.

Регулярные выражения – специализированная система поиска и обработки текста, основанная на сравнении части текста с образцами для поиска (шаблонами) (табл. 2).

Таблица 2

Распространенные регулярные выражения [3]

Метасимвол	Класс	Расширенный	Описание
^	Шаблон	Нет	Начало строки
$	Шаблон	Нет	Конец строки
\<	Шаблон	Да	Начало слова
\>	Шаблон	Да	Конец слова
.	Шаблон	Нет	Любой символ (в т. ч. – пробел)
[]	Шаблон	Нет	Набор символов
()	Шаблон	Да	Группировка символов
|	Шаблон	Да	Инфиксный оператор (или)
*	Квантификатор	Нет	Вхождение любое количество раз
+	Квантификатор	Да	Вхождение не менее одного раза
?	Квантификатор	Да	Вхождение не более одного раза
{n}	Квантификатор	Да	Вхождение n раз
{n,}	Квантификатор	Да	Вхождение не менее n раз
{n,m}	Квантификатор	Да	Вхождение от n до m раз

Шаблон – особый символ, заменяющий один или более обычных символов.

Квантификатор – указывает число вхождений символа или группы символов, которая находится перед ним в регулярном выражении.

Классификация регулярных выражений

– обычные регулярные выражения (basic regexp);

– регулярные выражения с расширенным синтаксисом (extended regexp).

Поиск текста с помощью grep

Команда grep осуществляет поиск и сравнение с образцом в заданном тексте. Текст может поступать команде из файла или стандартного потока ввода. Существуют три разновидности команды grep:

grep – возможно использование обычных регулярных выражений;

grep -F или **fgrep** – версия без возможности использования регулярных выражений;

grep -E или **egrep** – возможно применять расширенный синтаксис регулярных выражений.

grep 'bash$' /etc/passwd – поиск всех пользователей системы, использующих оболочку bash (выражение в апострофах для предотвращения интерпретации символа доллара, который обозначает конец строки).

egrep -c '^.{1.3}:' /etc/passwd – определить число пользователей, имя которых содержит не более трех символов.

Важные опции grep

-A n – вывести указанное число строк n после каждого совпадения с шаблоном поиска;

-B n – вывести указанное число строк n до каждого совпадения с шаблоном поиска;

-C n – вывести указанное число строк n до и после каждого совпадения с шаблоном поиска;

-i – не учитывать регистр символов при поиске;

-v – инвертировать результаты поиска – будут выведены все строки, НЕ содержащие совпадений;

-w – искать только целые слова (буквы, цифры и знак подчеркивания, окруженные другими символами и/или концом строки) ;

-c – считать только число совпадений, но не выводить совпадения;

-o – печатать только совпавшую с шаблоном часть строки, а не всю строку;

--include=*.{c,h} – включить в поиск только .h и .c файлы;

--exclude=*.{c,h} – исключить из поиска файлы с расширением .h и .c;

-r – рекурсивный поиск (используйте для поиска внутри файлов вместо find);

-l – показывать только имя файла, а не совпадающие строки;

grep -rl "honey" ~ – рекурсивный поиск файлов, содержащих текст «honey», в домашней директории.

Примеры использования шаблонов и квантификаторов

Наиболее распространены квантификаторы:

* – любое количество символов (кроме точки в именах скрытых файлов);

? – один символ или их отсутствие.

Пример – вывести список скрытых файлов, в именах которых после точки имеется шесть любых символов:

$ **echo .??????**

Результат:

.bashrc .config .emacs~ .gconfd .gnome2 .isotmp .mcoprc .themes

Перечисления Bash осуществляют подстановку вариантов строки, заданной в фигурных скобках. Пример – вывод на экран строк this1 и this52:

$ **echo this{1,52}**

Результат:

this1 this52

Базовое перенаправление ввода/вывода

> – перенаправление вывода (**echo 'Begin the lesson' > 1.txt**);

< – перенаправление ввода (**grep -i '^Begin' < 1.txt**);

| – передача результата выполнения команды.

Оператор pipe «|» приводит к запуску дополнительной дочерней оболочки (подоболочки, subshell) и расходованию времени (время уходит на создание дочерней оболочки – системного процесса, а также уничтожение процесса после окончания исполнения команды в подоболочке).

Особенности и трудные к пониманию детали

1) Бесполезное использование cat

cat 1.txt | grep -i '^Begin'

Не используйте cat, когда команда умеет работать сама с файлами!

Правильно:

grep -i '^Begin' 1.txt

или

grep -i '^Begin' < 1.txt

или (нетрадиционный синтаксис)

< 1.txt grep -i '^Begin'

2) Особенности переадресации ввода/вывода

Оператор «<>» переадресации работает только с одним файлом, указанным непосредственно рядом с ним.

Пример:

echo "kl" > 1.txt 2.txt

Результат: будет создан файл 1.txt с содержимым «kl 2.txt».

Дополнительная литература к лекции 3:

Береснев А.Л. Администрирование GNU/Linux с нуля. Гл. 4. Текстовые файлы и каталоги /2-е изд., перераб. и доп. СПб.: БХВ-Петербург, 2010. С. 62–86.

Лекция 4. Сценарии командной оболочки bash

Сценарий командной оболочки (скрипт) – программа, выполняемая командной оболочкой. Состоит из отдельных команд, объединенных общей целью, которые выполняются в командной оболочке последовательно.

Скрипты позволяют автоматизировать часто повторяемые действия, сильно сэкономить время при администрировании ОС. Стандартное расширение файлов скриптов командной оболочки Linux – .sh.

Можно указывать в команде явным образом интерпретатор для скрипта (**bash download.sh**), а можно использовать неявный вызов (**./script.sh**). При неявном вызове нужно 1) обязательно установить разрешение для файла на исполнение (**chmod +x script.sh**); 2) использовать конструкцию «she-bang», т.е. два символа «**#!**», в первой строке скрипта, за

которыми следует название интерпретатора. Последнее приведет к тому, что скрипт выполнится с использованием указанного интерпретатора. Пример содержимого скрипта script.sh (Octave – свободная программа для математических вычислений, использующая синтаксис языка Matlab. Является интерпретатором команд):

#!/bin/octave

2+5 % выведет ans = 7

Переменные в bash – это переменные оболочки, о которых уже говорилось в лекции 2. Для ввода переменной, содержащей пробелы, необходимо использовать кавычки или знак «обратный слэш» – «\». Последний прием называют экранированием текста (в данном случае – пробелов). **Экранирование** применяют для ввода текста, содержащего любые специальные символы: знаки табуляции, шаблоны подстановки, такие как «*» и др. Заэкранированные символы теряют свой особый смысл и интерпретируются как обычные символы:

echo "\"Отель * - прочел Штирлиц.\""**

Bash чувствителен к пробелам при установке переменных и выполнении некоторых операций. Неверные варианты установки переменных (выдают ошибку):

var_a= "Hello World"

var_a = "Hello World"

var_a ="Hello World"

Правильный вариант – без пробелов возле знака «равно»:

var_a="Hello World"

При обращении к несуществующей переменной будет выдана в качестве результата пустая строка. Чтобы предотвратить использование неинициализированных переменных и возможные ошибки используйте команду

set –u

• Для обращения к значению переменной указывайте перед именем «**$**».

• Для вывода количества символов в значении используйте конструкцию

${#имя_переменной}.

• Для уничтожения переменной используйте команду **unset**.

• **Переменные не типизированы!** Рассматриваются как строки, над которыми можно выполнять арифметические операции, если все элементы строки являются числами.

Экранирование имени переменной используется для совмещения вывода значения переменной и другого текста. Имеется два варианта синтаксиса:

echo ${f1}1

str1

$ echo "$f1"1

str1

Полезные встроенные команды

Переменные в сценариях bash интерпретируются как строки, но можно указать оболочке, к какому типу отнести конкретную переменную, как ее рассматривать. Встроенная функция **declare** для объявления типизированных переменных имеет опции

-a – массив;

-i – целое число;

-f – функция (без аргументов список функций);

-r – только для чтения;

-x – переменная на экспорт.

Выполнение следующего кода

declare -r f1

f1=56

приведет к выводу

bash: f1: доступная только на чтение переменная

При типизировании переменных следует четко придерживаться выбранных типов во избежание ошибок:

declare -i var

var="ax"

echo $var # выводит 0

printf – печать строки с форматированием.

Пример:

declare -r PI=3.14159265358979 # переменная только для чтения

printf "Число Пи с 2 десятичными знаками = %1.2f" $PI

eval – выполняет соединение всех аргументов в строку и ее выполнение. Никогда не используйте eval в том месте, где как аргумент может быть подставлен чужой код/ввод пользователя (небезопасно). Пример правильного использования:

c1="ps"

c2="ax"

eval "$c1" "$c2" # выводит список процессов

Чтение переменных

Встроенная утилита **read** позволяет считать значение из стандартного потока ввода

echo -n 'Введите строку:'

read var1

echo $var

Ввести значения трех переменных:

read var1 var2 var3

Если введено больше значений, все значения с третьего и далее будут сохранены в var3. Если введено меньше значений, оставшиеся переменные будут пустыми.

Возможно установить значение по умолчанию для переменной. Синтаксис

${имя_переменной:=значение_по_умолчанию}

Пример

echo ${V2:=12345}

12345

Если переменная уже определена, то установка значения по умолчанию не повлияет на ее значение.

При передаче результатов исполнения команд через конвейер «|», нужно помнить, что каждая команда (кроме последней) запускается в дочерней подоболочке, а присвоение переменных в дочерних оболочках не отражается на значениях переменных родительской оболочки:

i=5; i=1 | echo $i – печатает «5».

Массивы объявляются путем перечисления или присвоения элементов. Можно пропускать некоторые элементы, и они останутся неинициализированными. Элементы массива могут быть инициализированы с помощью квадратных скобок [xx]:

MYMAP[baz]=foo

Можно также объявить массив array следующим образом

declare -a array

Для разыменования (получения содержимого) элемента массива используйте фигурные скобки, то есть

${array[xx]}

Можно использовать следующие специальные обозначения:

${arr[*]} # Все элементы массива

${arr[@]} # аналог ${arr[*]}

${!arr[*]} # Все индексы в массиве

${#arr[*]} # Количество элементов в массиве

${#arr[0]} # Длина нулевой точки

Позиционные параметры представляют собой серию специальных переменных (от $0 до $9), которые содержат содержимое командной строки.

Для команды

some_program.sh word1 word2 word3

позиционные параметры будут содержать следующее: $0 будет содержать название программы «some_program», $1 будет содержать слово «word1», $2 будет содержать слово «word2», $3 будет содержать слово «word3».

Вопрос: что выведет следующий скрипт **some_program.sh**

```
#!/bin/bash
echo "Позиционные параметры"
echo '$ 0 =' $0
```

при его вызове следующим образом

./some_program.sh word1

из командной оболочки?

Другие специальные параметры bash:

${10} ... ${N} – элементы списка аргументов за пределами 9 (обратите внимание на синтаксис);

$*, $@ – строка, составленная из значений всех аргументов командной строки;

"$@" – символы в двойных кавычках будут считываться как один параметр;

"$*" – все символы будут считаны как один параметр, где стандартный разделитель можно заменить на первый символ из переменной IFS;

$# – количество аргументов командной строки;

$? – код возврата предыдущей команды по POSIX; (или 128 + код сигнала, который привел команду к остановке выполнения);

$$ – идентификатор процесса (PID) оболочки.

Знак «@» можно использовать вместо «*» в конструкциях с массивами, таких как **${arr[*]}**, и результат будет аналогичным, за исключением случаев, когда вокруг них есть двойные кавычки. В этом случае поведение такое же, как и в обрамленных кавычками строках: **${arr[*]}** возвращает все элементы как одно слово, в то время как **${arr[@]}** возвращает каждый элемент как отдельное слово.

Встроенная команда сдвига **shift** используется для изменения значений позиционных параметров. Команда, вызванная без параметров, сдвигает аргументы на один влево. Команда может принимать число в качестве аргумента, а именно, количество позиций, на которое осуществляется сдвиг.

shift 4

Данная команда сдвигает $ 5 до $ 1. Первые четыре прежних позиционных параметра при этом необратимо теряются.

Ниже приведен пример скрипта param.sh, демонстрирующего позиционный сдвиг

```
$ cat param.sh
#!/bin/bash
echo $1
echo $2
```

echo shifted:

shift

echo $1 echo $2

Пример работы данного скрипта

$ ./param.sh first second third

first second

shifted:

second third

Для установки значений позиционных параметров непосредственно внутри скрипта (а не при его вызове) можно использовать команду set:

set value1 value2 value3 – установить значения трех первых позиционных параметров.

Проверка условий

Команда test используется для проверки выражений (где EXPRESSION – выражение)

test <EXPRESSION>

Эта команда устанавливает код выхода $? в 0 (истина, true), или 1 (ложь, false) в зависимости от истинности проверяемого условия. Используя этот код, можно позволить Bash выполнять различные действия в зависимости от его значения с помощью условного оператора if.

Существует синоним команды test: команда «**[**».

Опции команды для файлов:

-e – файл существует;

-f – обычный файл;

-d – директория;

-h, -L – является символической ссылкой;

-s – не пустой;

-r – доступен для чтения;

-w – доступен для записи;

-x – доступен для исполнения;

-N – был изменен со времени последнего прочтения.

Проверка существования файла /etc/passwd:

if [-e /etc/passwd]

Проверка со строками

-z <STRING> – истинно, если <STRING> – пустая.

-n <STRING> – истинно, если <STRING> не пуст (это операция по умолчанию).

<STRING1> = <STRING2> – истинно, если строки равны.

<STRING1>! = <STRING2> – истинно, если строки не равны.

<STRING1> < <STRING2> – истинно, если <STRING1> появляется до <STRING2> при сортировке. **Внутри [ ] использовать \< !**

<STRING1> > <STRING2> – истинно, если <STRING1> появляется после <STRING2> при сортировке. **Внутри [ ] использовать \> !**

Сравнение целых чисел

<INTEGER1> -eq <INTEGER2> – истинно, если целые числа равны.

<INTEGER1> -ne <INTEGER2> – истинно, если целые числа НЕ равны.

<INTEGER1> -le <INTEGER2> – истинно, если первое целое число меньше или равно второму.

<INTEGER1> -ge <INTEGER2> – истинно, если первое целое число больше или равно второму.

<INTEGER1> -lt <INTEGER2> – истинно, если первое целое число меньше второго.

<INTEGER1> -gt <INTEGER2> – истинно, если первое целое число больше второго.

Другие варианты сравнения с использованием test

<TEST1> -a <TEST2> – истинно, если <TEST1> и <TEST2> являются истинными (AND). Обратите внимание, что -a также может использоваться в качестве проверки файлов.

<TEST1> -o <TEST2> – истинно, если либо <TEST1>, либо <TEST2> истинно (OR).

! <TEST> – истинно, если <TEST> - false (NOT).

(<TEST>) – группировка для установления приоритета выполнения операций. Внимание: при обычном использовании внутри [ ] «(» и «)» должны быть заэкранированы, используйте «\(» и «\)»!

-o <OPTION_NAME> – истинно, если опция оболочки <OPTION_NAME> установлена.

Пример скрипта для сравнения двух строк [5]

```
#!/bin/bash
    S1='string'
    S2='String'
    if [ $S1=$S2 ];
    then
        echo "S1('$S1') is not equal to S2('$S2')"
    fi
    if [ $S1=$S1 ];
    then
        echo "S1('$S1') is equal to S1('$S1')"
    fi
```

Перед написанием полезно продумать логическую структуру программы.

Запишем для примера **алгоритм вычисления положительной степени целого числа**.

1. Введенная степень и число являются целыми, а степень положительной?

– Если «да»: создать временную переменную и поместить в нее число.

– Если «нет», перейдите к шагу 6.

2. Умножить временную переменную на число, и сохранить результат во временную переменную.

3. Вычесть единицу из степени.

4. Степень больше 1?

– Если «да»: перейдите к шагу 2.

– Если «нет»: перейдите к шагу 5.

5. Вывести число на экран.

6. Выйти из программы.

Дополнительная литература к лекции 4:

BASH Programming – Introduction HOW-TO [Электронный документ]. Режим доступа http://tldp.org/HOWTO/Bash-Prog-Intro-HOWTO-11.html

Лекция 5. Программирование для bash. Расширенные возможности командной оболочки bash

В bash присутствует большое число специальных **операторов со скобками**. Установим их значение.

[– синоним test

[[– улучшенный по сравнению с test оператор в bash.

[[позволяет:

1) использовать && и || для группировки условий (вместо -a и -o);

2) использовать регулярные выражения.

Пример: оператор выбора по шаблону =~ и захват () символов

if [[$answer =~ ^y(es)?$]]

В результате в переменной ${BASH_REMATCH[1]} находится «es».

3) не комментировать названия переменных

if [[-f $file]]

Обычно [[используют для строк и файлов.

Для сравнения чисел используют (( )) (позволяет производить вычисления в стиле C):

if ((a > 5)); then echo "a is more than 5"; fi – не возвращает результат.

$(( )) – арифметическая подстановка – возвращает результат как текст. Используется для арифметических целочисленных вычислений.

Пример:

```
a="$(( 3 * ( 2 + 1 ) ))"
echo $a
9
```

Команда let может быть использована для обновления переменных:

```
let a="3 * (2 + 1)"
echo $a
9
```

Для обновления переменных можно также использовать и более удобный оператор **(( ))** (нижеприведенные команды эквивалентны):

```
(( a += 1 ))
(( a ++ ))
let a=a+1
```

let "a += 1"

Группировка команд возможна с помощью () или {}. Команды в фигурных скобках выполняются без запуска подпроцесса (подоболочки), а команды в круглых скобках выполняются в собственной подоболочке.

true && (echo a; sleep 1 && exit 0 || echo b) && echo c

В данном случае команда exit прервала выполнение команд в скобках, но не скрипта в целом. Поэтому, «echo c» срабатывает. Результат выполнения:

a

c

Другой пример:

true && {echo a; sleep 1 && exit 0 || echo b} && echo c

Команда exit прервала выполнение всей строки команд. Результат выполнения:

a

Операторы сравнения для строк и для чисел, изученные на прошлой лекции, не взаимозаменяемы. Пример:

$ [[2 -lt 10]] && echo less || echo not

less

$ [[2 < 10]] && echo less || echo not

not

Строка «2» при сортировке оказывается после строки «10», поэтому при сравнении строк «2» > «10».

Примеры операции возведения в степень:

echo $((5 ** 2)) #25

echo "5^2" | bc #25

Получение остатка от деления выполняется с использованием оператора «%»

echo $((5 % 2)) #1

В bash можно использовать и тернарный оператор

((t = 5<45?7:11)) # t=7

Подстановка параметров (Parameter expansion) – это процедура получения значения от объекта. При подстановке можно выполнять обработку значения. Выполняется с использованием конструкции ${…}

Мы уже ранее встречались с некоторыми примерами подстановки параметров. К примеру, ${#var} позволяет получить длину строки.

Для массивов ${#array[*]} и ${#array[@]} позволяет получить количество элементов в массиве.

${var#Pattern}**, **${var##Pattern} – произвести перед выводом значения переменной var удаление шаблона Pattern из значения. # – удаляется наименьшая подстрока, ## – жадный вариант команды (т.е., удаляется наибольшая подстрока).

${var%Pattern}, ${var%%Pattern} –аналогично предыдущему, но поиск производится с конца строки var.

${var:pos} – подстановка значения переменной var, начиная с позиции pos.

${var:pos:count} – подстановка count символов строки var, начиная с позиции pos.

${var/Pattern/Replacement} – замена первого вхождения шаблона Pattern на Replacement.

${var/Pattern/} – удаление первого вхождения шаблона Pattern.

${var//Pattern/Replacement} – замена всех вхождений шаблона Pattern на Replacement.

${var/#Pattern/Replacement} – замена вхождения шаблона Pattern на Replacement, при условии что Pattern расположен с начала строки var.

${var/%Pattern/Replacement} – замена вхождения шаблона Pattern на Replacement, при условии что Pattern расположен в конце строки var.

${!prefix*}** или **${!prefix@} – поиск всех объявленных переменных, имя которых начинается с prefix.

Условное выполнение команд

command1 && command2

command2 выполняется, если и только если command1 успешно завершен

command3 || command4

command4 выполняется, если и только если команда command3 не работает

Синтаксис условного оператора if

if COMMANDS; then COMMANDS; [elif COMMANDS; then COMMANDS;]

… [else COMMANDS;] fi

Точка с запятой нужна только в том случае, если на одной строке расположено более одного ключевого элемента конструкции **if**.

Многострочный синтаксис без точки с запятой:

```
if COMMANDS
then
   COMMANDS
elif COMMANDS
then
   COMMANDS
else
  COMMANDS
fi
```

Циклы

Цикл for используют для перебора значений из списка

```
for <имя> in <список>
do
команды, которые используют переменную $<имя>
done
```

Пример:

```
#!/bin/bash
# пример цикла по множеству значений
for A in раз два три четыре пять
do
echo "$A,"
done
echo "вышел заяц погулять"
```

Синтаксис for без аргументов приводит к обработке всех аргументов командной строки, включая пробелы. Попробуйте вызвать представленный ниже скрипт с аргументами и без:

```
#!/bin/bash
for a
do echo -n "$a "  done
echo
exit 0
```

Перебор всех аргументов скрипта:

```
for i in $@
```

Формирование последовательностей чисел для итерирования

В цикле for можно использовать командную подстановку

for i in `seq 0 5 15` # задает цикл от 0 до 15 с инкрементом 5

а также оператор (( ))

for i in ((i=0; i<15; i++)) # задает цикл от 0 до 15 с инкрементом 5

Последовательности чисел в самом bash формируются следующим образом:

for i in {0..15..5} # задает цикл от 0 до 15 с инкрементом 5

Пример: копирование файлов в каталог

```
#!/bin/bash
[ $# -lt 1 ] && exit 1
i=0
for FILE
do
  ((i++))
  if ((i == $#)); then break; fi
  echo "Copying $FILE"
  cp $FILE "${@: -1}"/
done
```

В примере показано, что внутри двойных скобок (( )) используют С-стиль операторов.

В скриптах bash можно использовать **вложенные циклы**:

```
for ((i = 0; i < a; i++)); do
  for ((j = i; j < a; j++)); do
    echo "$i $j"
  done
```

done

Цикл while выполняется, пока условие истинно

while [условие]

do

команды

done

Пример цикла while по переменной I:

```
I=0
while [ $I -lt 15 ];  do
printf "0x%02x " "$I"
((I++))
done
```

Цикл until выполняется, пока условие истинно

until [условие]

do

команды

done

Оператор **until** проверяет условие завершения цикла до итерации (отличие от Pascal):

```
#!/bin/bash
until [ "$var1" = end ];  do
 echo "Введите значение переменной #1 "
 echo "(end – выход)"
 read var1
 echo "значение переменной #1 = $var1"
done
```

Команда case проверяет значение заданной переменной на совпадение с шаблонами. **Команда select** позволяет вывести пользователю контекстное меню выбора опций, где каждая опция получает свой номер. Пример применения **case** для добавления возможности различной реакции программы на выбор пользователя:

```
#!/bin/bash
echo "Нажмите любую клавишу, а затем Enter "
```

```
read Key
case "$Key" in
[[:alpha:]] ) echo "Это буква";;
[[:digit:]] ) echo "Цифра";;
* ) echo "Другой символ";;
esac
```

Заметим, что в примере используются классы символов **[[:alpha:]]** (буква), **[[:digit:]]** (цифра). Классы по стандарту POSIX доступны в bash [7].

Пример использования оператора select для создания меню

```
PS3="Выберите терминал"
select term in tty tty2 tty3; do
    print TERM is $term
    break
done
```

При запуске кода на экран появится следующий текст меню выбора:

```
1) tty1
2) tty2
3) tty3
Выберите терминал
```

Содержимое переменной PS3 всегда выводится после списка вариантов.

Функции в bash

Функции bash можно объявлять следующим образом:

1) с ключевым словом function

```
function имя
{
команды
}
```

2) без ключевого слова, но с круглыми скобками после имени функции

```
имя () {
команды
}
```

Заметим, положение открывающей фигурной скобки не влияет на результат.

3) комбинируя два подхода

function имя ()

{

команды

}

Нельзя: вызывать функцию до объявления.

В следующем примере bash расценит mat как операцию вызова функции, что приведет к ошибке (необъявленная функция), если функция mat ранее не была объявлена.

mat

{

}

Нельзя: давать функциям имена, совпадающие с именами переменных.

Можно: описать функцию в отдельном файле и воспользоваться inline-подстановкой кода скрипта.

Для передачи параметров в функцию укажите все параметры через пробел после ее названия в строке вызова:

func 1 "2 3" # два параметра за счет кавычек

Удобно работать с принятыми параметрами, перебирая их в следующем цикле for:

for var in "$@“

do

команды, действия…

done

Пример: использование аргументов

```
#!/bin/bash
mat() {
a=$1
b=$2
c=$(( a + b ))
echo $c
```

}

mat $1 $2

Результат:

$ ./mat.sh 1 1

2

Here document – это специальный блок кода, который позволяет занести в переменную (или подать на вход команде) многострочный текст. Общий синтаксис here document (сокращенно – heredoc) таков:

команда(-ы) <<ограничитель

строки текста/команды

ограничитель

В качестве ограничителя может быть использована любая строковая переменная.

Пример:

cat <<End-of-message

This is line 1.

This is line 2.

End-of-message

Вывод:

This is line 1.

This is line 2.

Если использовать оператор переадресации вывода, содержащий на конце знак минус, **<<-**, то в тексте heredoc будут подавляться начальные знаки табуляции «TAB» в каждой строке [6].

Внутри here document можно использовать параметры, объявленные вне него.

Here string – это однострочный here document. Пример кода, демонстрирующего работу с синтаксисом Here string и получением массива (опция -a команды read) из строки:

String="Love Linux"

read -r -a Words <<< "$String"

echo "First word of String is: ${Words[0]}" # Love

echo "Second word of String is: ${Words[1]}" # Linux

Подстановка процесса – аналог командной подстановки, используется для передачи результатов исполнения одной команды другой команде. Синтаксис

>(command)

<(command)

Следующая команда выполнит то же самое, что и **ls -l | cat:**

cat <(ls -l)

Дополнительная литература к лекции 5:

Cooper M. Advanced Bash-Scripting Guide [Электронный ресурс]. 2014. Режим доступа http://tldp.org/LDP/abs/html/index.html

Шилдс Я. Функции сравнения и тестирования в Bash [Электронный ресурс]. Режим доступа https://www.ibm.com/developerworks/ru/library/l-bash-test/index.html

Bash Guide for Beginners. Chapter 4. Regular expressions. 4.3. Pattern matching using Bash features [Электронный ресурс]. Режим доступа http://tldp.org/LDP/Bash-Beginners-Guide/html/sect_04_03.html

Лекция 6. Управление правами и пользователями

DAC (Discretionary Access Control) – механизм избирательного управления доступом к файлам, реализован в UNIX (пользователи получают доступ к файлам в соответствии со списком доступа). В соответствии с DAC, у каждого файла есть владелец и группа пользователей, которой принадлежит файл. Каждый пользователь имеет уникальный идентификатор (**UID**) и может принадлежать различным группам. При создании файла пользователем группа пользователей файла совпадает с первичной группой самого пользователя (**GID**).

Команда **id** выводит информацию об идентификаторах пользователя:

$id

uid=1000(user) gid=1000(user) группы=1000(user),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),113(lpadmin)

Первичный ID владельца и группа владельцев записываются в метаданных файла

$ ls -l file

-rw-rw-r-- 1 user user 0 мар 01 12:51 file

Созданный пользователем user (столбец 3) файл имеет группу пользователей user (столбец 4).

Права доступа к файлам являются ключевым элементам понимания механизма избирательного управления доступа к файлам в UNIX. Существует **три класса прав**:

- – User access (u) – права доступа владельца файла;
- – Group access (g) – права доступа группы владельца файла;
- – Other access (o) – права доступа для всех остальных.

Права можно записывать в **символической нотации** (рис. 6):

r-- – наличие разрешения на чтение данного файла;

-w- – наличие разрешения на изменение;

--x – наличие разрешения на исполнение,

и в **восьмеричной нотации** (так называется, поскольку каждый класс прав может содержать три элемента, право на чтение, право на запись и право на исполнение, которые, записанные в двоичном формате друг за другом, образуют трехзначное двоичное число, принимающее одно из восьми возможных значений).

Итак, в восьмеричной нотации мы имеем число из трех бит (1/0 = есть/нет). Наличие всех трех прав в классе (rwx) соответствует числу 7 (рис. 7), поскольку $111_2 = 7_{10}$. Пример: права доступа rxwxr-x--x (в восьмеричной нотации, 751) – пользователь файла имеет все права на доступ к нему (rxw или 7), группа пользователей имеет права на чтение и исполнение файла (r-x или 5), остальные имеют права на исполнение (--x или 1).

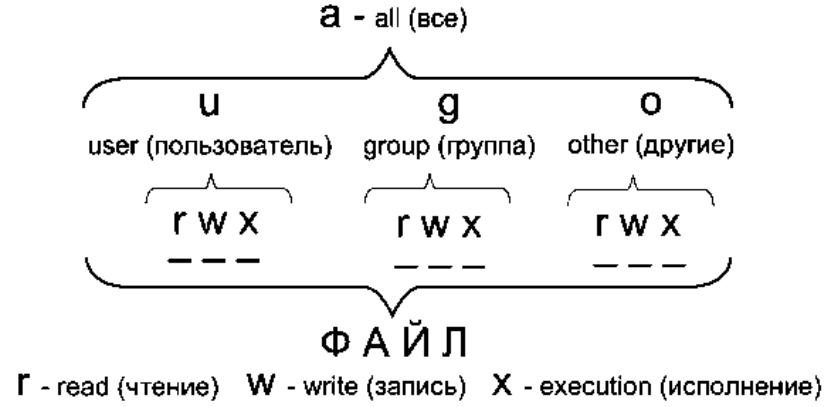


Рис. 6. Права доступа на файл в символической нотации

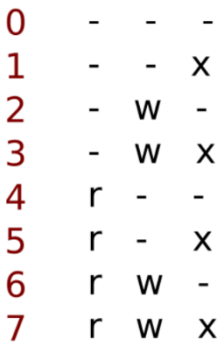


Рис. 7. Соответствие прав доступа на файл в восьмеричной и символической нотации

Права доступа к каталогам имеют более сложное значение. **r** – есть право **на чтение названий файлов** (вывод содержимого каталогов). Для чтения содержимого **требует права x**. **w – право изменять** (в т.ч. – удалять, создавать) файлы в каталоге, опять же, **требует права x**. Таким образом, получаем, что базовым правом для каталогов является **x – право на вход в каталог и доступа** к содержащимся внутри каталога файлам, а также **чтение индексных записей** (метаданных). Поэтому, правильные права для пользователей/групп на каталог должны быть **нечетными, также могут отсутствовать (000)**.

Внимание: право **записи на каталог автоматически дает право на удаление, создание** файлов в нем, невзирая на права на отдельные файлы. Однако для **редактирования** этих файлов потребуются права **на них**.

Часто используемые права для каталогов:

0 (---) – означает отсутствие прав (часто используется для запрещения доступа);

5 (r-x) – доступ на переход в каталог и чтение содержимого без возможности редактирования (часто используется);

7 (rwx) – разрешено все (часто используется для предоставления полного доступа).

Другие права используются редко. Например, права 1 (--x) на каталог предполагают возможность чтения содержимого файлов в нем, и перехода в него, без возможности просмотра списка содержимого. Это неудобно, потому редко используется.

Изменение прав владения осуществляется командой **chown**. Данная команда позволяет менять владельца файла или каталога, а также группу

пользователей-владельцев файла. Вызов команды может производить только суперпользователь.

Команда **chgrp** позволяет менять группу пользователей файла. Ее может выполнить владелец или член группы, а также суперпользователь.

Можно менять пользователя и группу одновременно командой chown:

 # ls -l tmpfile

-rw-r--r-- 1 family family 0 2014-05-03 12:33 tmpfile

 # chown root:root tmpfile

 # ls -l tmpfile

-rw-r--r-- 1 root root 0 2014-05-03 12:33 tmpfile

Опция -R позволяет менять права рекурсивно (для всего содержимого директорий). Опция -v позволяет выводить информацию об изменениях.

Когда команду chown применяют к символической ссылке, тогда изменяется владелец и группа файла, на который указывает ссылка. Это – поведение команды chown по умолчанию. Изменить поведение можно, установив опцию -h.

Опция --reference позволяет скопировать права с одного файла на другой:

 # ls -l file

-rw-r--r-- 1 user user 19 2018-01-01 12:09 file

 # ls -l tmpfile

-rw-r--r-- 1 family family 0 2014-05-03 12:33 tmpfile

 # chown --reference=file tmpfile

 # ls -l tmpfile

-rw-r--r-- 1 user user 0 2014-05-03 12:33 tmpfile

Установка прав доступа к файлам осуществляется командой **chmod**. Права можно указать в восьмеричной или в символьной нотации, изменять права могут суперпользователь и владелец файла/каталога.

В символьной нотации:

chmod <*класс изменение права*> <*файлы*>

Класс – один из следующих:

u – доступ владельца;

g – доступ группы владельцев;

o – доступ всех остальных;

a – доступ всех групп пользователей.

Можно применять следующие модификаторы к классу, а после модификатора указывать новые права:

+ – разрешить;

- – запретить;

= – установить.

Примеры:

chmod 770 1.txt && ls -la 1.txt

-rwxrwx--- 1 user user 0 мар 12 02:42 1.txt*

chmod g-w,o+r 1.txt && ls -la 1.txt

-rwxr-xr-- 1 user user 0 мар 12 02:42 1.txt*

chmod ug=rw,o= 1.txt && ll 1.txt

-rw-rw---- 1 user user 0 мар 12 02:42 1.txt

chmod u+x 1.txt && ll 1.txt

-rwxrw---- 1 user user 0 мар 12 02:42 1.txt

Для вновь создаваемых файлов и каталогов права устанавливаются так: из прав 777 для каталогов и 666 для файлов вычитается битовая маска (ее можно вывести на экран командой **umask**). Можно сменить umask:

touch firstfile

ls -l firstfile

-rw-r--r-- 1 root root 0 Dec 1 16:53 firstfile

umask

0022

umask 0002

touch secondfile

ls -l secondfile

-rw-rw-r-- 1 root root 0 Dec 1 16:54 secondfile

В основанных на Debian дистрибутивах маска для всей системы устанавливается в файлах

/etc/pam.d/common-session

/etc/pam.d/common-session-noninteractive

с помощью строки вида

session optional pam_umask.so umask=0002

Можно установить **специальные биты прав доступа** на файлы и каталоги. Они устанавливаются вместо «x» в символической нотации. Бит SUID – s в старшей триаде, SGID – s в средней триаде битов, sticky bit – t в младшей триаде битов (рис. 8). Большая буква означает отсутствие права «x», малая – наличие.

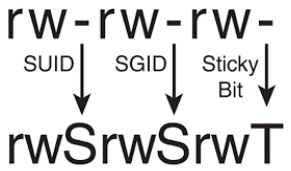


Рис. 8. Специальные биты доступа

Значение специальных битов прав доступа на файлы и каталоги показано в табл. 3.

Таблица 3

Значение специальных битов прав доступа

Права доступа в символической нотации	Результат применения к файлу	Результат применения к каталогу
s****	Исполнение команды от имени владельца файла (изменение эффективного ID пользователя EUID, см. лекцию 8)	-
*****s***	Исполнение команды от имени группы владельцев файла (изменение эффективного ID группы EGID, см. лекцию 8)	На вновь создаваемые файлы в каталоге будет установлена группа владельцев такая же, как у самого каталога
********t	–	Пользователь может удалять и переименовывать в каталоге только те файлы, для которых является владельцем

Действие Sticky bit не распространяется на суперпользователя и на владельца директории.

Специальные биты прав доступа устанавливаются командой chmod в символической нотации (rwsrwsrwt), или, когда к восьмеричной нотации прибавлена еще одна цифра

chmod 1555 – установка sticky bit

chmod 2555 – установка SGID

chmod 4555 – установка SUID

Учетные записи пользователей

Информация о пользователях хранится в файле /etc/passwd, который доступен для чтения всем пользователям. Содержит информацию

• имя пользователя;

• пароль (символ x для теневых паролей);

• UID пользователя;

• GID пользователя;

• справочная информация о пользователе;

• домашний каталог;

• оболочка.

Теневой файл /etc/shadow хранит хэш-суммы паролей пользователей для аутентификации. Использование системы теневых паролей снижает опасность взлома системы. Содержит

• имя пользователя;

• зашифрованный пароль;

• количество дней от начала эпохи UNIX (01.01.70) до момента последней смены пароля;

• минимальное время жизни пароля;

• максимальное время жизни пароля;

• период выдачи предупреждений;

• период до блокировки учетной записи;

• срок жизни учетной записи.

Регистрировать пользователей в системе может суперпользователь.

useradd <username> – регистрация нового пользователя.

useradd -D – выдать информацию о настройках useradd.

Каталог /etc/skel содержит начальное содержимое домашнего каталога, которое копируется в создаваемый домашний каталог пользователя.

Часто используемые опции (начинаются с «-») команды useradd:

• s – файл оболочки по умолчанию;

• d – путь к домашнему каталогу;

• m – необходимо создавать домашний каталог;

• M – не создавать домашний каталог;

• k – путь к альтернативному каталогу скелета;

• u – назначить UID;

• g – назначить GID (первичную группу);

• G – список групп пользователя;

• e – дата блокировки учетной записи;

• f – срок после устаревания пароля до блокировки учетной записи.

Если пользователь не должен входить в сеанс (пользователь-служба), то ему в качестве оболочки можно указать:

• /bin/false – системная команда, возвращающая ошибку;

• /dev/null – специальный файл-поглотитель;

• /sbin/nologin – возвращает код ошибки и сообщение о невозможности входа в сеанс.

usermod <options> <username> – команда предназначена для изменения учетных записей. Опции, в основном, совпадают с опциями команды useradd.

userdel <username> – удаление учетной записи. Для удаления файлов пользователя следует добавить к команде опцию -r: **userdel –r <username>**.

Группы пользователей предназначены для совместного доступа группы лиц к файлам. Информация о группах хранится в файле /etc/group. Формат записи: <имя группы>:<пароль группы>:<GID группы>:<список пользователей>

Временно изменить первичную группу

newgrp имя_группы

Постоянно изменить первичную группу

chgrp имя_группы

Другие полезные команды

groupadd <groupname> – создание группы;

groupdel <groupname> – удаление группы;

gpasswd A <username> <groupname> – назначение администратора группы;

gpasswd a <username> <groupname> – добавление пользователя к группе;

gpasswd d <username> <groupname> – удаление пользователя из группы (администратором группы);

gpasswd <password> – задать пароль на вход в группу для других пользователей, не входящих в группу.

Профили пользователей (т. е. специфичные для них настройки и команды, выполняемых при их входе в систему) хранятся в специальных файлах (рис. 9). Последовательность исполнения файлов профиля при входе в сеанс:

1. /etc/profile;
2. ~/.bash_profile;
3. ~/.bashrc;
4. /etc/bashrc.

При запуске оболочки из командной строки выполняются файлы 3 и 4.

Важные переменные окружения (следует устанавливать в файлах профиля):

- PATH – путь поиска исполняемых файлов;
- TERM – тип терминала;
- USER – имя пользователя;
- HOME – путь к домашнему каталогу.

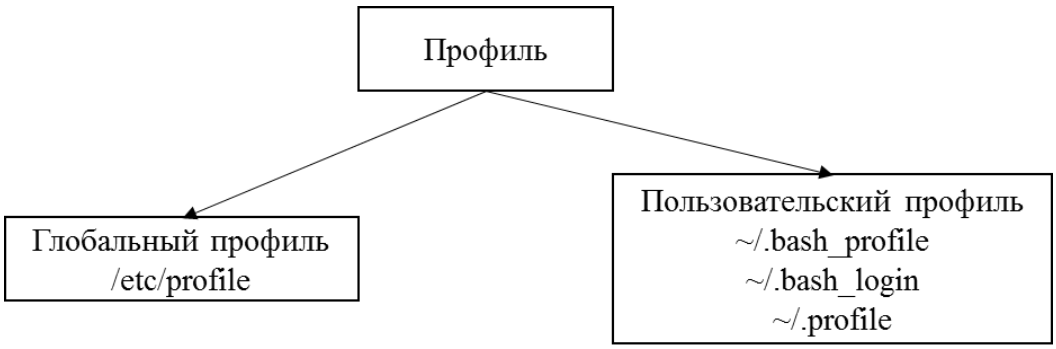


Рис. 9. Файлы профиля пользователей

Переменные окружения могут быть установлены (экспортированы) из файлов профиля командой **export**. Пример:

PATH = $PATH:$HOME/bin

export PATH

Команда **source <profile_file>** – выполнить в текущей оболочке файл профиля.

Наблюдение за активностью пользователей можно производить командой **who**. Опции команды:

-b – время последней загрузки системы;

-H – печать заголовка;

--login – информация о системных процессах, контролирующих вход в сеанс;

-q – имена всех пользователей в системе и их количество;

-u – подробная информация о сеансах;

-w – текущий статус всех сеансов;

-a – полная информация о статусе процессов, контролирующих вход в сеанс.

Информация хранится в файле /var/run/utmp. Информация об открытых и законченных сеансах хранится в файле /var/log/wtmp (команда **last**), о последних логинах – в файле /var/log/lastlog (команда **lastlog**).

Лекция 7. Основы администрирования Windows

Кратко в данной лекции осветим вопросы администрирования Windows, поскольку Windows – наиболее популярная система для персональных компьютеров.

Для администрирования операционной системы Windows нам потребуется запомнить некоторые ее особенности по сравнению с Linux. Основная особенность заключается в том, что при создании каталогов и выводе путей к ним используется обратный слэш для разделения частей пути вместо прямого. Прямой слэш «/» предваряет опции команд (аналог в Linux – это минус «-»).

Для администрирования ОС Windows необходимо изучить основные команды интерпретаторов DOS и Windows, **COMMAND.COM и cmd.exe**.

В операционной системе DOS командный интерпретатор – это первый процесс, выполняющийся после загрузки системы, COMMAND.COM. Он имеет два режима работы:

1) интерактивный режим выполнения команд пользователя;

2) пакетный режим, для выполнения скриптов с расширением .BAT.

После старта интерпретатора он выполняет файл AUTOEXEC.BAT, хранящий настройки системы: переменные среды с установками клавиатуры, звуковой карты, принтера, а также запускающий ряд процессов, таких как драйверы.

Основные команды COMMAND.COM (доступны и в cmd.exe):

имя_диска: (D:, C: и т.д.) – сменить директорию, перейдя на другой диск;

CHCP – отобразить или сменить текущую кодовую страницу (кодировку);

CHDIR, CD – сменить директорию или отобразить имя текущей директории. **CD** (символ можно отделять, а можно не отделять пробелом от названия команды) – перейти в корневую директорию, **CD..** – перейти на уровень выше. Команда толерантна к пробелам (можно не экранировать с помощью объединения строки внутри двойных кавычек). Перейти с одного диска на другой – нужно указать опцию /d: **cd /d C:\Windows**;

CLS – очистить экран;

COPY – копирование файла (MS-DOS спросит подтверждения замены файла, если файл существует). Пример: **COPY file1 file2**;

DEL, ERASE – удаление файла. Также – удаление всех файлов внутри указанной директории. Необходимо подтвердить процесс удаления, введя «Y» и нажав «Enter».

DIR – отобразить содержимое указанной директории (аналог ls из Linux);

ECHO – отображать (ECHO ON) или не отображать (ECHO OFF) команды;

ECHO text – вывести text на экран;

EXIT – выход из командной оболочки;

MKDIR, MD – создать директорию. Позволяет создавать вложенную систему каталогов без применения дополнительных опций;

REN, RENAME – переименовать файл или директорию. Синтаксис **REN старое_имя новое_имя**;

RMDIR, RD – удалить пустую директорию;

SET – установить значение переменной окружения, без аргументов – отображает значения всех переменных среды. Пример: **SET VARIABLE = значение**;

TYPE – вывести содержимое файла на экран;

При написании команд можно использовать относительные и абсолютные пути к файлам и каталогам. Относительный путь – как и в Linux – начинается от текущего каталога, а абсолютный должен начинаться с метки диска (например, C:).

Заметим, что все команды приведены с большой буквы, как они отображаются в системе MS-DOS по умолчанию. Регистр символов не важен в этих системах.

Синтаксис путей в Windows и MS-DOS:

1) Полный путь с указанием имени сервера (uniform naming convention, UNC) \\Server01\user\dir\1.txt

2) Локальный файл C:\user\dir\1.txt

3) Файл в корневом каталоге диска C:\1.txt (Windows) или C:1.txt (MS-DOS).

Запуск программы возможен по ее имени по абсолютному пути или по относительному. Если каталог с программой указан в %PATH%, можно не указывать абсолютный путь к ней.

Управляющие структуры batch-файлов

:label – определяет метку перехода для конструкции GOTO;

CALL – выполнить другой исполняемый файл, по окончании его исполнения – вернуться к исполнению текущего файла. Работает только в скриптах (не в консоли).

Пример:

CALL MyScript.cmd "1234" – вызовет скрипт MyScript.cmd с параметром 1234;

CALL OtherScript.cmd %_MyVariable% – вызовет OtherScript.cmd с параметром-переменной _MyVariable;

FOR – повторить команду для каждого указанного файла;

GOTO – перейти к метке;

IF – условный оператор;

PAUSE – прерывает выполнение программы и отображает сообщение с просьбой нажать любую клавишу для продолжения;

REM – комментарий (текст в строке за этой командой не интерпретируется, второй вариант – использование «**::**»);

SHIFT – сдвиг параметров (аналог одноименной Linux-команды, %0 принимает значение от %1, %1 - от %2, и т.д.).

Получение справки по командам

HELP – вывести список доступных команд. С именем команды дает справку по команде, например, HELP CD. Аналогичный эффект будет давать команда CD/? (или CD /?);

Копирование каталогов

XCOPY – копировать файлы и каталоги. Для копирования каталогов используйте команду **xcopy /s /i исходный_каталог каталог_назначения**. «/s» – скопировать все непустые каталоги рекурсивно; «/i» – создавать каталоги, если они не существуют.

Шаблоны

cmd.exe поддерживает шаблоны: **DEL a*.*b** – удалить все файлы, начинающиеся на букву **a** с расширением, заканчивающимся на букву **b**.

Переменные

Переменные можно подразделить на переменные оболочки и переменные среды. При обращении к переменной оболочки указывают один знак «%» перед ее названием. При вызове из .bat-скрипта, необходимо использовать два знака «%» перед названием переменной. Название переменной среды окружают знаками «%», по одному знаку с обеих сторон, независимо от использования вне скрипта или внутри него.

for /L %n IN (1,2,7) DO @ECHO %n

Разберем данный код. Символ «@» подавляет вывод на экран самого кода команды (ECHO OFF – подавляет вывод всех последующих команд, поэтому, чаще всего первой строкой скрипта является команда @ECHO OFF). Цикл for … in .. do позволяет выполнить многократно тело цикла, для каждого значения переменной-итератора (%n) внутри последовательности чисел от 1 до 7 с шагом 2 (можно писать IN (1 2 7), используя запятые вместо пробелов, с тем же эффектом). Опция /L нужна для генерации

последовательности чисел (не нужна, когда явно указаны все вхождения, например, **FOR %n IN (1 3 5 7) DO …**). Без опции /L код будет выполнен для значений переменной 1, 2 и 7.

При выполнении внутри bat-файла код следует изменить так:

for /L %%n IN (1,2,7) DO @ECHO %%n

Заметим, что имена переменных чувствительны к регистру, в отличие от имен команд: %n и %N – это разные переменные.

В CMD.EXE можно создавать вложенные циклы.

Операции сравнения для переменных окружения внутри тела цикла невозможны напрямую, поскольку оболочка раскрывает (подставляет) значения переменных до выполнения самого цикла. Можно использовать три приема, чтобы обойти это. Покажем их на примере цикла, внутри которого значение переменной %VAR% устанавливается, если оно не установлено ранее.

Первый прием работает в MS-DOS и использует вызов другого исполняемого файла со скриптом:

FOR %%A IN (5 7 9) DO CALL P2.BAT %%A

где P2.BAT содержит

IF "%VAR%"=="" SET VAR=%1

Второй и третий прием работают только в CMD.EXE (Windows). Это – использование специальной опции

SETLOCAL ENABLEDELAYEDEXPANSION

в совокупности с другим синтаксисом обращения к переменной !VAR!, а также использование функции (FOO с одним параметром, создается с помощью меток):

```
FOR %%A IN (5 7 9) DO CALL :FOO %%A
GOTO END
:FOO
IF "%VAR%"=="" SET VAR=%1
EXIT /B 0
:END
```

Как видно из прошлого примера, вызов функций осуществляется через команду CALL с указанием метки и параметров, передаваемых

функции. Для выхода из функции используется команда выхода с кодом успешного завершения 0: **EXIT /B 0**.

Запуск интерпретируемых исполняемых файлов .bat

Создайте файл с расширением .bat, запишите в него свой скрипт, и введите его имя в качестве команды командной оболочки для его выполнения.

Условный оператор

Условный оператор, проверяющий существование файла:

```
IF EXIST "temp.txt" (
   ECHO found
) ELSE (
   ECHO not found
)
```

Круглые скобки позволяют разделить команду на несколько строк.

Сравнение переменной со строкой

```
IF /I "%var%"=="car" (
   ECHO car!
)
```

/I – не учитывать регистр символов.

Арифметическое сравнение

```
SET /A var=1
:: выведет true (равно)
IF "%var%" EQU "1" ECHO true
:: выведет true (не равно)
IF "%var%" NEQ "0" ECHO true
:: выведет true (больше или равно)
IF "%var%" GEQ "1" ECHO true
:: выведет true (меньше, чем)
IF "%var%" LSS "2" ECHO true
```

GTR – больше, чем; **LEQ** – меньше или равно.

NOT – логический оператор «не».

Проверка кода возврата предыдущей команды (аналог $? в bash):

```
IF %ERRORLEVEL% NEQ 0 Echo error found
```

Важные переменные среды:

%PATH% – список путей к каталогам, где находятся исполняемые файлы;

%HOME%, %USERPROFILE% (Microsoft Windows) – расположение домашнего каталога пользователя;

%ALLUSERSPROFILE% – каталог общих профилей всех пользователей;

%APPDATA% – каталог данных приложений;

%COMPUTERNAME% – имя хоста (компьютера);

%ComSpec% – путь к командному интерпретатору по умолчанию;

%HOMEDRIVE% – диск с домашней папкой;

%HOMEPATH% – путь к домашней папке пользователя на диске с домашней папкой;

%ProgramFiles% – каталог установки программ;

%ProgramFiles(x86)% *(только в 64-битных версиях)* – каталог установки 32-разрядных программ;

%SystemDrive% – системный диск (диск, на который установлена операционная система);

%SystemRoot% – директория с Windows;

%TEMP% и %TMP% – каталог для временных файлов;

%USERDOMAIN% – рабочая группа пользователя;

%USERNAME% – имя пользователя;

%windir% – директория, имеющая имя «Windows», на системном диске.

Значения по умолчанию для этих переменных показаны в табл. 4.

Команда chkdsk – проверка диска

Опции команды:

/F – исправить ошибки, обнаруженные на диске;

/V – показывать полный путь к каждому файлу;

/R – обнаруживать и пытаться восстановить поврежденные сектора (*bad-сектора*).

Опции /I и /C позволяют производить менее глубокое, ускоренное сканирование.

Таблица 4

Значения по умолчанию для переменных среды в Windows

Переменная	Значение по умолчанию в Windows XP	Значение по умолчанию в Windows Vista/7/8
%ALLUSERSPROFILE%	C:\Documents and Settings\All Users	C:\ProgramData
%APPDATA%	C:\Documents and Settings\имя_пользователя\ Application Data	C:\Users\имя_пользователя \AppData\Roaming
%COMPUTERNAME%	Имя компьютера	Имя компьютера
%ComSpec%	C:\Windows\System32\cmd.exe	C:\Windows\System32\cmd.exe
%HOMEDRIVE%	C:	C:
%HOMEPATH%	\Documents and Settings\{username}	\Users\{username}
%PATH%	C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;также, пути к программам	C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem; также, пути к программам
%ProgramFiles%	%SystemDrive%\Program Files	%SystemDrive%\Program Files
%ProgramFiles(x86)%	%SystemDrive%\Program Files (x86)	%SystemDrive%\Program Files (x86)
%SystemDrive%	C:	C:
%SystemRoot%	The Windows directory, usually C:\Windows, formerly C:\WINNT	%SystemDrive%\Windows
%TEMP% и %TMP%	%SystemDrive%\Documents and Settings\имя_пользователя\Local Settings\Temp	%SystemRoot%\TEMP, %USERPROFILE%\AppData\Local\Temp
%USERDOMAIN%	Имя рабочей группы	Имя рабочей группы
%USERNAME%	Имя пользователя	Имя пользователя
%windir%	%SystemDrive%\WINDOWS	%SystemDrive%\Windows

chkdsk c: /r – исправление поврежденных секторов на диске C:

Поиск директорий и файлов осуществляется выводом списка файлов и каталогов внутри заданной директории (в примере ниже – текущей) с последующей фильтрацией результатов командой **FIND**:

TREE | FIND "Love Linux"

или

CHKDSK /V | FIND "Love Linux"

Также в современных версиях ОС Windows существует команда **Where – поиск файлов (Windows 2003 и новее)**:

WHERE myfile.doc – поиск файла myfile.doc в текущем каталоге и каталогах, указанных в переменной PATH;

WHERE /T myfile.doc – отобразить также размер, дату и время изменения файла.

WHERE /R C:\ myfile.doc – рекурсивный поиск в корневом каталоге диска C:

Windows PowerShell: командлеты, конвейер, регулярные выражения, работа с файловой системой

Windows PowerShell – командная оболочка с открытым исходным кодом, входящая в состав современных ОС Windows (7, 8, 10). PowerShell может также работать в ОС Linux, UNIX и MacOS.

Командлет (*cmdlet*) – команда, выполняющая одну специфичную функцию в оболочке Windows PowerShell. Названия командлетов формируются из глагола и существительного, разделенных дефисом, и интуитивно понятны.

Get-Help – справка по команде (**Get-Help Service**);

Get-Content C:\1.txt – чтение файла 1.txt (вывод на экран);

Get-Service – информация об установленных службах;

Stop-Service -Name MpsSvc – остановка службы (брандмауэра);

Start-Service -Name MpsSvc – запуск службы;

Get-Process – получить список процессов;

Stop-Process -name WINWORD – остановить процесс WINWORD.

Сравнение команд различных оболочек представлено в табл. 5.

В командлетах можно использовать регулярные выражения. Например, команда

Get-Command *-Service

выведет справку по командам, относящимся к службам.

Очень распространено использование конвейера (|) для передачи вывода одной команды другой команде:

Get-Service | Get-Member

Командлет Get-Service выдает на выход объекты, а не простой текст. Команда Get-Member позволяет получить подробную информацию об поступивших на его вход объектах.

Таблица 5

Краткое описание основных команд в различных командных оболочках [8]

Команда Windows PowerShell (cmdlet)	Команда Windows PowerShell (псевдоним)	cmd.exe, COMMAND.COM	Команда Bash	Описание команды
Get-Location	gl, pwd	cd	pwd	Показать путь к текущей директории
Set-Location	sl, cd, chdir	cd, chdir	cd	Переместиться по дереву директорий
Pop-Location	popd	popd	popd	Изменить текущую директорию на последнюю из стека
Push-Location	pushd	pushd	pushd	Поместить текущую директорию в стек
Copy-Item	cpi, copy, cp	copy	cp	Копировать файлы
Remove-Item	ri, del, erase, rmdir, rd, rm	del, erase, rm dir, rd	rm, rmdir	Удалить файл или директорию
Rename-Item	rni, ren	ren, rename	mv	Переименовать файл/директорию
Move-Item	mi, move, mv	move	mv	Переместить файл/директорию
Get-ChildItem	gci, dir, ls	dir	ls	Получить список файлов или директорий
Get-Content	gc, type, cat	type	cat	Вывести содержимое файла
Write-Output	echo, write	echo	echo	Выводить строки
Set-Variable	sv, set	set	set	Установить значение переменной
Select-String	sls	find, findstr	grep	Поиск строк по шаблону
Get-Process	gps, ps	tlist, tasklist	ps	Вывести список процессов системы
Stop-Process	spps, kill	kill, taskkill	kill	Передать процессу сигнал / остановить процесс
Get-Help	help, man	help	man	Получить справку
Clear-Host	cls, clear	cls	clear	Очистить экран

Более сложный командлет:

Dir C:\Temp s -File | Sort -Property LastWriteTime | Select -Last 1

Dir (сокращение от командлета Get-ChildItem) с параметром -File позволяет получит список файлов в директории (C:\Temp). Далее, список файлов по конвейеру поступает на сортировку по значениям свойства LastWriteTime (время последнего изменения) командлета Sort-Object

(сокращенно – Sort). Первый для команды позиционный параметр -Property можно опускать. Наконец, Select-Object осуществляет выбор последнего объекта в списке (-Last 1) – последнего измененного файла.

Командлет Where-Object осуществляет фильтрацию данных на основе условий:

-9,1,2,3,4,5,6 | Where-Object -FilterScript {$_ -lt 4}

отфильтрует данные, сохранив значения, меньшие 4:

-9

1

2

3

Управление компьютером

Управление компьютером в Windows может осуществляться из окна «Управление компьютером». В меню Пуск следует щелкнуть правой кнопкой на слове «Компьютер» и выбрать «Управление». В дереве объектов можно выбрать «Диспетчер устройств» для открытия специальной программы управления устройствами. В диспетчере устройств для каждого устройства можно переустановить драйвер (правая кнопка мыши – Обновить драйвер), временно отключить устройство или удалить его из системной конфигурации (не использовать), просмотреть подробную информацию (правая кнопка мыши – Свойства). В подробных свойствах на вкладке «Сведения» можно узнать ИД Оборудования. Код экземпляра устройств (ИД оборудования) представляет собой одну из строк:

PCI/VEN_xxxx&DEV_xxxx&SUBSYS_xxxxxx

USB/VID_xxxx&PID_xxxx

ACPI/ATKxxxx

HDAUDIO/FUNC_xx&VEN_xxxx&DEV_xxxx&SUBSYS_xxxxxx

Для устройств, которые не распознаются в системе (помечаются желтым знаком вопроса рядом со значком в дереве устройств), и для которых Windows не удается найти драйвер, необходимо установить драйвер вручную. Для этого полученную строку с ИД оборудования нужно найти в соответствующей онлайн базе (PCI ID Repository https://pci-ids.ucw.cz/, USB ID Database https://usb-ids.gowdy.us/read/UD/, или других).

Таким образом можно получить информацию о производителе и модели данного устройства. Далее, остается найти для данного устройства драйвер, загрузив его, например, с официального сайта производителя.

Запуск и остановка служб осуществляются также из «Управления компьютером», элемент «Службы» в дереве объектов. Нужно щелкнуть на имя службы на правой вкладке правой кнопкой мыши, затем выбрать Запуск, Остановку или Перезапуск службы (рис. 10).

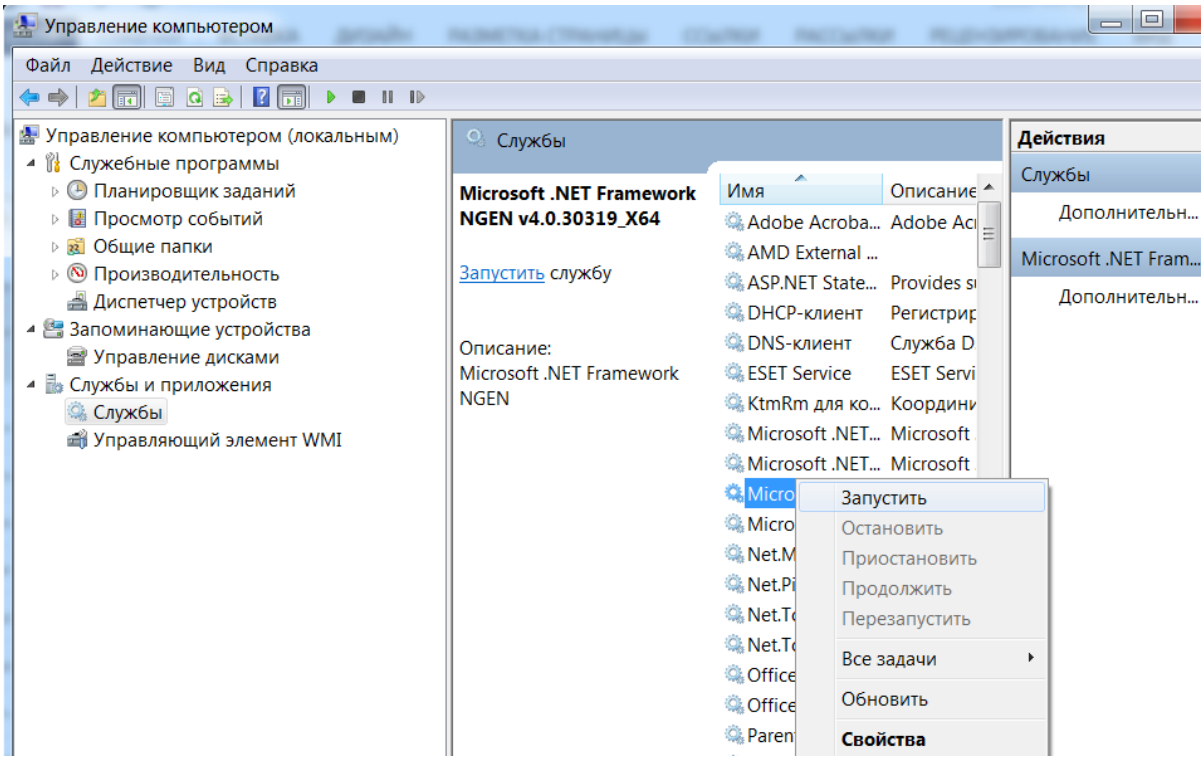


Рис. 10. Программа «Управление компьютером» в Windows 7

Управление дисками (рис. 11) также доступно из окна «Управление компьютером». После выбора данного элемента откроется окно, где по нажатию правой кнопки мыши на разделах диска можно выполнять с ними ряд операций: сжимать данные, форматировать раздел. Заметим, что активный системный диск нельзя форматировать (рис. 11).

Для управления компьютером предназначена и утилита **msconfig**, вызываемой одноименной программой из меню «Пуск». В меню «Пуск» в строке «Найти файлы и папки» следует набрать msconfig, нажать «Enter». Откроется окно, где можно выбрать, какие службы запускать при старте системы по умолчанию (Службы), и какие программы (Параметры автозагрузки, рис. 12).

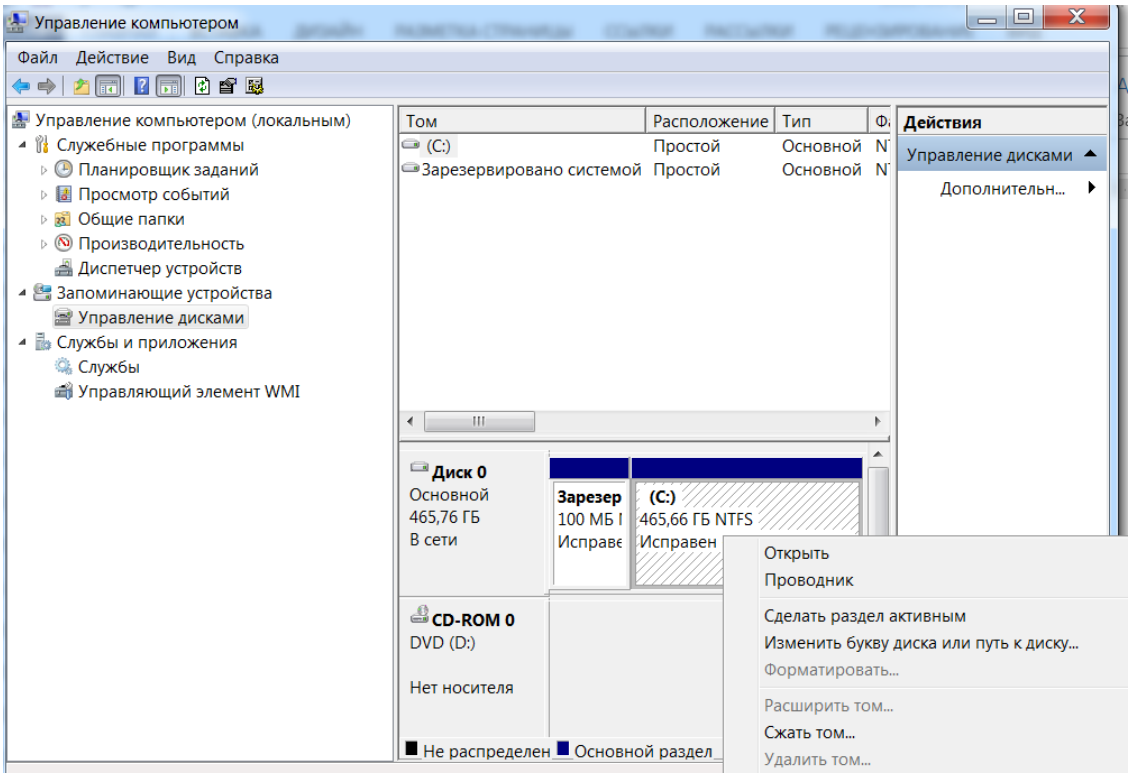


Рис. 11. «Управление дисками» в Windows 7

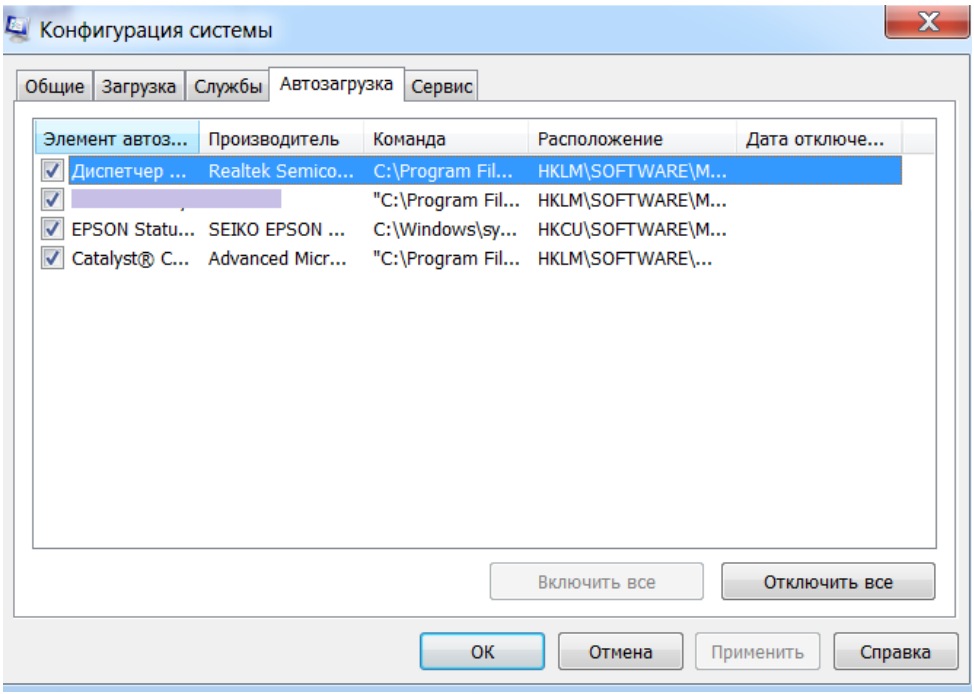


Рис. 12. Просмотр параметров автозагрузки в Windows 7

Полезная утилита **gpedit.msc** (Редактор локальной групповой политики), запускаемая аналогично предыдущей утилите, позволяет отредактировать групповые политики. Например, полезно настроить запрет запуска некоторых программ: надо последовательно выбирать пункты «Конфигурация пользователя», «Административные шаблоны», «Система»,

«Не запускать указанные приложения Windows», «Включить» (рис. 13), «Параметры», «Показать», «Вывод содержания». В последнем окне ввести в строки названия приложений, запуск которых запрещен.

В этом же утилите можно указать сценарии запуска и завершения работы. Окно для выбора файлов сценариев доступно после нажатия на пункты «Сценарии запуск/завершение» в левой части окна (рис. 14) и «Автозагрузка» или «Завершение работы» в правой части.

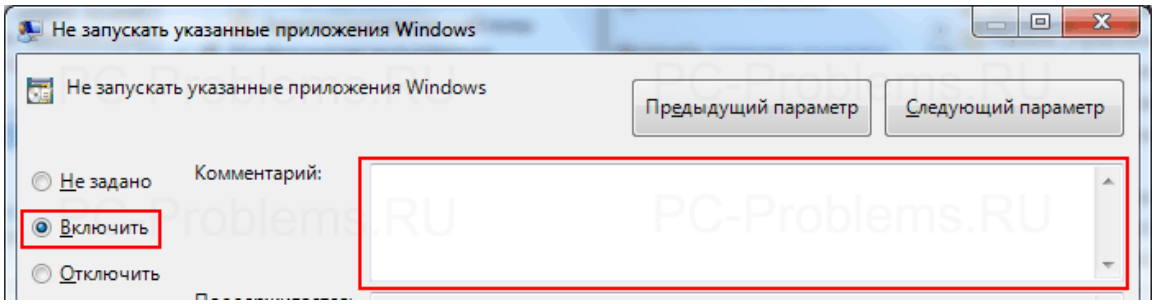


Рис. 13. Запрет запуска некоторых программ в Windows 7

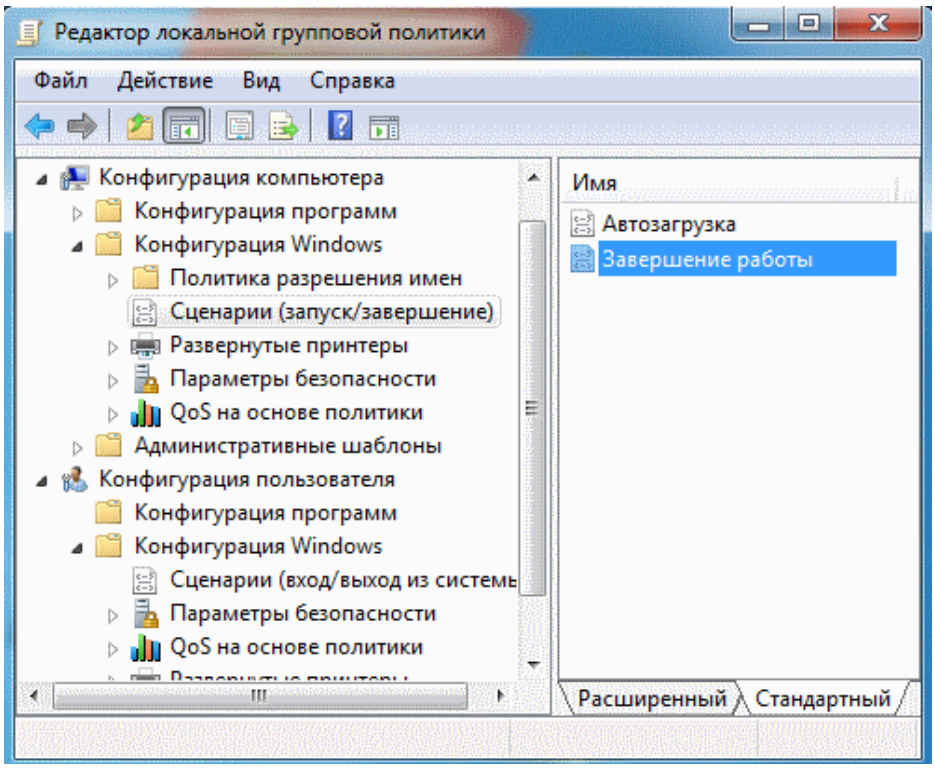


Рис. 14. Сценарии в редакторе локальной групповой политики в Windows 7

Недостаток – утилита **gpedit.msc** доступна только в версиях Windows Professional, Ultimate и Enterprise, но не в домашней версии (Home).

Дополнительная литература к лекции 7:

Lee X. PowerShell as cmd.exe or Bash [Электронный ресурс]. Режим доступа: http://xahlee.info/powershell/commands.html

Глава 2. РАБОТА С ДАННЫМИ И ПРОЦЕССАМИ

Лекция 8. Процессы и сигналы

Абсолютное большинство современных ОС являются многозадачными.

Многозадачность – это одновременное выполнение ряда задач. Каждую из задач принято называть процессом операционной системы. Многозадачность на одноядерных вычислительных устройствах осуществляется за счет механизма переключения контекста – каждый из процессов получает небольшое количество времени процессора, после чего его данные выгружаются из регистров (возможно, кэша, рис. 15), а данные другого процесса загружаются.

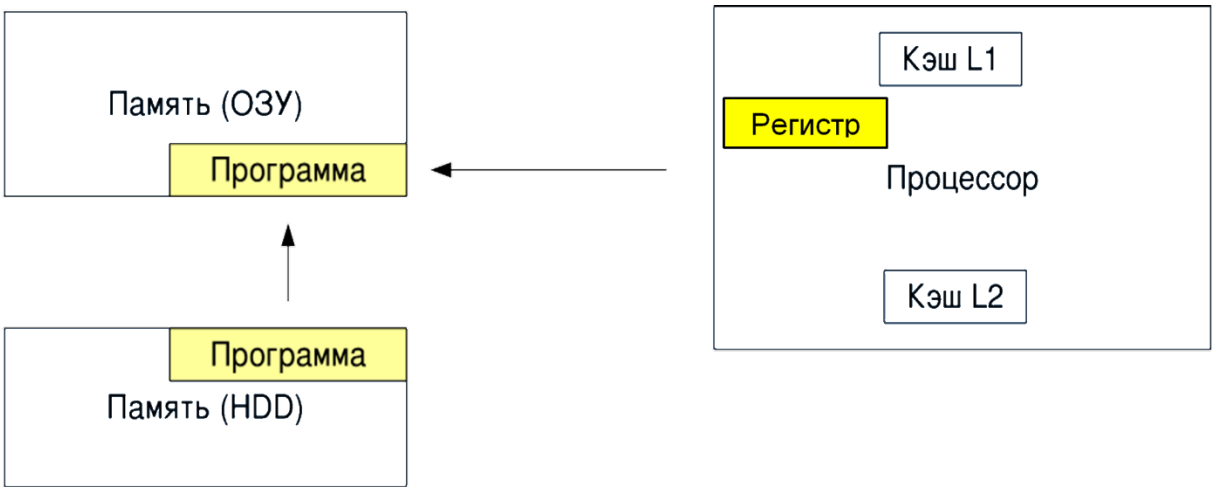


Рис. 15. Виды памяти компьютера

Переключение контекста происходит с фиксированными временными интервалами (упреждающая многозадачность), или по сигналу от программы, разрешающему ее прерывания выполнения (кооперативная многозадачность).

В GNU/Linux многозадачность основана на следующих принципах:

- Используется механизм переключения контекста и упреждающая многозадачность;
- Используется механизм изменения соотношения времени работы процессов за счет назначения приоритетов.
- Процессы с большим приоритетом вытесняют процессы с меньшим приоритетом (вытесняющая многозадачность).

Отличие процесса от программы, задания и задачи. Программа есть файл, предназначенный для исполнения. **Процесс** – экземпляр программы, исполняемый под управлением операционной системы. **Задание** – загруженный в память компьютера набор инструкций. **Задача (job)** – команда, запущенная пользователем (в командной оболочке).

Одна задача может приводить к запуска ряда процессов.

Задачи могут выполняться удаленно, а также ставиться в очередь задач (job queue).

Каждый процесс работает в своем виртуальном адресном пространстве, получить доступ к адресному пространству другого процесса могут только некоторые системные процессы. Пользовательские процессы могут взаимодействовать с ядром посредством механизма системных вызовов.

Ядро Linux – основная программа, которая обеспечивает взаимодействие пользовательских программ и оборудования. Файл ядра в Ubuntu хранится в папке /boot и называется vmlinuz-номер_версии.

Системные вызовы – функции, реализованные в ядре операционной системы. Процессы ядра работают в пространстве ядра (kernel space). Приложения работают в пользовательском пространстве (user space), а системные вызовы выполняются в пространстве ядра. Место ядра и системных вызовов в общей структуре ОС показано на рис. 16.

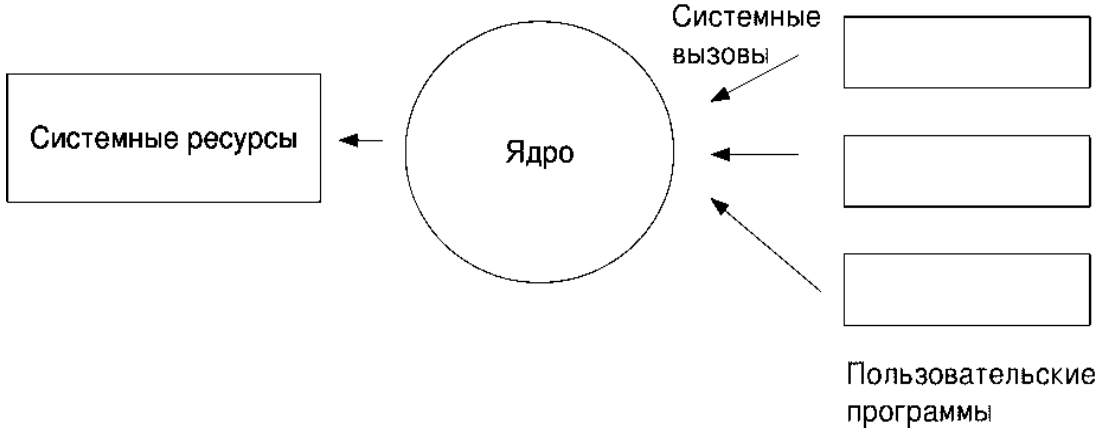


Рис. 16. Ядро и системные вызовы в структуре операционной системы

time <программа> – команда определяет общее время работы программы, время работы на стороне пользователя и на стороне ядра.

Пример работы команды (выводится время полное работы программы ls, время работы на стороне пользователя и на стороне ядра):

$ time ls

real 0m0.103s

user 0m0.052s

sys 0m0.049s

В Linux память, используемую процессом, можно подразделить на виртуальную и реальную (физическую). **Физическая память** – реально используемая процессом часть оперативной памяти. **Виртуальная память** процесса содержит исполняемый код и данные, в том числе, при обработке новых данных могут выделяться новые участки виртуальной памяти, которые связываются с изначальной. При использовании разделяемых библиотек каждый процесс Linux не хранит свою копию библиотеки в оперативной памяти. Вместо этого, код и данные процесса связываются в виртуальных адресных пространствах этого процесса и других процессов, совместно использующих библиотеку [9].

Участок виртуальной памяти процесса копируется в физическую память только тогда, когда процесс пытается ее использовать. Linux создает таблицу страниц памяти процесса, виртуальные области – это отдельные страницы, не загруженные в оперативную память. Когда процесс пытается получить доступ к данным или коду, хранящимся только в виртуальной памяти, а не в физической, то возникает ошибка доступа и срабатывает механизм загрузки страниц виртуальной памяти в физическую.

Образ процесса в оперативной памяти включает в себя:

– Данные пользователя.

– Программу пользователя.

– Системный стек (пространство ядра).

– Блок управления процессом.

Процесс имеет ряд характеристик (**атрибутов**), которые можно просмотреть с помощью команды ps:

− Идентификатор процесса (PID): уникальный номер для обозначения процесса.

− Идентификатор родительского процесса (PPID).

– Пользовательский приоритет (число NICE).

– Терминал (TTY): терминал, к которому подключен процесс.

– ID реального и эффективного пользователя (RUID и EUID, обычно совпадают, отличаются при использовании бита доступа SUID), ID реальной и эффективной группы (RGID и EGID): эффективная группа отличается при использовании бита доступа SGID.

Для каждого процесса создается свой каталог в псевдофайловой системе /proc (рис. 17). Здесь также можно просмотреть атрибуты процесса.

/proc – псевдофайловая система, отражающая дерево процессов в виде дерева директорий, имеющих название, соответствующее PID процесса. Для некоторых процессов ядра можно изменять параметры.

В каждой директории есть файлы:

• cmdline – командная строка процесса;

• status – подробная информация о статусе процесса;

• fd – мониторинг файлов, открытых процессом;

• cwd – ссылка на текущий рабочий каталог процесса;

• environ – окружение.

```
root@01-PK:/tmp# ls /proc
1     1288  1795  2038  2494  2687  3310  6786  acpi         mdstat
10    129   18    2043  2499  2694  3312  6826  asound       meminfo
1085  13    1804  2049  25    27    3318  6847  buddyinfo    misc
11    133   1813  2054  2501  2736  3319  6881  bus          modules
1101  134   1820  2073  2503  276   3320  6888  cgroups      mounts
1117  135   1825  2083  2505  2773  34    7     cmdline      mtrr
1120  1353  1851  2095  2516  2775  3478  7284  consoles     net
1125  136   1866  2097  2517  2780  35    7575  cpuinfo      pagetypeinfo
1128  139   1869  21    2524  2781  3502  7576  crypto       partitions
1140  140   1883  2141  2534  2795  36    7577  devices      sched_debug
1149  141   1894  216   2549  2798  3701  7595  diskstats    schedstat
1151  142   1895  2197  2551  28    3702  7596  dma          scsi
```

Рис. 17. Псевдофайловая система /proc

Механизм порождения процессов – используется системный вызов fork() для дублирования процесса. В дочернем процессе обычно производят системный вызов exec(), чтобы запустить отдельный, независимый процесс, а в родительском – системный вызов wait(), чтобы ждать сигнала завершения от дочернего процесса.

У каждого процесса есть ровно один родитель, но может быть несколько потомков, или дочерних процессов. Поэтому, систему процессов удобно изображать в виде дерева.

Если родительский процесс завершается раньше дочернего, то процесс-сирота (orphaned process) наследуется процессом init (PID = 1). Если дочерний процесс завершается раньше, чем родительский процесс успел вызвать wait(), то дочерний помечается как zombie, буква Z в статусе процесса (или «defunct» в конце имени процесса).

Число процессов в системе ограничено. Узнать максимально поддерживаемое число процессов для запущенной ОС Linux:

cat /proc/sys/kernel/pid_max

По спецификации Linux, 4194304 – предел для архитектуры x86_64, а 32767 – для архитектуры x86. Максимальное значение PID при этом на единицу меньше.

Уникальные идентификаторы процесса

- PID (Process ID) – уникальный порядковый номер процесса;
- PPID (Parent Process ID) – PID родительского процесса, позволяющий выстроить иерархию процессов;
- UID (User ID) – ID пользователя, от имени которого выполняется процесс;
- GID (Group ID) – ID группы пользователей, от имени которой выполняется процесс.

По умолчанию, процесс связывается с терминалом, из которого запущен (pts/2 на рис. 18).

Определить текущий используемый терминал позволяет команда **tty**.

```
root@01-PK:/tmp# ps -l
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S     0  3701  3010  0  80   0 - 14084 poll_s pts/4    00:00:00 sudo
4 S     0  3702  3701  0  80   0 - 13900 wait   pts/4    00:00:00 su
4 S     0  3703  3702  0  80   0 -  5612 wait   pts/4    00:00:00 bash
0 R     0  7630  3703  0  80   0 -  7539 -      pts/4    00:00:00 ps
```

Рис. 18. Список процессов пользователя с основными идентификаторами процесса

Вывод команды **ps -l** (рис. 18) содержит информацию о статусе процесса (S), флагах процесса (F) и других важных атрибутах:

NI (Nice Number) – пользовательский приоритет процесса;

SZ – общее количество памяти, занимаемое процессом;

WCHAN – идентификатор системного вызова.

Различают следующие статусы процесса:

• D – приостановлен и не может быть прерван;

• R – выполняется или находится в очереди;

• S – приостановлен;

• T – трассируется;

• Z – defunct (zombie).

Категории процессов

• Процессы ядра. Выполняются в пространстве ядра. Располагаются в оперативной памяти и не имеют соответствующих программ в виде исполняемых файлов. Соответствующий код находится в модулях ядра. Пример – процесс упорядочивания процессов в очереди (scheduling).

• Фоновые неинтерактивные процессы. Имеют соответствующие исполняемые файлы. Предназначены для обслуживания соединений по какому-либо сетевому протоколу или регулярное выполнение рутинных операций в системе.

• Прикладные процессы. Все остальные процессы, которые порождаются в рамках сеанса работы пользователя.

Процессы можно запускать **в интерактивном режиме и в фоновом режиме**. В интерактивном режиме может работать только один процесс. В фоновом режиме процесс не подключен к потокам ввода-вывода текущего терминала.

./script.sh & – запуск скрипта в фоновом режиме;

<Ctrl>+<Z>, затем **bg <номер задачи>** – перевести задачу из интерактивного режима в фоновый;

fg <номер задачи> – перевести задачу из фонового режима в интерактивный.

Однако при завершении родительского процесса все запущенные из него фоновые задачи завершаются. При необходимости закрытия командной оболочки необходимо выполнять «отвязку» фоновых процессов от родительской командной оболочки.

nohup ./script.sh & – запуск с «отвязкой» команды от родительской оболочки;

Для «отвязки» ранее запущенной команды от оболочки, переведите ее в фоновый режим выполнения, а затем выполните команду

disown -h номер_задания (например, %1 для первого запущенного задания, можно писать просто «1»).

Также для запуска задач на серверах популярна команда screen, которая позволяет не прекращать выполнение программ при закрытии терминала.

Программа Screen

screen – запуск нового терминала Screen;

screen -ls – вывести список запущенных экземпляров screen;

< Ctrl >+< a >+< d > – выйти из screen (screen продолжит работу);

screen -r [номер_процесса] – вновь присоединиться к ранее запущенной сессии screen. Если сессия – единственная, номер процесса можно опустить;

screen -R – вновь присоединиться к ранее запущенной сессии screen, или создать новую сессию, если ни одной сессии не запущено.

screen -d [номер_процесса] – отсоединить сессию screen от терминала. Команду следует исполнять извне запущенной сессии при потере управления терминалом для дальнейшего подключения сессии к другому терминалу и использованию.

screen -d -r [номер_процесса] – отсоединить сессию screen от терминала и вновь присоединить к текущему терминалу

< Ctrl >+< d > или exit – выйти из-под запущенной сессии.

Другие важные команды для управления процессами

jobs – просмотр списка задач.

kill %<номер задания> – завершение работы фонового процесса (задачи).

ps – вывести информацию о процессах, связанных с текущим терминалом. Поля: PID – ID процесса, TTY – терминал, TIME – суммарное процессорное время работы процесса;

ps -f – вывести более подробную информацию UID – пользователь, PPID – ID родителя, C – уровень загрузки процессора, STIME – время запуска процесса; CMD – командная строка процесса;

ps -l – вывести еще более подробную информацию;

ps -e, ps -A или **ps aux** – вывести информацию обо всех процессах в системе. В результатах данной команды (рис. 19) присутствуют следующие важные поля:

VSZ – объем используемой процессом виртуальной памяти;

RSS – объем используемой физической памяти;

%CPU – процент использования ядра CPU;

%MEM – процент использования оперативной памяти.

```
sh-4.4$ ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
32388        1  5.3  0.0 1342600 51704 ?        Ssl  22:36   0:01 --CODINGGROUND--
32388       11  0.0  0.0  29716  2272 pts/0    Ss   22:36   0:00 sh
32388       17  0.0  0.0  29716  2044 pts/0    S+   22:36   0:00 /bin/bash
32388       18  0.0  0.0  29716  2264 pts/1    Ss   22:36   0:00 sh
32388       20  0.0  0.0  39256  1660 pts/1    R+   22:36   0:00 ps aux
```

Рис. 19. Полный список процессов в онлайн-терминале Coding Ground

top – вывести непрерывно обновляющийся список процессов. Горячие клавиши top:

P – сортировка по нагрузке на CPU (по умолчанию);

M – сортировка по объему занимаемой памяти;

W – сохранить изменения;

q – выход;

k – послать сигнал процессу;

i – отключить вывод неактивных процессов;

f – показать доступные поля;

клавиши < и > меняют последовательно поле сортировки.

ps -C bash – вывести процессы, содержащие в команде запуска заданную строку;

pgrep <имя команды> – получить PID процесса по заданной строке;

pgrep -l <имя команды> – вывести также название процесса;

pstree – просмотр списка процессов в виде дерева.

Каждый процесс может содержать несколько потоков, которые выполняются параллельно. Поток требует меньших затрат на порождение, и

кроме того, разделяет общие данные с другими потоками – имеет одно адресное пространство с ними (табл. 6). Потоки могут исполняться при этом на различных ядрах процессора, но в пределах одного устройства. В то же время, процессы могут взаимодействовать друг с другом, находясь даже под управлением различных операционных систем, через унифицированные механизмы межпроцессного обмена.

Таблица 6

Отличия потока от процесса

Процесс	**Поток**
Независим	Является частью процесса
Имеет собственное адресное пространство	Делит одно адресное пространство с другими потоками процесса
Имеет свое состояние (переменные)	В рамках процесса используют общее состояние (переменные)
Взаимодействуют через специальные механизмы связи	Быстро взаимодействует с другими потоками процесса

С точки зрения скорости параллельных вычислений использования потоков предпочтительнее использования процессов. Однако это редко возможно при решении современных задач моделирования, прогнозирования, анализа, оптимизации, обработки данных в связи с колоссальным объемом перерабатываемой информации, которая не может уместиться на одном вычислительном устройстве, или огромным объемом операций, которое одно устройство также не может выполнить.

Определение списка процессов, использующих файл

Команда **fuser** определяет PID процессов, которые используют файл. Также популярна для этих целей и другая команда, **lsof**:

lsof ~/1.txt – определить, какой процесс открыл файл 1.txt;

lsof +D ~/dir/ – вывести список всех открытых файлов в директории ~/dir (рекурсивно);

lsof +d ~/dir/ – вывести список всех открытых файлов только в самой директории ~/dir, но не глубже;

lsof -c bash – найти все файлы, открытые процессом(-ами) с заданным именем (bash);

lsof -p 10989 – найти все файлы, открытые процессом с PID=10989.

Обмен сигналами – механизм межпроцессного взаимодействия, позволяет реагировать процессам на события и оповещать друг друга об их наступлении.

В Linux каждый сигнал имеет имя, которое начинается с «SIG». Также стандартные сигналы Linux обознаются целыми числами, начиная от единицы.

SIGHUP (1) – генерируется при возникновении проблемы с управляющим терминалом или завершении управляющего процесса;

SIGINT (2) – сигнал, посылаемый при нажатии [CTRL] + [c]. Завершает работу процесса в терминале;

SIGQUIT (3) – генерируется при нажатии пользователем комбинации клавиш выхода [Ctrl] + [d];

SIGKILL (9) – немедленное прекращение работы процесса без действий по очистке памяти;

SIGTERM (15) – сигнал по умолчанию для завершения программ.

Сигнал можно 1) игнорировать (исключение: сигналы SIGKILL и SIGSTOP); 2) перехватить – при возникновении сигнала будет вызвана функция-обработчик; 3) обработать, применив действие по умолчанию.

Передача сигналов в Linux осуществляется командами **kill, killlall, pkill**.

kill -<номер сигнала> <номер процесса> – отправить сигнал заданному процессу;

kill -l – получить список доступных в ОС сигналов;

killall -<номер сигнала> <командная строка вызова процесса> – отправить сигнал заданному процессу, отыскав его по части команды вызова, начиная с первого символа (по умолчанию, регулярные выражения не используются);

pkill -<номер сигнала> <командная строка> – (заметьте «минус» перед номером сигнала) отправить сигнал заданному процессу, зная лишь произвольную часть командной строки процесса (также можно использовать регулярные выражения для задания аргумента «командная строка»). Поэтому, эта команда опаснее, чем killall (можно непреднамеренно

завершить некоторые процессы, для которых часть командной строки совпала с введенным шаблоном).

Перехват сигналов осуществляется **установкой ловушки (trap)**.

trap <command> <signal> – перехватить сигнал **signal** и отреагировать на него командой **command**;

trap -p – вывести список ловушек;

trap <signal> – удалить ловушку для сигнала **signal**.

Пример: установка ловушки для сигнала SIGINT

$ trap "echo Oops" INT

$ trap -p

trap -- 'echo Oops'

$ <Ctrl>+<C>

Oops

Дополнительная литература к лекции 8:

David A Rusling. The Linux Kernel. Chapter 4. Processes [Электронный документ] Режим доступа: https://www.tldp.org/LDP/tlk/kernel/processes.html

Береснев А.Л. Администрирование GNU/Linux с нуля. Гл. 5. Процессы /2-е изд., перераб. и доп. СПб.: БХВ-Петербург, 2010. С. 87–106.

Лекция 9. Отложенное и регулярное выполнение заданий. Текстовые файлы и потоки

Цели отложенного выполнения – осуществить через определенный период времени однократный запуск команды или задачи. Регулярное выполнение предназначено для регулярных задач: обновление системы, проверка дисков, сканирование компьютера на наличие вредоносных программ и т.д.

sleep 10m && ./script.sh – выполнить script.sh ровно через 10 минут.

Команда at ставит задания в очередь (обслуживается службой atd):

at now +5 minutes

> ls

> ps

[Ctrl-D]

Указание времени: now + <n> minutes, hours, days, weeks, month, day, HH:MM, midnight, noon, teatime (4pm), pm, am, today, tomorrow.

atq перечисляет ожидающие задания пользователя. Если пользователь является суперпользователем, выводятся задания всех пользователей. Формат выходных строк (по одному для каждого задания): номер задания, дата, час, класс задания.

atrm удаляет задания по номеру задания.

batch выполняет команды при снижении уровня средней загрузки системы. Среднее значение нагрузки должно быть ниже 0,8. Значение можно изменить, указав его команде **atrun**.

В файлах /etc/at.allow, /etc/at.deny хранятся списки пользователей, которым разрешено и запрещено пользоваться командой at.

Для регулярного выполнения задач используется служба cron. В различных версиях GNU/Linux используются разные пакеты cron.

/etc/crontab – общесистемная таблица заданий;

/var/spool/cron – каталог с пользовательскими таблицами периодических заданий.

Файл системной таблицы заданий /etc/crontab

SHELL=/bin/sh PATH=/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin # runparts

02 4 * * * root runparts /etc/cron.daily 22 4 * * 0 root runparts /etc/cron.weekly

42 4 1 * * root runparts /etc/cron.monthly

Поля 1–5 – указание периода выполнения задания. Шестое поле – пользователь, с правами которого выполняется задание. Седьмое поле – исполняемая команда.

По последней строке можно сказать, что задание /etc/cron.monthly (файл для списка ежемесячных заданий cron) будет запускаться в 4 ч 42 мин. первого числа каждого месяца.

Формат задания времени:

- минуты часа (0–59);
- час суток (0–23);
- календарное число (1–31);

- месяц (1–12);
- дни недели, начиная с воскресенья (0–6).

Внимание: если команде **crontab подать на взод файл, то исходная таблица заданий перезапишется таблицей заданий из файла!**

Cron сканирует директорию /var/spool/cron для поиска файлов crontab, названных аналогично учетным записям из файла /etc/passwd. Найденные crontab загружаются в память. Cron также просматривает /etc/crontab и файлы в каталоге /etc/cron.d. Затем Cron пробуждается каждую минуту, прочитывая все загруженные crontab, и определяет, какие команды следует запускать в данную минуту. Также cron проверяет, изменились ли таблицы, и перезагружает их в случае необходимости. Таким образом, cron не нужно перезапускать каждый раз, когда файл crontab модифицируется.

Для редактирования таблиц можно использовать команду **crontab -e**.

Текущая таблица будет открыта в редакторе по умолчанию, его можно изменить, редактируя значение переменной оболочки VISUAL или EDITOR. Если ни одна из переменных среды не определена, используется редактор по умолчанию /usr/bin/editor. Изменение editor: **sudo update-alternatives --install /usr/bin/editor editor ваш_любимый_редактор 1**

При открытии файла создается его дескриптор – неотрицательное целое число. С этим числом связывается поток ввода/вывода в файл/из файла. Для любого процесса ядро UNIX/Linux всегда создает три стандартных потока:

- поток ввода (stdin), файловый дескриптор 0;
- поток вывода (stdout), файловый дескриптор 1;
- поток вывода ошибок (stderr), файловый дескриптор 2.

Поток ввода всегда открыт только на чтение, когда как поток вывода и поток ошибок – только на запись. Поток ввода по умолчанию связан с клавиатурой, потоки вывода и ошибок – с дисплеем. Можно использовать следующие операторы перенаправления потоков:

< – оператор перенаправления стандартного потока ввода;

> или **1>** – операторы перенаправления стандартного потока вывода;

2> – оператор перенаправления стандартных потоков ошибок.

Версия операторов перенаправления без перезаписи файла: **>>, 1>>, 2>>**

Направить стандартный поток вывода и стандартный поток ошибок в один и тот же файл:

>файл 2>&1 или **&>файл** или **>&файл**

Закрыть поток перед вызовом команды:

>&- и **2>&-**, соответственно.

>| – перенаправление потока вывода с гарантированной перезаписью файла;

2>| – перенаправление потока ошибок с гарантированной перезаписью файла.

set -o noclobber – запрет перезаписи файлов при перенаправлении.

set +o noclobber – разрешение перезаписи файлов при перенаправлении (по умолчанию).

set -o – просмотр опций.

$ set -o noclobber

$ ls -l > 1.txt

bash: ls.txt cannot overwrite existing file

$ ls -l >| 1.txt

В данном примере файл 1.txt будет перезаписан, несмотря на то, что установлена опция оболочки noclobber.

program < indata.dat >> outdata.txt – на вход программе program подаются данные из файла indata.dat, а вывод программы записывается в файл outdata.txt.

Конвейер (pipe) позволяет объединить поток вывода одного процесса с потоком ввода другого процесса. Организация конвейера осуществляется с помощью символа «|».

Команда **tee** используется для отладки сложных конвейеров и позволяет как вывести информацию в файл, так и передать дальше по конвейеру.

ps | tee ps.txt – выводит информацию из потока ввода в файл(ы) и в поток вывода (tee – пример фильтра).

Итак, **фильтр** – это команда, обрабатывающая данные и пропускающая их далее по конвейеру.

more, less – постраничный просмотр текстовых файлов. Общее название – пейджеры (pager). По умолчанию для просмотра страниц man используется именно less, что можно изменить при помощи опции man -p <pager> или переменной окружения MANPAGER.

Команды less и more позволяют просматривать несколько файлов. Для просмотра файлов целиком также можно использовать команду cat, для вывода текста построчно в обратном порядке – команду tac.

По умолчанию при отображении содержимого файла с помощью cat выводимый вывод охватывает всю ширину экрана. Чтобы ограничить занятую ширину до 80 столбцов, используйте команду fold.

fold имя_файла

fold -w10 имя_файла – вывести файл в поле шириной 10 столбцов.

Другие полезные команды для вывода содержимого файлов

head -<n> <filename> – вывести n первых строк файла или потока (по умолчанию 10);

tail -<n> <filename> – вывести n последних строк файла или потока (без опции -n выведет 10 строк);

tail -f <filename> – динамически выводит последние строки файла, удобно следить за журналами.

Подсчет строк, слов, символов в файле/тексте

wc /etc/passwd

38 60 1948 /etc/passwd

wc -l <filename> – подсчет только числа строк.

Команда cut выводит только заданные символы, байты или поля из файла. Разделитель полей команды cut по умолчанию – TAB. Требуемые для вывода символы строки указывают после опции -c через запятую или тире. Другие опции:

b – выводить байты;

f – выводить поля;

d – установить символ-разделитель полей.

$ ls -ld

drwx------ 51 user1 user1 3752 Oct 22 21:04 .

$ ls -ld | cut -c1-10,35-43

drwx------ 3752

cut -f1,3 d: /etc/passwd – вывести список пользователей системы и их идентификаторы UID.

В последнем примере показано использования списка «первое и третье поле». Список может состоять в общем случае из байтов, символов или полей. Примеры списков:

2 – второй байт, символ или поле;

2-5 – все байты, символы и поля со второго по пятый;

-3 – все до четвертого;

5- – начиная с пятого;

1,3,6 – первый, третий и шестой байты, символы или поля.

1,3- – первый и все байты, символы или поля, начиная с третьего.

Сравнение файлов и каталогов

diff <опции> <объект1> <объект2> – сравнить файлы или каталоги.

diff ps1.txt ps2.txt > servicepack.patch; patch ps1.txt servicepack.patch – наложить патч («заплатку») на файл ps1.txt (применить изменения из файла ps2.txt). Файлы-«заплатки» создаются с помощью утилиты diff, а обновление файлов производится утилитой patch.

cmp <file1> <file2> – побайтное сравнение файлов;

diff3 <file1> <file2> <file3> – сравнение трех файлов.

Контекстный вывод и сравнение (опция -c):

file1.txt:

яблоки

ананасы

груши

морковь

file2.txt:

яблоки

груши

морковь

петрушка

Команда

diff -c file1.txt file2.txt

Результат

***** file1.txt 2018-02-21 17:10:29.764650235 +0300**

--- file2.txt 2018-02-21 17:10:50.768329841 +0300

****

***** 1,4 ******

яблоки

- ананасы

груши

морковь

--- 1,4 ----

яблоки

груши

морковь

+ петрушка

Разберем вывод команды. Первый файл обозначен «***», второй – как «---». Строка «**********» – это просто разделитель. Следующая строка имеет три звездочки («***»), за которыми следует диапазон строк из первого файла (в этом случае строки с 1 по 4, разделенные запятой). Затем четыре звездочки «****». После идет содержимое измененных строк в контексте (т.е., окруженных соседними строками). Если строка не изменяется, она имеет префикс двух пробелов. Однако если строка изменена, она имеет префикс из символа и пробела. Значения символов следующие:

! – указывает, что эта строка первого файла заменена на другую во втором файле;

+ – указывает строку во втором файле, которую необходимо добавить в первый файл.

- – указывает строку в первом файле, который необходимо удалить, чтобы получит второй файл.

После строк из первого файла есть три дефиса («---»), затем диапазон строк, затем четыре дефиса («----»). Это указывает диапазон строк (контекст) во втором файле, который будет выведен, и имеет отличия от первого файла.

Унифицированный режим (опция -u) аналогичен контекстному, но он не отображает лишнюю информацию.

diff -u file1.txt file2.txt

Результат:

--- file1.txt 2018-02-21 17:10:29.764650235 +0300

+++ file2.txt 2018-02-21 17:10:50.768329841 +0300

@@ -1,4 +1,4 @@

яблоки

- ананасы

киви

морковь

+ петрушка

Проверка идентичности (целостности) файлов может быть произведена командой sum или командами вычисления хэш-функций md5sum, sha256sum и другими.

md5sum fileA.txt fileB.txt fileC.txt > h.md5

Файл с хэш-суммами

cat hash.md5

fbc1e92a71d7e21f94d2454747a60506 fileA.txt

59541f64efd430fa276d5127aea4d4a3 fileB.txt

4d1c191bcff213e4a8c50ba998da58ab fileC.txt

Проверка целостности файлов по хэш-суммам

md5sum -c hash.md5

fileA.txt: OK

fileB.txt: OK

fileC.txt: OK

Команда **xxd** может создать шестнадцатеричный дамп файла, а также преобразовывать шестнадцатеричный дамп обратно в двоичную форму. Он также может выводить шестнадцатеричный дамп как массив C:

xxd -i t.bin

Результат:

unsigned char t_bin [] = {

0x6b, 0x72, 0x03, 0x63, 0x2c, 0x79, 0x2b, 0x6c, 0x6a, 0x6a, 0x6f, 0x72,

```
  0x6e, 0x0f
};
unsigned int t_len = 14;
```

Лекция 10. Текстовые файлы и потоки (часть 2)

Простое форматирование текста осуществляется командой **fmt**. Она форматирует текст так, чтобы каждая строка имела заданную равную длину (по умолчанию – 75 символов). Опция -w позволяет указать ширину текста в символах, опция -s означает только разделять строки, но не объединять их.

Текст можно **форматировать для печати**. Вывести в стандартный поток вывода подготовленный к печати текст позволяет команда pr.

pr +2 – вывести файл, начиная со второй страницы;

pr -2 – форматировать для печати в две колонки.

Иные опции:

-w – устанавливает ширину страницы (по умолчанию – 72 символа);

-o – создать отступ слева в заданное количество символов;

-d – двойной интервал.

Команда **expand** предназначена для **перевода символов табуляции в тексте в пробелы**. По умолчанию каждый символ табуляции заменяется на восемь пробелов, что можно изменить, указав опцию «-Число». Обратное преобразование пробелов в табуляцию осуществляет команда **unexpand**. Опция **unexpand** -t – установить позиции табуляции (если указано только одно значение, то позиции табуляции будут расставлены через данный интервал); опция -a – заменить все пробелы, а не только в начале строк.

Команда **uniq** позволяет удалить все (за исключением одной) строки, встречающиеся более одного раза, из файла/потока ввода. Работает с предварительно отсортированными файлами. Полезные опции команды:

-c – посчитать количество повторов каждой строки;

-d – вывести только повторяющиеся строки;

-f<n> – пропустить n полей до определения уникальности;

-i – игнорировать регистр;

-u – вывести только неповторяющиеся строки;

Вывести дублирующиеся строки, сравнивать, начиная с шестого поля:

cat file.txt

Привет Иванов И.И. MAR 2019 4.290

Кошка Петров В.В. FEB 2019 3e10

Кошка Петров В.В. FEB 2008 3e10

Алтын Friday B.W. MAY 1994 1.000

uniq -f5 -d file.txt

Кошка Петров В.В. FEB 2019 3e10

Заменой связки **sort + uniq** является команда **sort -u** (см. далее).

Команда **comm** позволяет сравнить два файла и вывести в три столбца: 1) строки, которые есть только в первом файле; 2) строки, которые есть только во втором файле; 3) строки, которые есть в обоих файлах. Численная опция команды отменяет вывод соответствующего столбца. Команда работает только для отсортированных предварительно файлов. Если на вход команды поданы неотсортированные файла, команда не отрабатывает корректно (см. пример ниже, слово «Еж», присутствующее в обоих файлах, не отражается в третьем столбце):

cat 1.txt

Кошь

Мышь

Еж

cat 2.txt

Мышь

Кошь

Еж

comm 1.txt 2.txt

Кошь

Мышь

comm: данные файла 1 не отсортированы

comm: данные файла 2 не отсортированы

Еж

Кошь

Еж

Команда **tr** чрезвычайно полезна для **удаления или замены символов** в строке. Опции команды:

-c/-C – использовать не указанное множество символов, а его дополнение до полного множества символов (т.е., инвертировать выбор символов);

-d – удалять символы из множества;

-s – заменить повторяющиеся символы из множества на один символ;

tr без опций заменяет встреченные в тексте элементы из первого множества символов на соответствующие элементы из второго множества символов.

tr [a-z] [A-Z] – заменить все строчные буквы на заглавные.

ps | tr -d '\n ' – удаление символов (перевод строки и пробел);

Вывод: **PIDTTYTIMECMD3410pts/700:00:00bash….**

echo 'boot' | tr -s [:alpha:] – исключение повторяющихся букв из слова;

Вывод: **bot**

tr -cs "[:alpha:]" "\n" <file1 >file2 – замена всех небуквенных символов на символ новой строки. Опция -s позволяет заменить идущие вплотную небуквенные символы только на один символ новой строки.

Команда **join – объединить два отсортированных файла по заданным полям**.

-t – разделитель полей;

-1 <n> -2 <m> – номера общих полей;

-a<n> – добавить в вывод непарные строки из файла n;

-v<n> – вывести только непарные строки из файла n.

cat 1.txt

drwxr-xr-x 2 root root 4096 Jan 9 21:18 mysql

cat 2.txt

2 10 20

join -1 2 -2 1 1.txt 2.txt

drwxr-xr-x 2 root root 4096 Jan 9 21:18 mysql 10 20

Команды **nl**, а также **cat -b**, позволяют осуществить нумерацию строк файла или текста в потоке вывода.

Команды **hexdump, xxd, od** позволяют получить двоичную, восьмеричную, десятичную или шестнадцатеричную репрезентацию файла.

Двоичная:

xxd -b file.txt

Восьмеричная:

hexdump -o file.txt

od -o file.txt

Шестнадцатеричная:

xxd file.txt

od -x file.txt

hexdump -C file.txt

Команда xxd имеет опцию -r, которая позволяет декодировать шестнадцатеричные данные при использовании этой же программы (в общем, стандартного формата записи с отступами) при кодировании.

Команда paste позволяет построчно соединить два файла:

paste <file1> <file2> – через [TAB];

paste -d ' ' <file1> <file2> – через пробел;

paste - - <file> – вывести файл, соединив по две строки в одну;

paste -s <file> – соединить все строки файла.

Разделение файлов на части осуществляется командой split. По умолчанию, команда записывается по 1000 строк текста в каждый создаваемый ей выходной файл xaa, xab, xac и т.д. Можно указать следующие опции команды:

-l<кол-во строк> – количество строк в частях файла;

-b<кол-во байтов> – количество байтов в частях файла;

-n – разрезать на n файлов;

-d – использовать численный суффикс x00, x01, …

csplit <file> <expression> – разделить файл на две части: до вхождения заданной строки и после нее; **{n}** в конце команды позволяет выполнить поиск шаблону expression и разделение n раз (на выходе – файлы xx00, xx01,…). Помимо {n}, можно использовать следующие шаблоны поиска:

целое_число – копировать файл до указанной строки, не включая ее;

/регулярное_выражение/ [смещение] – копировать до строки, включающей регулярное выражение, но не включать соответствующую строку.

%регулярное_выражение % [смещение] – пропустить строки до строки, включающей регулярное выражение, но не пропускать саму эту строку.

«смещение» – число строк, указанное в виде «+число» или «-число», на которое нужно сместиться после нахождения регулярного выражения.

xargs – выполнить команду, составив ее из входящих аргументов.

Часто используют связку xargs + find.

Пример:

поиск и удаление временных файлов (начинаются с «~»)

find ~ -type f -name '^~*' -exec rm -f {} \; – с опцией find -exec;

find ~ -type f -name ' ^~*' | xargs rm -f – применяя конвейер + xargs.

Сортировка строк в тексте осуществляется командой **sort.** Команда без опций осуществляет сортировку посимвольно в порядке возрастания (буквы – от «A» до «Z», цифры идут раньше букв, строчные буквы идут раньше прописных).

Опции команды:

-b, --ignore-leading-blanks – опустить ведущие пробелы;

-d, --dictionary-order – словарная сортировка (учитываются буквы, цифры и пробелы);

-f, --ignore-case – игнорировать регистр символов;

-g, --general-numeric-sort – сравнение по числовому значению в общей математической нотации (подходит как для сравнения чисел с плавающей запятой, так и целых чисел);

-h, --human-numeric-sort – сравнение размеров («5K», «2M»);

-i, --ignore-nonprinting – сортировать только по печатным символам;

-k, --key POS – сортировать по столбцу POS, а далее – по столбцам после него. Варианты команды:

-k POS1,POS2 – сортировка по полям от POS1 до POS2, далее – с начала строки. Чтобы отключить сортировку с начала строки, используйте -s.

-k POS1.SYM1,[POS2.SYM2] – сортировка по полю POS1, начиная с символа SYM1 в поле, [до символа SYM2 в поле POS2, далее – с начала строки]. Чтобы отключить сортировку с начала строки, используйте -s.

-m, --merge – объединить заранее отсортированные файлы и не сортировать.

-M, -month-sort – сравнение дат по месяцам «JAN» < ... < «DEC».

-n, --numeric-sort – численное сравнение;

-r, --reverse – сортировка в обратном порядке;

-o, --output=ФАЙЛ – записать результат в ФАЙЛ вместо стандартного вывода;

-s, --stable – устойчивая сортировка (если две строки совершенно идентичны по всем полям, используемым для сортировки, то они гарантированно появятся в результатах сортировки в исходном порядке);

-t, --field-separator=РАЗД – использовать разделитель «РАЗД» для полей вместо пробела;

--parallel=N – ограничить количество потоков для сортировки до n (по умолчанию сортировка выполняется параллельно, число потоков равно числу процессоров, но не больше восьми);

-u, --unique – удалить все идентичные строки, кроме первой (аналог uniq).

Примеры сортировки

Содержимое файла file.txt

cat file.txt

Привет Иванов И.И. March 2019 4.290

Кошка Петров В.В. February 2019 3e10

Кошка Петров В.В. February 2008 3e10

Алтын Friday B.W. May 1994 1.000

Сортировка по четвертому полю. После этого – сортировка по строке целиком (POSIX-свойство), поэтому Петров 2008 поднимается выше, чем Петров 2019:

sort -k4,4 file.txt

Кошка Петров В.В. FEB 2008 3e10

Кошка Петров В.В. FEB 2019 3e10

Привет Иванов И.И. MAR 2019 4.290

Алтын Friday B.W. MAY 1994 1.000

Устойчивая (stable) сортировка по четвертому полю. После сортировки по нему в остальном сохраняется исходный порядок строк (Петров 2019 выше Петров 2008):

sort -k4,4 -s file.txt

Кошка Петров В.В. FEB 2019 3e10

Кошка Петров В.В. FEB 2008 3e10

Привет Иванов И.И. MAR 2019 4.290

Алтын Friday B.W. MAY 1994 1.000

Сортировка со второй буквы первого слова

sort -k1.2 file.txt

Алтын Friday B.W. MAY 1994 1.000

Кошка Петров В.В. FEB 2008 3e10

Кошка Петров В.В. FEB 2019 3e10

Привет Иванов И.И. MAR 2019 4.290

Устойчивая сортировка со второй буквы первого слова до конца слова (без «,1» устойчивая сортировка не произойдет, так как сортировка выполнится до конца строки, и Петров 2008 поднимется выше Петров 2019):

sort -k1.2,1 -s file.txt

Алтын Friday B.W. MAY 1994 1.000

Кошка Петров В.В. FEB 2019 3e10

Кошка Петров В.В. FEB 2008 3e10

Привет Иванов И.И. MAR 2019 4.290

Сортировка по числу с плавающей запятой в шестом столбце:

sort -k6 -g file.txt

Алтын Friday B.W. MAY 1994 1.000

Привет Иванов И.И. MAR 2019 4.290

Кошка Петров В.В. FEB 2008 3e10

Кошка Петров В.В. FEB 2019 3e10

Простая численная сортировка сортирует по числу до точки или буквы «e», означающей экспоненту числа с плавающей точкой:

sort -k6 -n file.txt

Алтын Friday B.W. MAY 1994 1.000
Кошка Петров В.В. FEB 2008 3e10
Кошка Петров В.В. FEB 2019 3e10
Привет Иванов И.И. MAR 2019 4.290

Смена разделителя для сортировки. Сортировка по отчеству (W<В<И):

sort -t '.' -k2 file.txt
Алтын Friday B.W. MAY 1994 1.000
Кошка Петров В.В. FEB 2008 3e10
Кошка Петров В.В. FEB 2019 3e10
Привет Иванов И.И. MAR 2019 4.290

Лекция 11. Потоковые редакторы

awk и **sed** – две программы для UNIX и Linux-систем, предназначенные для обработки текстовой информации. Их называют потоковыми редакторами, Текст поступает в виде потока на обработку, файл тоже можно передать на вход потокового редактора, и он будет обработан построчно, как текстовый поток.

Программы sed и awk можно использовать в качестве интерпретаторов файлов со скриптами. Однако самое распространенное применение этих потоковых редакторов – внутри однострочных скриптов («oneliner»), которые обычно вводятся напрямую с командной строки. Умение написать однострочный скрипт для произвольной обработки текста или данных ценится в кругу Linux-программистов и может считаться важным показателем умения пользователя работать с командной оболочкой и знания основных команд Linux.

Интерпретатор/потоковый редактор sed

Для многих целей sed сейчас заменен языком perl, но для простых преобразований внутри сценариев командной оболочки sed имеет довольно большую распространенность. sed представляет собой многофункциональный редактор, служащий для обработки текстов. Этот редактор использует собственный язык программирования, который является полным по Тьюрингу. На языке sed можно написать даже игры, такие как тетрис (https://github.com/uuner/sedtris).

Существуют различные программные реализации sed, среди которых наиболее распространена GNU sed. Чтобы узнать, какая реализация установлена на компьютере, достаточно набрать команду

sed --version

Наиболее часто используемой командой является **s** (замещение), которая заменяет один шаблон другим.

sed s/dog/pet/g in> out

заменит «dog» на «pet» в файле и выведет его в файл **out**; «**g**» означает «глобальную» замену, то есть, замену всех вхождений шаблона, а не только первого.

Для экранирования сложных шаблонов следует использовать одинарные кавычки:

echo "abccbd" | sed "s/a\([bc]*\)d/\1/g"

В данном примере заменяются все вхождения буквы «a», за которой следуют b либо c, а далее произвольное количество букв, завершавшиеся буквой «d» на захваченную группу символов с номером 1 (\1). Для захвата группы символов используются круглые скобки, скобки экранированы обратным слэшем.

Теперь можно самостоятельно разобрать, почему команда

echo "abccbd" | sed "s/a*\([bc]\)\(cb\)/\2\1/g"

выведет на экран

cbcd

Заметим, что в последнем примере «**d**», часть строки вне регулярного выражения, не подвергается преобразованиям.

GNU sed дает возможность трактовать скобки как регулярные выражения всегда, включив опцию «использовать регулярные выражения» «**-r**»:

echo "abccbd" | sed -r "s/a*([bc])(cb)/\2\1/g"

Пример: вывести те группы из файла /etc/groups, членом которых является пользователь root, при этом заменить все «:» на «_»:

sed -n /root/s/:/_/gp /etc/group

В этом примере показан комбинированный эффект опций **-n** – подавить вывод строк и **-p** – печать совпадающих с шаблоном строк.

Другие опции sed:

i – вставка строки перед заданной строкой;

a – добавление строки после заданной строки;

c – изменение всей строки на заданную (удаление + вставка строки);

q – выход без дальнейшей обработки строк;

e команда – выполнить команду;

f файл – обработать текст из файла;

= – команда печати номера строки.

Вставка перед второй строкой

who | sed '2i

новая строка'

Результат:

root tty1 Mar 13 17:23

новая строка

user tty7 Mar 13 17:04

Вставка после второй строки

who | sed '2a

новая строка'

Результат:

root tty1 Mar 13 17:23

user tty7 Mar 13 17:04

новая строка

Вывод номеров строк, содержащих «Привет»

sed '/Привет/=' file.txt

Другие примеры sed

sed '2,4d' – удалить вторую и четвертую строки текста.

Модификатор «!» инвертирует выбор:

sed '2,4 !d' – удалить все строки, кроме второй и четвертой.

Для использования переменной оболочки в sed экранируйте команды двойными кавычками, а не одинарными:

pattern=TO_DELETE

sed "/^$pattern/d" file.txt

Можно использовать и многие другие разделители для команд, например, точку с запятой. Это удобно, чтобы не экранировать слэш:

sed 's;/home/user;;g' file.txt

Данная команда удалит текст «**/home/user»** из файла **file.txt.**

Для удаления повторов используйте обратную ссылку внутри замены:

echo первый первый второй | sed 's_\([а-я]*\) \1_\1_'

Для захвата всего шаблона используют «&»:

echo 'цветы ландыша' sed 's/[а-я]*/(&)/g'

Результат:

(цветы) (ландыша)

Модификатор /i команды замены позволяет игнорировать регистр букв:

echo 'Цветы ландыша' sed 's/[а-я]*/(&)/gi'

Результат:

(Цветы) (ландыша)

Опция **-e** позволит выполнить несколько операций над одним текстом:

echo 'Цветы ландыша' sed -e 's/[а-я]*/(&)/g' -e 's/ы/ок/'

Результат:

(Цветок) (ландыша)

sed '2,$ s/кот/бык/i' file.txt – редактировать от второй строки до конца файл file.txt

sed '/Привет/,/Пока/ s/кот/бык/i' – редактировать от слова «Привет» до слова «Пока» файл file.txt.

sed '5,/^$/ d' – удалить текст от пятой до первой пустой строки.

awk – это командный интерпретатор, главным образом предназначенный для обработки структурированных записей, содержащих текст.

При обработке производится разделение потока ввода на записи, каждая из которых содержит ряд полей. Каждый раз, когда интерпретатор awk встречает разделитель записей, он начинает новую запись. По умолчанию разделитель записи является символом новой строки. Это можно изменить. Внутри записи поля разделяются разделителем полей.

Работа awk заключается в следующем. Последовательно к каждой записи применяется ряд правил, записанных в основном теле скрипта awk

(ограничивается фигурными скобками {...}). Кроме того, можно задать правила, применяемые перед началом обработки всех записей (с помощью BEGIN{…}) и после окончания обработки всех записей (с помощью END{}).

Преимущества и недостатки awk

+ – простота, удобство использования в командной строке для обработки текста/данных, имеющих структуру (разделенных на записи и поля);

- – исполнение программ интерпретируемого языка (awk) медленнее, чем компилируемого (C);

- – не является языком глобального назначения, не предназначен для интенсивных вычислений, т. е. язык – специализированный;

+/- – все переменные находятся в глобальной области видимости за исключением параметров, переданных в функцию.

При вводе с консольной строки скрипт awk заключается между одинарными кавычками, а текст внутри скрипта – в двойные кавычки:

```
awk 'BEGIN {print "k"}'
# выводит k
```

Как показано в примере выше, скрипт awk может не принимать на вход текста, если используется BEGIN. Однако основное назначение awk – обработка текста, и, поэтому, обычно можно видеть следующие типы (шаблоны) вызова awk:

Команды | awk '{команды}' – в цепочке конвейера;

awk '{команды}' file.txt – для обработки file.txt;

awk '{команды}' < file.txt – аналог предыдущей команды;

awk -f file.awk file.txt – выполнить awk-скрипт из файла file.awk для обработки file.txt;

./file.awk – выполнить исполняемый файл, содержащий awk-скрипт (файл file.awk).

Нижеприведенный скрипт построчно выводит в стандартный поток вывода (при нашем обучении – будет выводиться на экран) файл file.txt:

```
awk '{print $0}' file.txt
```

Интерпретатор awk имеет ряд **встроенных команд** для обработки текста. Самой частой используемой командой awk является print. В форме

print или print $0 команда выводит в стандартный поток вывода текущую обрабатываемую запись. print $1 распечатает содержимое первого столбца записи, print $2 – второго, и так далее.

В awk также имеется ряд **встроенных переменных**. Общее число записей хранится в специальной переменной NR. Общее число столбцов в текущей записи – в переменной NF. Другие специальные переменные – разделитель полей для потока ввода FS и потока вывода OFS, разделитель записей в потоке ввода RS и в потоке вывода ORS, имя текущего обрабатываемого файла FILENAME и число записей в нем FNR (удобно использовать, если в качестве входных файлов для awk указано несколько имен файлов через пробел). Аналогично изменению переменной FS действует и параметр awk **-F.** OFMT представляет шаблон формата вывода (для форматирования чисел), его значение по умолчанию %.6g.

Выведем поля 3 и 4 с разделителем «=», использовав файл **/etc/passwd**, хранящий информацию о пользователях и аутентификации, где разделителем будем считать знак «:»:

awk -F':' 'BEGIN{OFS="=";} {print $3,$4;}' /etc/passwd

Результат:

0=0

1=1

2=2

...

Заметим, что все специальные переменные вводятся без знака $.

Самоконтроль: что выведет на экран команда **print $NF**?

Awk поддерживает **регулярные выражения**. Например, для выбора всех строк (записей), содержащих «honey», следует использовать однострочный скрипт (англ. one-liner)

awk '/honey/ {print}'

Регулярные выражение внутри шаблонов

• **/^a/** – поле начинается с **a**;

• **/a$/** – поле кончается **a**;

• **[abc]** – любой из символов **a**, **b** и **c**;

• **[a-p]** – любой символ диапазона;

• ***** – 0 или больше вхождений регулярного выражения;

• **+** – 1 или больше вхождений регулярного выражения;

• **?** – 0 или 1 вхождение регулярного выражения;

• **ab|cd** – **ab** или **cd.**

Чтобы искать текст, а не использовать регулярное выражение, применяйте экранирование:

• \+ – экранирует +

Фильтрация по выражению (выражение в отличие от шаблона включает оператор сравнения, в том числе, «~», который означает «содержит»).

$ awk '$3~0 {print}‘ file.txt

1 2 0 5

6 6 0 7

$ awk '$3!~0 {print}' file.txt

Петров Иван Иванович

1 2 1 5

Правило BEGIN выполняется только один раз, прежде чем будет прочитана первая входная запись. Аналогично, правило END выполняется только один раз, после того, как все данные прочитаны. Например,

$ awk '

> BEGIN {print "Ищем студента…"}

> /Petrov/ {++n}

> END {print "\"Petrov\" появляется в ",n, " записях." }' my_students.txt

Результат:

"Petrov" появляется в 3 записях.

Циклы и ветвление в awk

if (условие) {операторы} [else {операторы}]

while (условие) {операторы}

for (выражение; условие; выражение) {операторы}

for (индекс in имя_массива) {операторы}

Операторы:

exit – завершить исполнение скрипта;

next – перейти к обработке следующей записи;

break – завершение цикла;

continue – переход к следующей итерации цикла.

Вывод полей записи через одно:

```
awk '{ for(k=1; k<=NF; k+=2); {print $k}}'
```

Отбор записей, где первое поле больше 10:

```
awk '{ if ($1>10) {print}}'
```

Массивы в awk используют строковые индексы, а не числовые. Это позволяет обращаться к элементам не только с указанием чисел в качестве идентификатора элемента, как, например в языке C: a[5], но и с использованием, например, строки в качестве идентификатора. Поясним это на примере элемента массива a с идентификатором «k»:

```
echo "" | awk '{a["k"]=10; print a["k"]}'
# выводит 10
```

Самоконтроль: напишите one-liner для вычисления корня из числа, заданного пользователем с помощью here document и awk.

При обращении к элементам многомерных массивов в awk используют несколько индексов, разделенных запятой:

```
#объявление массива размерностью 4x4
for (i=1; i<5; i++)
 for (j=1; j<5; j++)
        vec[i,j]=i+j;
```

Следующая функция транспонирует массив:

```
{
    if (max_nf < NF)
        max_nf = NF
    max_nr = NR
    for (x = 1; x <= NF; x++)
        vector[x, NR] = $x
}
END {
    for (x = 1; x <= max_nf; x++) {
        for (y = 1; y <= max_nr; y++)
```

```
            printf("%s ", vector[x, y])
        printf("\n")
    }
}
```

Примера ввода:

1 2 6

5 6 7

7 8 44

10 29 39

Вывод:

1 5 7 10

2 6 8 29

6 7 44 39

В awk предопределены следующие **встроенные функции** (табл. 7).

Таблица 7

Встроенные функции awk

sin(expr)	синус expr
cos(expr)	косинус expr
exp(expr)	возведение в степень expr
log(expr)	натуральный логорифм expr
sqrt(expr)	извлечение корня expr
int(expr)	целая часть числа
length(s)	длина строки s
printf(fmt, ...)	форматированный вывод по спецификации fmt (аналог функции языка C)
substr(s, m, n)	подстрока в n символов строки s, начинающаяся с m
getline()	чтение следующей строки
index(s1, s2)	номер позиции, с которой s1 совпадает с s2, иначе – 0
split(s, M, c)	строка s разбивается элементы массива M по разделителю c (по умолчанию FS=" "); функция возвращает число полей

Пример: вывести только строки файла file.txt длиной более 20 символов

awk 'length($0) > 20' file.txt

Функция getline позволяет получить содержимое строки в переменную без какой-либо другой обработки, изменения указателей и переменной $0.

Например, при выполнении следующей команды

echo "2

3

4

5

6

7

8" | awk '{if ((getline t) > 0) { if ((getline t) > 0) {print(t)}; print $0}}'

на экран будет выведено:

4 2 7 5

Как видно, значение большее нуля возвращается, если команда getline выполнилась успешно, и следующая строка для считывания существовала. Для последней строки, содержащей число «8», не существует следующей строки. Первое условие – ложно, и вывода последней строки не происходит.

Функция split позволяет разделить строку на части, посчитать число элементов, и сохранить результат в массив:

count = split(string, array_name, regexp);

Пример:

count = split("my sweet home", my_arr, / /);

В переменной count сохранится число элементов (count = 3), в массив my_arr будет записано три элемента (последний – «home»).

Возможно использование системных вызовов для вызова сторонних утилит из awk:

echo "22 23 24" | awk '{system("echo -n "$1" "$3"| wc -m")}'

Результат:

5

Данный скрипт выводит количество символов в первом и третьем столбце введенного текста с учетом пробела между ними.

Дополнительная литература к лекции 11:

The GNU Awk User's Guide. 2002. Режим доступа ftp://ftp.gnu.org/old-gnu/Manuals/gawk-3.1.0/html_chapter/gawk_toc.html#SEC_Contents

Лекция 12. Работа с жесткими дисками и файловыми системами

Устройство файловой системы покажем на примере файловой системы ext4, используемой многими современными дистрибутивами Linux.

Вся область записи диска делится на **загрузочную запись и несколько разделов**. В каждом разделе содержатся **загрузочные блоки и группы блоков (цилиндров)**. Каждая группа блоков состоит из суперблока (информация о количестве блоков, о файловой системе), массива индексных дескрипторов и блоков данных (рис. 20).

Особые значения номеров inode (в файловой системе ext4):

1 – bad block (испорченный сектор);

2 – inode корневой файловой системы (/);

5 – загрузчик;

8 – журнал (скрытый файл).

stat – информация о метаданных файла;

ls -li – отобразить inode файла.

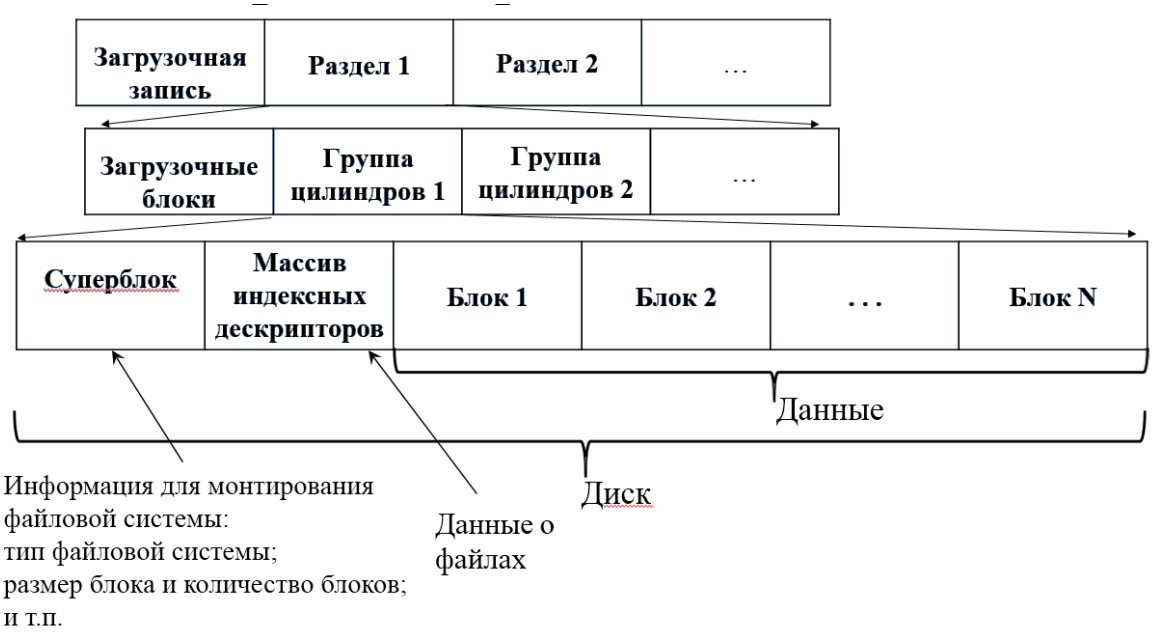


Рис. 20. Устройство файловой системы ext4

Жесткая связь – разные имена для файла, имеющего один заданный inode.

ln <filename> <newname> – создать жесткую связь;

cp -l – создавать жесткие связи вместо копирования.

Символическая ссылка (мягкая связь) – указатель на другой файл.

ln -s <filename> <linkname> – создать символическую ссылку;

cp -s – вместо копирования создавать символические ссылки.

При копировании файлов с использованием cp автоматически копируются файлы по ссылкам.

cp -d – копировать сами ссылки, а не файлы.

Устройство магнитного диска

Накопители на жестких магнитных дисках состоят из одного или нескольких магнитных дисков и магнитных головок (head) (рис. 21). Магнитная головка перемещается над поверхностью одного магнитного диска.

Магнитная поверхность диска разбита на дорожки (track). Дорожки одного диаметра на всех магнитных поверхностях образуют цилиндр (cyl), при этом справедлива формула

$$track = cyl \times head \qquad (1)$$

Каждая дорожка разбита на секторы (sect), стандартный размер которых 512 байт.

Объем диска в байтах:

$$capacity = cyl \times head \times sect \times 512 \qquad (2)$$

Стандартное количество секторов в дорожке – 63.

Основные параметры диска (геометрия) – количества цилиндров cyl и головок head.

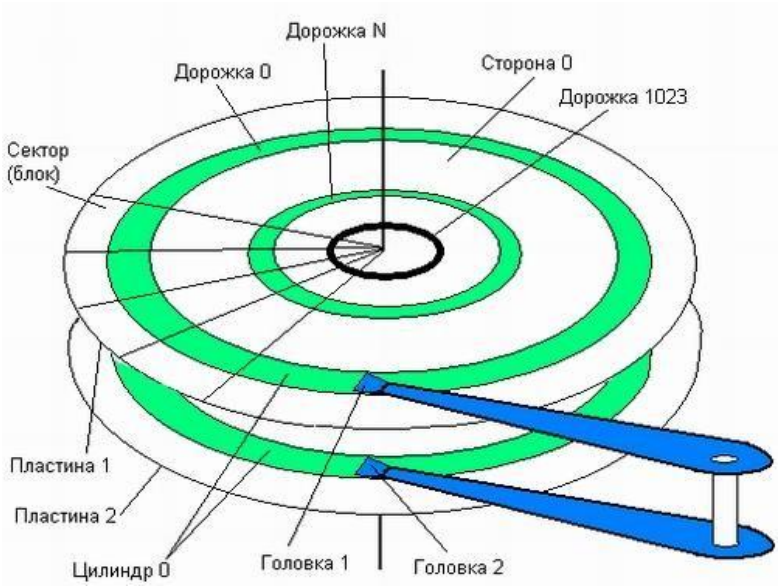


Рис. 21. Устройство жесткого диска

/sbin/fdisk -l – получить сведения о геометрии диска.

Вывод команды:

Disk /dev/sda: 160.0 GB, 160041885696 bytes

255 heads, 63 sectors/track, 19457 cylinders, total 312581808 sectors

Units = sectors of 1 * 512 = 512 bytes

Sector size (logical/physical): 512 bytes / 512 bytes

I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk identifier: ……

Твердотельные накопители (SSD) имеют более высокую стоимость за ГБ объема, меньшее число циклов перезаписи, но 1) намного более быстрые (скорость чтения), 2) бесшумные, нет механических частей, 3) легкие, 4) потребляют меньше энергии.

Диск делится на разделы (partitions) (максимум – четыре первичных), один из них может быть расширенным (extended) и содержать внутри любое количество логических разделов (logical partitions).

Имена жестких дисков SATA/SCSI

/dev/sda – Primary Master;

/dev/sdb – Primary Slave;

/dev/sdc – Secondary Master;

/dev/sdd – Secondary Slave.

Команда ls -l для файлов устройств вместо размера выводит **главный ID (major number) и вспомогательный ID (minor number)**. Главный ID – номер драйвера для данного типа устройств в Linux (8 – SATA жесткий диск, 4 – терминал, 1 – псевдоустройство – /dev/null, /dev/random, dev/zero), вспомогательный – порядковый номер устройства данного типа (0 – для всего диска, 1,2,3,4 – для первичных разделов, 5–15 – для логических разделов).

Изменение разделов при помощи fdisk производится командой

fdisk <имя устройства>

Команды интерактивной утилиты fdisk:

m – список доступных команд;

q – завершение работы без сохранения изменений;

l – список доступных типов разделов;

d – удалить раздел;

n – создать раздел;

t – установка типа созданного раздела;

w – записать таблицу разделов;

p – напечатать таблицу разделов;

x – дополнительные функции (для экспертов).

Пример:

Command (m for help): n

Partition type:

p primary (3 primary, 0 extended, 1 free)

e extended

Select (default e): e

Select partition 5

First sector (102326202-485063864): 102326202 Last sector, +sectors or +size{K,M,G} (102326202-485063864, default 485063864):

Using default value 485063864

Partition 5 of type Linux and of size 195 GiB is set

Создание файловой системы может быть выполнено командой

mkfs <раздел>

Опции команды:

• t – тип ФС;

• c – проверка на bad blocks;

• b – установить размер блока (1024/2048/4096) в байтах.

Пример (создание файловой системы ext4)

 # mkfs -t ext4 /dev/sda6

Проверка целостности файловой системы

При сбое питания или другом сбое файловая система (ФС) может потерять целостность. При этом может произойти

• частичная потеря данных, над которыми производились операции;

• появление inode, которые указывают на уже освобожденные блоки, и, наоборот, появление блоков данных, на которые не ссылается ни один inode;

• другие искажения системы метаданных.

При запуске системы Ubuntu на экране можно видеть запись /dev/sda5 – clean или ошибки, как например «orphaned block…». Первая запись свидетельствует о флаге clean файловой системы (целостное состояние), вторая запись – о нарушении целостности ФС.

fsck <опции> <раздел> – проверить целостность ФС.

Проверять целостность можно только для отмонтированной ФС! Иначе, возможна полная потеря данных.

Утилиты восстановления целостности файловой системы не восстанавливают данные, а лишь возвращают ее к неповрежденному, непротиворечивому состоянию (consistency).

 # fsck -t ext3 /dev/sdb2 – проверить целостность файловой системы ext3;

ls -l /sbin/fsck.* – список утилит для проверки ФС.

Файловая система ext4, используемая часто для Ubuntu, поддерживает журналирование. **Журналирование** – инструмент поддержания целостности ФС. Восстановление по журналу позволяет избежать в большинстве случаев запуска fdisk после сбоя работы Ubuntu.

Как работает журналирование – на диске выделяется место для хранения информации об операциях с блоками. Все операции с диском происходят как транзакции, где в журнале записывается полная информация о транзакции. При внезапном сбое по журналу можно понять, какие этапы были сделаны, а какие – нет. Например, при перенесении файла на другой раздел: 1) сначала копируется содержимое, 2) устанавливается новый inode, 3) затем удаляется старый inode. При сбое после шага 2) файл останется в двух местах, если не ведется журнал.

Ext2 – ФС без поддержки журналирования (между тем, данные нетрудно восстановить утилитами восстановления данных). Ext3, Ext4 – ФС с журналированием. **Доступные режимы журналирования (ext3):**

Journal – как данные, так и метаданные, сохраняются в журнале.

Ordered – в журнале сохраняются только метаданные. Метаданные сохраняются в журнал после данных. Это – значение по умолчанию.

Writeback – аналог ordered, за исключением того, что метаданные сохраняются в журнал до или после того, как данные будут записаны на диск.

В Ext4 журналирование можно отключить (режим Off).

Схема отключения журналирования (стоит использовать только для SSD с малым ресурсом жизни, для флэш-накопителей с установленным Linux), команды выполняются от суперпользователя (знак решетка в начале строки, выделенной жирным шрифтом):

демонтирование раздела

 # umount /dev/sdaX

отключение журнала

 # tune2fs -O ^has_journal /dev/sdaX

проверка диска

 # e2fsck -f /dev/sdaX

Ext3 поддерживает файлы до 2 TB, файловая система может быть до 32 TB. Ext4 поддерживает файлы до 16 TB, файловая система может быть до 1 EB (Exabyte = 1024 Petabyte = 1024*1024 TB).

Монтирование файловых систем

Монтирование – процесс подключения файловой системы, существующей на блочном устройстве, к корневой файловой системе.

Точка монтирования – директория, в которую монтируется устройство.

/mnt – директория для точек монтирования ФС внутренних накопителей;

/media – директория для точек монтирования ФС на съемных носителях.

Команда mount используется для монтирования ФС. Синтаксис

mount <опции> <устройство> <точка монтирования>

Команда без опций **mount** – вывести таблицу примонтированных устройств;

Отмонтировать устройства/разделы можно командами

umount <устройство>

umount <точка монтирования>

Примеры:

 # mkdir /mnt/new_disk

 # mount /dev/sda5 /mnt/new_disk

Отмонтировать корневую ФС можно

1) в однопользовательском режиме (уровень init = 1)

 # init 1

 # umount /home

 # umount /dev/sda

2) загрузившись с другой ОС, используя, например, установочный диск. При этом, для чтения ФС другая ОС должна иметь поддержку этой файловой системы. Поэтому, для чтения отмонтированной ФС ext4 удобно использовать Linux (Windows не работает с ext4).

Работа с разделом подкачки

Своппинг (подкачка) – перемещение неактивных страниц памяти из оперативной памяти на диск для экономии места в оперативной памяти (в Linux – исторически, чаще, это раздел диска, в Windows – файл pagefile.sys в корне диска). В Linux своп-раздел (или файл) также используется для гибернации (в Windows – файл hyberfil.sys в корне диска).

 # /sbin/swapon -s – вывести информацию об использовании подкачки.

Создание файла подкачки объемом 2 Гб:

dd if=/dev/zero of=swap.file bs=1k count=2097152

 # /sbin/mkswap swap.file

Создание раздела подкачки

 # /sbin/mkswap -c <раздел>

Включить подкачку

 # swapon <файл> или <раздел>

Выключить подкачку

 # swapoff <файл> или <раздел>

Подкачка может нарушить безопасность при использовании зашифрованных данных, так как на диске могут оказаться незашифрованные данные из оперативной памяти!

Файл /etc/fstab содержит список файловых систем, монтируемых автоматически при загрузке, или монтируемых по требованию пользователя. В каждой строке присутствуют поля: имя монтируемого устройства (file

system); точка монтирования (mount point); тип файловой системы (type); опции монтирования (options); опция резервного копирования (dump) с помощью утилиты dump: 1 – включено, 0 – выключено; порядок проверки раздела (pass, 0 – не проверяется, 1 – для корневого раздела, 2 – для остальных разделов).

<file system> <mount point> <type> <options> <dump> <pass>

proc /proc proc defaults 0 0

 # mount <точка монтирования> или <файл устройства> – монтирование при наличии записи в fstab.

Таблица 8

Опции монтирования файловых систем

Опция монтирования	Цель использования
defaults	опции rw, suid, dev, exec, auto, nouser, asynch
asynch	асинхронный режим ввода-вывода.
auto/noauto	включить/отключить автомонтирование во время загрузки
exec/noexec	разрешение/запрет на выполнение файлов
suid/nosuid	можно/нельзя устанавливать бит SUID
user/nouser	можно/нельзя монтировать обычному пользователю
ro	монтировать только для чтения
rw	Монтировать для чтения и записи

После изменения fstab применить изменения без перезагрузки можно данным образом:

mount -o remount раздел

Оптимизация производительности диска

hdparm <имя файла устройства> – вывести текущие параметры жесткого диска;

hdparm <опции> <имя файла устройства> – установить параметры жесткого диска.

Пример: вывод полной информации о диске

$sudo hdparm -i /dev/sda

/dev/sda:

Model=WDC WD10EZEX-08WN4A0, FwRev=01.01A01, SerialNo=.....

Другие опции команды

 # hdparm -t /dev/sda – тест скорости чтения с диска

 # hdparm -I /dev/sda – список опций для ускорения диска (использовать часть из них без опыта может быть опасно для диска!)

 # hdparm -M 128 /dev/sda – «тихий режим» (управление скоростью диска – число от 128 до 254)

Низкоуровневое поблочное копирование:

dd if=<входящий файл> of=<исходящий файл>

Опции:

count=<n> – количество блоков для копирования;

skip=<n> – число блоков, которые нужно пропустить с начала во входном файле;

seek=<n> – число блоков, пропускаемых в выходном файле до начала записи.

Резервное копирование всего диска целиком

 # dd if=/dev/sdX of=/dev/sdY bs=64K conv=noerror,sync,notrunc status=progress

bs – размер блока. По умолчанию 512 байт (осталось с 80-х годов). Лучше использовать значения около 1M;

noerror – продолжать работу, игнорируя все ошибки чтения. Поведение по умолчанию – dd должно останавливаться при любой ошибке;

sync – заполняет входные блоки с нулями, если есть ошибки чтения, поэтому данные в источнике и приемнике остаются синхронными, несмещенными;

status=progress – показывает статистику.

Чтобы сэкономить место, можно сжимать данные, созданные dd, с помощью gzip, например,

dd if=/dev/sdX | gzip -c> /mnt/removable_disk/image.img

Можно восстановить свой диск из образа с помощью:

gunzip -c /mnt/removable_disk/image.img | dd of=/dev/sdX

ddrescue – инструмент копирования информации с поврежденного диска. Копирование информации с такого диска выполняется в два прохода. Первый проход: скопировать каждый блок без ошибок чтения, а ошибки записать в 1.log.

 # ddrescue -f -n /dev/sdX /dev/sdY 1.log

Второй проход: три новых попытки чтения плохих блоков:

 # ddrescue -d -f -r3 /dev/sdX /dev/sdY 1.log

На новом диске следует исправить ошибки файловой системы

 # fsck -f /dev/sdY

или сначала попытаться **восстановить потерянные/частично поврежденные данные** (т. е., данные, которые утеряны из-за ошибок файловой системы, например, потери части inode) утилитами **testdisk, photorec, scalpel**.

Команда tar

tar <опции> <имя архива> <файлы и каталоги>

Извлечь файлы

tar -xvzf archive.tar.gz

Пояснение к команде:

x – извлечь файлы из архива;

v – печатать имена распаковываемых файлов;

z – файл является gzip-сжатым файлом;

f – имя файла с архивом;

j – bzip-сжатие.

tar -xvjf archive.tar.bz2

Извлечь в заданную директорию

tar -xvzf archive.tar.gz -C /opt/folder/

Извлечь только заданные файлы

tar -xvz -f archive.tar.gz "./1.txt" "./2/3.txt"

Для того чтобы просмотреть содержимое tar-архива и не извлекать файлы, используйте параметр «-t».

Архивировать директорию (или файл)

tar -cvf my_archive.tar dir/

tar -czvf my_archive.tar.gz dir/

tar -cjvf my_archive.tar.bz dir/

Добавить файл в существующий несжатый архив (сжатый – надо распаковать и запаковать заново)

tar -rv -f my_archive.tar 555.txt

Резервное копирование с помощью tar папки ~/ documents на примонтированный удаленный сервер

tar -cvz -f /mnt/srv/docs_$(date +%Y%m%d).tar.gz ~/ documents

Команда rsync

rsync <опции> <откуда> <куда> – копирование файлов с синхронизацией. Опции:

-v – информировать о проводимых операциях;

-r – копировать данные рекурсивно (но не сохранять метки времени и права доступа);

-a – рекурсивное копирование, сохраняющее символические ссылки, права, метки времени;

-z – сжимать данные при пересылке;

-h – удобный формат чисел в выводе размера файлов;

-e протокол – использовать указанный протокол;

--partial – продолжить прерванную передачу файлов;

--progress – показывать процент выполнения операций;

-P = --progress и --partial;

-u – пропуск файлов, имеющих в месте назначения более позднюю дату модификации.

Копирование директории на сервер и с сервера

rsync -avz dir/ user@10.0.1.214:

rsync -avz user@10.0.1.214:dir/ .

Копирование через защищенное соединение

rsync -avzhe ssh --progress user@8.8.8.8:dir/ .

Продолжить прерванную передачу файлов:

rsync -avzhe ssh -P user@8.8.8.8:dir/ .

Резервное копирование с помощью rsync (по умолчанию, старые файлы в destination получают тильду перед названием):

rsync -a -b source/ destination/

Залог сохранности данных – регулярное резервное копирование!

Места хранения резервных копий:

- Съемные носители;
- Удаленные машины / облачные сервисы.

Глава 3. АДМИНИСТРИРОВАНИЕ СЕРВЕРНЫХ СИСТЕМ

Лекция 13. Управление программным обеспечением (ПО)

Системы управления программным обеспечением служат для администрирования операционной системы Linux и других UNIX-подобных систем. Это набор программного обеспечения, позволяющего устанавливать, настраивать, обновлять и удалять программы и их компоненты.

Задачи управления программным обеспечением – реализация удобного механизма взаимодействия программиста и пользователя, управление внутренними ресурсами системы, разработка и введение стандартов, то есть, администрирование.

Новое программное обеспечение можно установить с помощью пакетов, которые хранятся в репозиториях (рис. 22). **Репозиторий** (хранилище пакетов, рис. 23) – место хранения файлов, свободно распространяющихся в сети. В репозиториях программы хранятся уже скомпилированные с метафайлами, в которых описаны основные параметры и свойства программы, а также текущая версия установочного пакета. Такой способ установки имеет ряд минусов: не все разработчики программного обеспечения могут себе позволить содержать репозиторий, для установки требуется интернет, пакеты скомпилированы под конкртеную архитектуру. Однако безусловным плюсом является удобство работы с пакетами программного обеспечения и их установкой.

Второй способ – установка программ из исходных кодов, которые свободно распространяются в интернете – способ довольно трудоемкий для начинающего пользователя. Способ будет рассмотрен в лекции позже.

Рис. 22. Схема вариантов установки программного обеспечения

Начнем знакомство с **менеджерами пакетов**, и первым будет утилита под названием **rpm (RPM Package Manager)**, которая разрабатывалась для **Red Hat Linux**, позже она была перенесена и в другие дистрибутивы Linux.

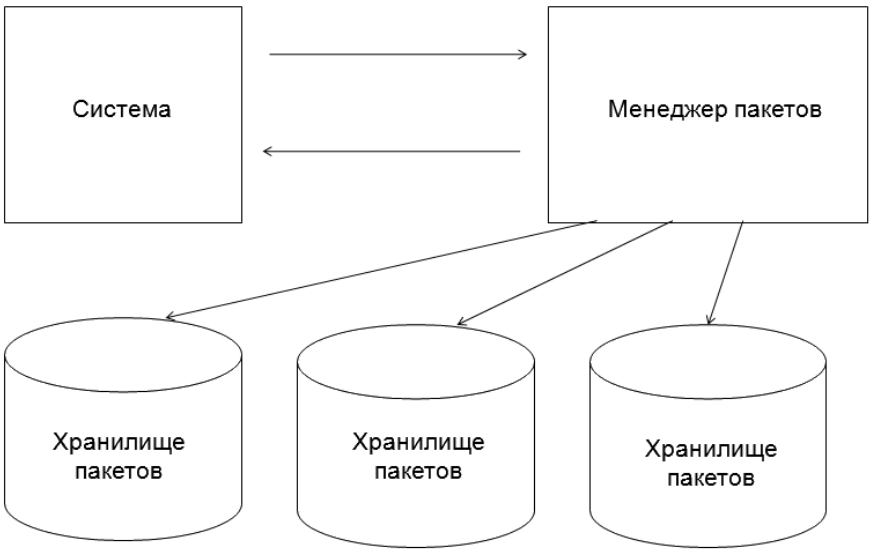


Рис. 23. Схема работы менеджера пакетов

Синтаксис утилиты: **rpm [режим опции] [пакет].rpm**

Режимы работы и основные опции:

-q – получение информации (запрос);

-i – установка;

-U –обновление;

-V – проверка пакетов;

-e – удаление;

-v – показать подробную информацию;

-i – получить информацию о пакете (**-qi**);

-I – список файлов пакета (**-qI**);

-c – список конфигурационных файлов пакета (**-qc**);

-a – список пакетов, установленных в системе;

--force – выполнять действие принудительно;

--nodeps – не проверять зависимости;

--test – проверить последствия удаления или установки пакета.

Yellowdog Ubdater, Modified (YUM) – консольный менеджер RPM-пакетов, представляющий собой оболочку для **RPM**, распространяется под лицензией **GNU**.

Основные команды **yum**:

#yum help – список команд и опций;

#yum list – список названий пакетов из репозиториев;

#yum list installed – список установленных пакетов;

#yum repolist – список подключенных репозиториев;

#yum install [пакет] – установить пакет;

#yum remove [пакет] – удалить пакет;

#yum update [пакет] – обновить пакет;

#yum info [пакет] – вывести информацию о пакете;

#yum search [строка] – найти пакет по строке в описании.

В современных дистрибутивах, основанных на RedHat, применяется обновленный, более быстрый по сравнению с yum, менеджер пакетов **dnf**. Он имеет сходный синтаксис с yum.

Advanced Packaging Tool (apt) – утилита для управления ПО в операционных системах **Debian**, а также системах, основанных на **Debian**, таких как **Ubuntu** и **Mint**. Утилита имеет широкое распространение, есть как консольные версии, так и большое число графических оболочек, таких как **Synaptic** и **Adept**.

Синтаксис утилиты: **apt [опции] [команда] [пакет]**.

Полезные опции для apt:

-t – версия релиза, для которой устанавливать пакет;

-f – выполнить операцию принудительно;

-o – строка конфигурации.

Основные команды apt:

download – скачать пакет без установки;

update – обновление информации о пакетах в репозиториях;

upgrade – обновление системы без удаления пакетов;

search – поиск пакетов в локальной базе;

show – просмотр информации о пакете;

install – установка пакета;

remove – удаление пакета;

purge – полное удаление пакета (вместе с конфигурационными файлами).

Рассмотрим и некоторые другие утилиты:

aptitude – оболочка с графическим интерфейсом для **Advanced Packaging Tool**.

dselect – программа управления с графическим интерфейсом. Является старейшим фронтендом к **dpkg**. На сегодняшний момент практически полностью вытеснена **Advanced Packaging Tool**.

dpkg – основное программное обеспечение системы управления пакетами **Debian**, а также систем со схожей архитектурой.

dpkg -i [пакет].deb – установка пакета **.deb**.

Расположение установленных программ в файловой системе

Во время установки программного обеспечения, система управления распределяет файлы программы по следующим системным папкам:

/usr/bin – исполняемые файлы программ;

/usr/sbin – исполняемые файлы, запускаемые с правами администратора;

/usr/lib – библиотека программы;

/usr/share – остальные файлы программы.

/usr/local – ПО, собранное самостоятельно из исходных кодов.

С помощью команды **“dpkg -s [название пакета]**” можно получить список файлов и их местонахождение в системных папках.

Теперь разберем, как устанавливать программы с открытым исходным кодом. Этот способ установки программ подходит для компьютеров, которые не имеют доступа в Интернет. Обычно программы с исходным кодом распространяются в виде упакованных архивов, которые перед установкой необходимо распаковать, используя, например, следующие команды:

tar -Jxvf [пакет].tar.xz

tar -zxvf [пакет].tar.gz

Дальше запускается файл конфигурации:

./configure

Скрипт командной оболочки **configure** будет пытаться определить правильные значения переменных, использующихся в процессе компиляции. Эти переменные необходимы для создания файлов **Makefile**, содержащих правила компиляции, сборки и установки, в каждом каталоге устанавливаемой программы. Также **configure** выполняет определение необходимых заголовочных файлов, содержащих системные зависимости.

Переходим к установке:

make install

Удаление:

make uninstall – удаление установленных файлов (обратно make install);

make clean – удаление скомпилированных файлов в месте сборки (обычно – очистка каталога build).

Пример Makefile с пояснениями в виде комментариев (#) приведен ниже.

```
# объявление переменной
# -c – собрать объектный файл. -Wall – включить все предупреждения
CFLAGS=-c -Wall
# цель по умолчанию (выполняется, если команда make вызвана без указания цели)
all: project
# это – цель сборки. После двоеточия указывают, от чего она зависит.
# Отсутствие зависимости позволяет не перестраивать весь проект при изменении одного файла с кодом.
project: 1.o 2.o 3.o
# в следующей строке обязателен отступ (TAB). Это команды (компиляции)
	$(CC) 1.o 2.o 3.o -o project
1.o: 1.cpp
	$(CC) $(CFLAGS) 1.cpp
2.o: 2.cpp
	$(CC) $(CFLAGS) 2.cpp
3.o: 3.cpp
	$(CC) $(CFLAGS) 3.cpp
# цель для очистки проекта (удаление скомпилированных файлов)
clean:
	rm -rf *.o project
```

Часто при компиляции из исходных кодов конфигуратор (**configure**) не может найти ту или иную библиотеку. Иногда названия библиотеки может не совпадать с названием пакета в репозитории. В таком случае можно:

1) найти недостающую библиотеку

apt-cache search ключевое_слово

2) воспользоваться системами управления с графическим интерфейсом, например, **synaptic**, который выполнит поиск нужной библиотеки внутри пакетов.

Ошибки могут возникнуть при **конфликте пакетов друг с другом**, тогда для решения проблемы придется вручную удалять наименее важные пакеты и проверить целостность оставшихся.

Исполняемые программы в Linux делятся на два типа:

1. Статически скомпонованные:

«+» – уже содержат в себе все необходимые библиотечные функции.

«+» – не требуется предварительная установка компонентов.

«–» – занимают больше места по сравнению с динамически скомпонованными.

2. Динамически скомпонованные:

«+» – занимают меньше места по сравнению со статически скомпонованными.

«–» – требуют установку дополнительных компонентов (библиотек).

Расположение библиотек в Linux

Системное программное обеспечение стандартно устанавливается в следующих расположениях:

– исполняемые файлы: **/bin, /sbin, /usr/bin, /usr/sbin**;

– библиотеки: **/lib, /usr/lib, /lib64, /usr/lib64**.

Как правило, в 64-разрядных ОС Linux поддерживаются и 32-, и 64-разрядные программы. **Путь к библиотекам для 64-разрядных библиотек: /lib64, /usr/lib64**, а к 32-разрядным /lib, /usr/lib.

Пользовательское программное обеспечение может быть установлено в системные расположения, если оно предоставляется в системе как пакет (т.е., возможна его установка через менеджера приложений) или каталог /opt

(традиционно, поскольку в прошлые десятилетия пользовательское ПО часто устанавливали с оптических дисков).

Библиотеки в свою очередь делятся на статические (static library) и динамические (shared library). Статические библиотеки используются на этапе сборки программы, а динамические во время ее выполнения.

Каждая разделяемая библиотека Linux имеет расширение so.X.Y.Z. **X – версия публичного интерфейса (API)**, считается, что каждый раз, когда он меняется, **программа теряет обратную совместимость** (нельзя работать с новой версией, если программа использует старый интерфейс). **Y – ревизия**, обычно добавляет **новые функциональные возможности к программе**. При смене ревизии публичный **интерфейс не меняется**, то есть программы, работающие со старой версией, смогут работать и с новой. **Некоторые части интерфейса могут быть признаны устаревшими (deprecated), но не могут быть удалены**, так как это нарушит API. **Z – патч**, то есть обновленная версия программы, **исправляющая ошибки, улучшающая реализацию/работоспособность**. Патч, кроме того, не меняет никоим образом публичный интерфейс, и **не добавляет новые функции**, поэтому программа с патчем и без не только обратно совместима, но и **прямо совместима** – старая версия может гарантированно быть использована вместо новой с точки зрения совместимости (как правило, старую версию использовать нежелательно с точки зрений наличия ошибок/пониженной безопасности/меньшей производительности).

Ключевые системные библиотеки:

linux-vdso.so.1 – виртуальный динамический разделяемый объект – библиотека для быстрого использования системных вызовов;

/lib/ld-linux.so.2 и /lib64/ld-linux-x86-64.so.2 – исполняемый файл, отвечающий за определение необходимых динамических библиотек для запущенного приложения, и их связывание с приложением.

Общие библиотеки:

/etc/ld.so.cache – файл, указывающий, где искать динамические библиотеки по умолчанию (общие библиотеки);

/etc/ld.so.conf – конфигурационный файл для генерации /etc/ld.so.cache;

ldconfig – утилита для генерации /etc/ld.so.cache.

LD_LIBRARY_PATH – переменная для добавления путей к библиотекам (в случае совпадения имен приоритетом обладают библиотеки, подключенные через LD_LIBRARY_PATH)

Иногда ошибки могут возникать при запуске программ (отсутствие библиотек).

Не найдена разделяемая библиотека: **Error loading shared library libopenblas.so.3: No such file or directory**

Решение:

$ ldd «исполняемый файл, использующий разделяемые библиотеки»

В списке библиотек ненайденные помечены будут как «Not Found».

$ sudo apt install apt-file

$ apt-file update

$ apt-file find [название ненайденной библиотеки]

$ sudo apt install [название пакета]

Централизованное управление программным обеспечением (ПО)

Puppet – кроссплатформенная клиент-серверная программа для централизованного управления ПО на множестве компьютеров. Используется для администрирования в крупных компаниях, позволяет легко настроить сеть и управлять ей на основе различных операционных систем.

Лекция 14. Системные журналы. Процесс загрузки и уровни выполнения

В системах Linux существует пять основных стадий загрузки ОС:

1. **BIOS** – отвечает за базовый ввод и вывод данных с устройства на устройство. Загружает главную загрузочную запись (**MBR = Master Boot Record)** – в BIOS всегда задан пользователем или по умолчанию порядок опроса дисков (Boot priority, boot sequence). Загрузка происходит с первого диска, на котором найдена действительная запись MBR.

2. **MBR** – основная загрузочная запись на диске (диск может быть **/dev/hdX** или **/dev/sdX**). Вызывает загрузчик ОС (в Ubuntu Linux – Grub). Имеет размер 512 байт.

Загрузочная запись состоит из кода загрузчика (446 байт), таблицы разделов и сигнатуры, указывающей на корректность загрузочной записи (всегда 55h AAh, «h» означает шестнадцатеричный код). Из-за ограниченного размера загрузочной записи таблица разделов может содержать не более четырех разделов по 16 байт. Каждый раздел имеет, помимо других атрибутов, свой код: 82h – Linux swap, 83h – Linux, 85h – Linux extended (расширенный). Вместо одного из разделов может присутствовать расширенный раздел, который указывает на расширенную загрузочную запись. Каждая расширенная загрузочная запись (EBR), в свою очередь, содержит указатель на следующую EBR, таким образом, число разделов не ограничивается.

Если вместо BIOS используется более новый EFI, то вместо MBR используется таблица разделов GPT. Она состоит из 34 блоков по 512 байт, записанных симметрично с двух концов диска для дублирования и облегчения восстановления повреждений. Блоки имеют номер от 0 до 33, блок 0 соответствует MBR для совместимости.

3. **GRUB (Grand Unified Bootloader)** – загрузчик операционной системы под лицензией **GNU**. Позволяет выбирать загружаемую ОС (Windows, Linux и др.) и используемое ядро Linux среди установленных.

Возможности загрузчика Grub:

- Поддерживает работу с файлами.
- Поддерживает файловые системы: **ext2, ext3, ReiserFS, XFS, JFS, FAT32, FAT16, NTFS, ISO** и многие другие
- Загружает ядра **GNU/Linux, GNU/HURD, FreeBSD, SUN Solaris.**
- Осуществляет загрузку **Windows** (передаёт управление другим загрузчикам).
- Обладает встроенной командной оболочкой.
- Поддерживает загрузчик **EFI** (с версии 1.98).
- Защита паролем пунктов меню.
- Поддерживает загрузку через сеть по протоколу **TFTP** и **BOOTP**.

Администрирование Grub:

/boot/grub/menu.lst или **/boot/grub.conf** – последовательность команд, выполняемых **grub**. Эти файлы генерируются на основе пользовательского файла **/etc/default/grub**.

sudo update-grub – обновить **/boot/grub/menu.lst** и **/boot/grub.conf** на основе **/etc/default/grub.**

<c> – перейти в оболочку **grub** (во время загрузки);

<b> – продолжение загрузки.

Команды grub:

help – помощь;

geometry (hd0) – определить структуру **HDD**;

root (hd0, 1) – раздел, с которого читается загрузочная запись (**0 диск, 1 раздел**);

setup (hd0) – установить загрузчик в главную загрузочную запись;

quit – выйти из командной оболочки **grub**;

default – образ загрузки по умолчанию;

timeout – задержка в секундах перед загрузкой образа по умолчанию;

splashimage=(hd0,4)/boot/grub/splash.xpm.gz – фоновое изображение при загрузке;

title – описание образа для загрузки;

initrd – имя образа начального электронного диска;

chainloader (file) – передать управление другому загрузчику (указать сектор).

Установка:

/sbin/grub-install – установить загрузчик grub из командной оболочки bash.

4. **Kernel** – ядро операционной системы. Запускает программу **/sbin/init**.

5. **Init** – система инициализации в **System V** (ОС Linux, выпущенная в 80-х годах компанией AT&T. Инициализация служб по образцу System V на долгие годы становилась стандартом инициализации ОС Linux), которая просматривает файл **/etc/inittab** для определения уровней выполнения и

запускает остальные процессы. Команда **init** позволяет переключить уровень выполнения **runlevel**:

init 5

Runlevel – уровень запуска и выполнения служб:

0 – выключение системы;

1 – однопользовательский режим (только root);

2 – обычно эквивалентен уровню 3. На некоторых системах (RedHat) не поддерживает NFS (см. лекцию 15);

3 – многопользовательский режим;

4 – в некоторых дистрибутивах, уровень графического сеанса;

5 – в большинстве дистрибутивов – уровень графического сеанса (в т.ч., для Ubuntu);

6 – перезагрузка системы.

/sbin/runlevel – узнать текущий уровень выполнения.

На конечном этапе загрузки происходит изменение runlevel от однопользовательской ОС (1) до многопользовательской ОС (3) с графическим интерфейсом (5). Серверные ОС остаются в режиме выполнения 3.

Уровни выполнения, журналирование с помощью syslog – это элементы концепции **System V («*System Five*»)**. Сейчас в большинстве дистрибутивов операционной системы Linux они частично или полностью заменены службой **systemd**, но обычно могут быть использованы и установлены параллельно.

На каждом уровне выполнения при переходе на него, а также при завершении работы на нем, запускаются скрипты в каталогах **/etc/rc0.d - /etc/rc6.d** (это настроено в файле **/etc/inittab**), цифра соответствует уровню выполнения.

Для добавления в ОС собственной автозапускаемой программы **command1** используйте команду

sudo update-rc.d command1 start 70 2 3 4 5 . stop 30 0 1 6 .

70 и 30 – приоритет запуска (меньшее число означает более раннюю загрузку и/или завершение).

2 3 4 5 – уровни работы программы;

0 1 6 – уровни, где программа не будет запущена.

Скрипт должен быть создан на основе файла /etc/init.d/skeleton.

Для всех уровней загрузки автозапуск можно настроить путем внесения строки в файл **/etc/rc.local**.

Служба syslog

Syslog – стандарт отправки и регистрации сообщений о происходящих в системе событиях, использующихся в компьютерных сетях, работающих по протоколу **IP**.

syslogd – демон службы **syslog**.

/etc/syslog.conf – файл конфигурации службы **syslog**.

Все службы и системные приложения Linux раньше относились к четко определенной группе. Принадлежность к группе определялась в соответствии с задачами, решаемыми приложением. Если приложение обрабатывало почту, оно относилось к группе mail, если оно отвечало за вопросы безопасности системы – к группе security и т. д. В файле конфигурации **/etc/syslog.conf** описываются действия (*action*) при получении сообщений с определенным приоритетом (*priority*), принимаемых от определенных групп (источников сообщений *facility*, рис. 24). Можно использовать шаблон глобальной подстановки «*» (*glob, wildcard*), указывая вместо конкретного названия, что подходит любой приоритет сообщения (см. ниже «Способы фильтрации сообщений»). Например, строка в файле конфигурации

mail.err /var/log/mail.err

означает прием сообщений группы «mail» с приоритетом «err» и запись их в файл /var/log/mail.err.

Отправить сообщение процесс может, например, использовав утилиту logger:

logger -p mail.err -t "MAIL DELIVERY FAILURE" 'Mail delivery failed to the following recepients: user@gmail.com'

-p – селектор (источник + приоритет);

-t – тема (заголовок) сообщения.

Последним параметром утилите logger передается сам текст сообщения.

Рис. 24. Схема конфигурации syslog

Источники сообщений:

/var/log – директория, основное место хранение лог-файлов (системных журналов).

/var/log/syslog или **/var/messages** – глобальный системный журнал, который содержит записи с момента запуска системы (запуск ядра, различных служб, обнаруженные устройства, интерфейсы и многое другое).

/var/log/auth.log или **/var/log/secure** – записи об авторизациях пользователей, включая неудачные попытки.

/var/log/dmesg – драйвера устройств.

/var/log/cron – сообщения служб **cron** и **at**.

/var/log/daemon – сообщения от демонов (служб).

/var/log/kern – сообщения ядра.

/var/log/lpr – сообщения службы печати.

/var/log/user – сообщения пользовательских программ.

/var/log/syslog – собственные сообщения службы **syslog**.

Приоритеты:

debug – отладочная информация.

info – информационное сообщение.

notice – основные события.

warning (warn) – предупреждение.

err (error) – ошибка.

crit – критическое событие.

alert – требуется немедленное вмешательство.

emerg (panic) – система не работоспособна.

Основные способы фильтрации сообщений:

[источник].* – все записи.

[источник].=[уровень] – только записи указанного уровня.

[источник].[уровень] – все записи не меньше указанного уровня.

[источник].![уровень] – все записи не меньше указанного уровня.

[источник].!=[уровень] – все записи, кроме указанного уровня.

Ротация системных журналов

В каталоге **/var/log** находятся архивы системных логов, так как эта информация может быть довольно громоздкой, имеется команда для ротации (**logrotate**) этих файлов, благодаря чему нет перемешивания новых и устаревших записей. Команда **logrotate** запускается автоматически, но есть возможность запустить ее вручную. Команда нумерует файлы, добавляя **“.[число]”** в конец файла, а также может удалять, сжимать, создавать и посылать по почте файлы журналов.

dmesg – беглый обзор информации о последней загрузки.

tail – последние записи в журнал.

more – постраничный просмотр.

less – просмотр и поиск информации.

logger – помещение в журнал своих сообщений.

Ротация журналов настраивается в общем файле /etc/logrotate и специфичных файлах для отдельных служб внутри директории /etc/logrotate.d. Доступны следующие опции:

weekly, dayly, monthly – периодичность ротации.

rotate <n> – кол-во хранимых старых журналов.

create 0666 root – с какими правами создавать новый журнал.

compress – сжимать файлы журналов.

notifempty – запрет ротации пустых журналов.

include <file> – включить файл или множество файлов.

mail – посылать журналы по e-mail.

size – максимальный размер журнала.

Пример ежедневной ротации лог-файла /var/log/filefire/main.log для некоторой псевдослужбы «filefire»:

```
/var/log/filefire/main.log {
    daily
    rotate 3
    postrotate
        filefire reload > /dev/null
    endscript
    compress
}
```

Запуск служб вручную (System V)

/sbin/service network или /etc/inid.d/network start – запуск службы network вручную.

/sbin/chkconfig -list – получить информацию о процессе начальной загрузки.

/sbin/chkconfig --level 2345 network on – включить исполнение network на уровнях 2345.

/sbin/chkconfig --add <служба> – добавить службу.

/sbin/chkconfig --del <служба> – удалить службу.

/sbin/chkconfig [level levels] on | off | reset – включить, выключить, перезапустить.

/sbin/runlevel или who -r – определить текущий уровень выполнения.

/sbin/init 1 – перейти на уровень выполнения 1.

Завершение работы и перезагрузка ОС

/sbin/init 0 или **halt** – немедленное завершение работы ОС.

/**sbin/init 6** или **reboot** – немедленная перезагрузка ОС.

halt -p или **poweroff** – завершение работы с отключением питания.

/sbin/shutdown [опции] [время] [сообщение] – команда выключения, перезагрузки, остановки, завершения сеанса локальных или удаленных компьютеров.

Опции shutdown:

-c – отменить начавшуюся остановку системы.

-a – создать файл **/fastboot** и не проверять файловую систему при загрузке.

-F – создать файл **/forcefsk** – проверить файловую систему при загрузке.

-h – остановка системы.

-r – перезагрузка системы.

-R – послать пользователям сообщение, но не перезагружать систему.

Примеры:

/sbin/shutdown -r 16:00 'Reboot at 16:00!' – перезагрузка в 16:00.

/sbin/shutdown -h now – остановка прямо сейчас.

/sbin/shutdown -h +10 – остановка через 10 минут.

Работа с systemd

systemd имеет два основных преимущества по сравнению с System-V: параллельный запуск служб при инициализации ОС, запуск служб по требованию.

В systemd единицей взаимодействия и настройки является юнит. Базовые виды юнитов:

service – управляет демонами;

socket – конфигурационный файл сокета;

device – конфигурационный файл с правилом udev для дерева устройств;

mount – отвечает за монтирование ФС;

automount – отвечает за автоматическое монтирование файловой системы.

Запуск служб вручную (systemd)

systemctl start <unit> – запустить юнит;

systemctl stop <unit> – остановить юнит;

systemctl restart <unit> – перезагрузить юнит;

systemctl reload <unit> – перезагрузить конфигурацию юнита;

systemctl status <unit> – проверка статуса юнита;

systemctl enable <unit> – сделать юнит активным (можно запускать);

systemctl disable <unit> – сделать юнит неактивным (нельзя запускать).

Для отладки работы юнитов можно просмотреть только информацию об юнитах с проблемами:

systemctl --failed

Общий журнал расположен в /var/log/journal.

journalctl -b – вывести журналы, записанные с начала работы системы;

journalctl /usr/lib/systemd/system – все сообщения утилиты systemd.

journalctl _PID=ЧИСЛО – вывести журнал по PID процесса (используется при отладке работы юнитов).

Создание юнитов

Системные юниты находятся в директории /usr/lib/systemd/system, а пользовательские юниты – в директории /etc/systemd/system.

Для юнитов создаются файлы конфигурации «.service». Юнит имеет обязательные секции «[Unit]» и «[Install]», а также «[Сервис]». При отсутствии файла юнита, но наличии файла System V с тем же именем без суффикса «.service», юнит будет создан «на лету». Также, для сервиса Test файлы с суффиксом «.conf» из каталога Test.service.d будут прочтены после разбора основного файла конфигурации юнита.

Пример файла юнита test1

[Unit]

Description=Test

[Service]

ExecStart=/usr/sbin/test1

[Install]

WantedBy=multi-user.target

Важные опции раздела [Unit]

Required= – настраивает зависимости в виде жестких требований. Если какой-либо юнит, на который опирается текущий юнит, не был запущен, то текущий юнит не будет запускаться.

Wants= – слабое требование. Юнит попробует стартовать, если какая-либо зависимость не была запущена.

Before =, After = – настройка зависимостей между юнитами.

Важные опции раздела [Install]

RequiredBy – обратная (какие юниты зависят от текущего) жесткая зависимость;

WantedBy – обратная мягкая зависимость;

sudo apt install syslog-ng – обеспечивает включение syslog в systemd наряду с journalctl.

В systemd для использования /etc/rc.local должен быть включен сервис rc-local.service:

systemctl status rc-local.service

Лекция 15. Сетевые службы Linux

Сетевая служба (*сервис, демон*) – специальный процесс (или взаимосвязанный ряд процессов) операционной системы, отвечающий за работу с локальными или удаленными соединениями заданного типа.

Сетевая служба создает **сокет** – пара ip-адрес + номер порта и привязывает сокет к данному порту в операционной системе. Говорят, что в таком случае служба выполняет **прослушивание порта**, т.е., ожидает запросов на установку соединения через этот порт. **Порт** – целое число, имеет номер от 0 до 65535. Номера портов от 1 до 1023 («общеизвестные порты») зарезервированы организацией IANA (Internet Assigned Numbers Authority) под стандартные службы. Порты с номерами от 1024 до 49151 также должны проходить регистрацию в организации. Это обеспечивает то, что никакие две службы не будут пытаться прослушать один порт. Порты с большими номерами являются динамическими.

Некоторые общеизвестные порты и протоколы, по которым осуществляется работа через эти порты.

- 20 – данные, передаваемые через протокол FTP;
- 21 – управляющая информация, передаваемая через протокол FTP;
- 22 – протокол удаленного шифрованного соединения SSH;
- 23 – удаленное соединение Telnet;
- 25 – почтовый протокол SMTP (отправка писем);
- 53 – протокол DNS;
- 80 – протокол HTTP;
- 110 – почтовый протокол POP3 (прием писем с сервера);
- 143 – почтовый протокол IMAP (прием и отправка заголовков писем);

- 443 – протокол HTTPS;
- 465 – шифрованный протокол SMTP;
- 993 – шифрованный протокол IMAP;
- 995 – шифрованный протокол POP3.

В файле /etc/services можно найти полный список сетевых служб. Здесь хранятся имя службы, номер порта, используемый протокол, псевдонимы службы.

netstat -an – получить список активных соединений.

Опции netstat:

-a – выводить спсиок портов;

-n – выводить информацию в виде чисел;

-t – только TCP-порты;

-u – только UDP-порты.

В выводе команды можно видеть следующие состояния соединения:

LISTEN (LISTENING) – идет прослушивания порта, создан сокет;

ESTABLISHED – установленные соединения.

Создание удаленного подключения telnet:

telnet имя_или_адрес_компьютера

В UNIX было популярно использовать утилиты rsh/rlogin для входа на удаленный компьютер. В файле /etc/hosts.equiv – доверенные компьютеры, вход с которых осуществляется для заданных пользователей без пароля при входе через rsh/rlogin. Но эти утилиты не используют шифрование (как и telnet) и в настоящий момент их категорически **не следует применять**.

Связь с удаленным компьютером через зашифрованное соединение

Пакет OpenSSH:

- sshd – сервер службы SSH, прослушивает порт 22 (TCP);
- ssh – клиент службы SSH, позволяет начать сеанс связи с удаленным компьютером;
- scp – программа для удаленного копирования;

/etc/ssh/sshd_config – конфигурационный файл настройки сервера;

/etc/ssh/ssh_config – настройка клиентов ssh и scp.

ssh имя_пользователя@имя_или_адрес_компьютера – начать работу с удаленным компьютером. Разрешение имен в ip-адреса производится с использованием файла **/etc/hosts**.

Копирование файлов через ssh:

scp пользователь@компьютер:<откуда-копировать> <куда-копировать>

Пример:

scp 1.txt user@10.0.0.100:/home/user/shared_docs/

Традиционно аутентифицируются либо по паролю (менее безопасно), либо через связку «публичный» + «приватный» ключ.

Для аутентификации используются протоколы шифрования RSA (в примере) или DSA:

ssh-keygen -t rsa – создание пары ключей (открытый и закрытый) асимметричного шифрования. Публичный (открытый) ключ ~/.ssh/id_rsa.pub передается на все компьютеры, с которыми должна осуществляться связь, его содержимое копируется как строка в файл авторизованных (которые разрешены для авторизации) ключей

cat id_rsa.pub >> ~/.ssh/authorized_keys;

Этот файл находится в директории .ssh в домашней папке пользователя, под которым осуществляется вход на удаленном компьютере. На файл authorized_keys, должны быть выставлены права 600.

chmod 600 ~/.ssh/authorized_keys

На саму директорию .ssh – права 700:

chmod 700 ~/.ssh

Наконец, владельцем директории .ssh и файлов в ней должен быть тот пользователь, под которым осуществляется вход.

Приватная часть ключа (закрытый ключ) ~/.ssh/id_rsa строго никому никогда не передается. Это – конфиденциальная информация, которую вместо пароля используют для входа на удаленные машины.

При подозрении на то, что приватный ключ скомпрометирован (доступ к нему получили третьи лица), **следует немедленно создать новую пару ключей и удалить все старые публичные ключи на удаленных машинах**, чтобы третьи лица, завладевшие утерянным приватным ключом,

не смогли получит к ним доступ. **Аналогично следует поступать и при компрометации пароля пользователя** (установить новый пароль).

~/.ssh/authorized_keys – публичные ключи для беспарольного доступа.

~/.ssh/known_hosts – известные хосты, с которых подключение осуществляется без дополнительного подтверждения.

Протокол передачи файлов FTP

ftp < адрес_сервера> – синтаксис команды;

sftp < адрес_сервера> – ftp через **зашифрованное** соединение (через ssh). **Всегда используйте это команду вместо незашифрованоного ftp**;

Команды интерактивного режима ftp/sftp:

ftp> help – вызов справки;

ftp>? – аналог help;

ftp> ls – вывод списка файлов и директорий; **! ls** выполняет команды для локального, а не удаленного компьютера;

ftp> cd 123 – перейти в каталог 123;

ftp> cdup – аналог cd .. в Linux;

ftp> ascii – передача файлов в текстовом режиме ASCII;

ftp> binary – передача файлов в двоичном режиме (нельзя передавать двоичные данные в текстовом режиме – приводит к потере данных!);

ftp> get 1.jpg – загрузить файл 1.jpg с удаленного компьютера (без подтверждения перезаписи!);

ftp> put 1.jpg – загрузить файл 1.jpg на удаленный компьютер (без подтверждения перезаписи!);

ftp> mget *.txt – загрузить файлы с удаленного компьютера;

ftp> mput *.txt – загрузить файлы на удаленный компьютер;

ftp> delete 1.txt – удалить файл 1.jpg с удаленного компьютера;

ftp> mdelete *.txt – удалить все текстовые файлы с удаленного компьютера.

Копирование файла в неинтерактивном режиме (аналог scp):

sftp user@10.0.0.100:docs/1.txt ~/

Утилита wget

wget <адрес_или_шаблон> – загрузка файлов с удаленного адреса;

Опции:

-c – возобновление загрузки в случае разрыва соединения;

-i <file> – взять адреса загружаемых объектов из файла;

-r – рекурсивная загрузка каталогов;

-l <num> – ограничить рекурсию до уровня num;

-x – создавать необходимую структуру каталогов;

--passiveftp – использовать пассивный режим ftp.

Пример использования:

wget -c http://releases.ubuntu.com/16.04.4/ubuntu-16.04.4-desktop-amd64.iso – загрузить образ диска Ubuntu 16.04 (система с 5-летней поддержкой до апреля 2021 года)

NFS (Network File System) – протокол сетевого доступа к файловым системам. Универсален, не зависит от ОС. Используется протокол RPC для обмена данными между клиентом и сервером. Ресурсы nfs можно монтировать при помощи команды mount.

/etc/exports – файл настройки nfs. Возможное содержимое файла:

/home/user/dir 10.1.12.0/24 (ro,async,all_squash,insecure) – предоставить каталог в режиме «только для чтения» для узлов сети 10.1.12.0/24.

sync/async – синхронный/асинхронный режим доступа – определяет, возможен ли ответ сервера до окончания записи на диск изменений, async может привести к потере данных при обрыве соединения;

all_squash – опция означает, что подключения будут выполнятся от имени анонимного пользователя.

sudo apt install nfs-kernel-server nfs-common – установка поддержки NFS в Ubuntu;

showmount -e – вывести список ресурсов, предоставленных в системе;

/etc/init.d/nfs-kernel-server restart – перезапуск службы nfs (Ubuntu);

exportfs -a – считать заново настройки /etc/exports;

mount -t nfs 10.5.122.221:/home/user /mnt/uhome – монтирование ресурса NFS;

Возможно записать в /etc/fstab информацию для удобства монтирования NFS-ресурсов через нажатие кнопки на панели дисковых устройств в Nautilus:

10.5.122.221:/home/user /mnt/uhome nfs user,rw,noauto 0 0

Служба печати CUPS

sudo apt install cups – установка CUPS через менеджер пакетов apt.

Служба обслуживания очереди заданий на печать имеет название cupsd. Она должна быть запущена для работы CUPS. Конфигурационные файлы CUPS находятся в директории /etc/cups:

• cupsd.conf – основной файл настроек сервера;

• printers.conf – описания и настройка принтеров;

• classes.conf – описания классов (групп принтеров);

• client.conf – индивидуальные настройки для клиентов (формат файлов схож с форматом сервера Apache). В данном файле должна быть строка **ServerName server.example.com:port с указанием адреса сервера печати.**

Перед изменением следует создать резервные копии конфигурационных файлов!

С помощью команды

sudo /etc/init.d/cups restart

осуществляется перезапуск службы cupsd после изменений конфигурации.

В основном файле конфигурации должны присутствовать следующие строки, которыми устанавливается прослушивание порта по умолчанию на сервере 10.0.10.129, разрешается удаленный доступ к средствам печати (Allow @LOCAL):

Listen 127.0.0.1:631

Listen 10.0.10.129:631

<Location />

Order allow,deny

Allow @LOCAL

</Location>

Веб-интерфейс для конфигурирования доступен по адресу http://localhost:631. Также для конфигурирования существует утилита командной строки lpadmin.

Примеры:

lpadmin -p laser -o job-page-limit=10 – установить ограничение в 10 страниц;

lpadmin -p laser -u allow:stud1 – разрешить печать пользователю stud1;

lpadmin -p laser -o Resolution=600dpi – изменить разрешение печати;

lpoptions -l – получить текущие настройки.

Печать с помощью CUPS документа test.txt осуществляется командой

lp test.txt

Опции lp:

-d – имя принтера;

-h – имя узла сети;

-i – идентификатор задания;

-n – количество копий;

-q – приоритет задания;

-u – имя пользователя;

-H – дополнительные опции задания;

-P – список страниц для печати.

lp -d HP1 -n 3 1.txt – печать трех копий файла.

lpstat -a – состояние очереди печати;

lp -i HP-10 -H immediate – перемещение задания в очереди на первую позицию (-H immediate);

cancel – снять задание с печати.

Пакет SAMBA позволяет Linux-компьютеру работать в сети Windows. Предназначен для общего использования ресурсов: файлов и каталогов, а также устройств.

sudo apt install samba – установка SAMBA в Ubuntu Linux.

/etc/samba/smb.conf – основной файл настройки.

Для печати важны следующие **настройки smb.conf**:

workgroup = COMP

...

security = user

…

[printers]

….

browsable = yes

guest ok = yes

Для Windows-компьютеров следует добавить опцию

use client driver = yes

Применение изменений выполняется командами

sudo restart smbd

sudo restart nmbd

Пример части основного файла конфигурации SAMBA

[global]

netbios name = IKT-201

workgroup = COMP

security = user

guest account = nobody

[homes]

comment = Home directories

read only = no

[PUB]

comment = public share

path = /home/samba/pub

read only = no

guest ok = yes

[STUD]

path = /home/samba/student

browseable = no

valid users = student

Важные настройки в файле конфигурации:

• global – раздел общих настроек;

• printers – описывает методы подключения к принтерам;

• homes – настраивает доступ к домашним каталогам пользователей;

• PUB – настройки для публичного доступа;

• STUD – настройки для пользователя student;

• netbios name – имя компьютера в сети NetBIOS;

• workgroup – название рабочей группы NetBIOS;

• security – идентификация пользователей или через использования домена;

• share – права доступа проверяются при подключении к ресурсу, но после подключения к компьютеру, предоставляющему ресурс;

• user – права доступа при подключении к компьютеру, предоставляющему ресурсы;

• domain – данный компьютер является членом домена NT, аутентификация производится на контроллере домена (PDC);

• server – задает имя иного сервера аутентификации;

• guest account – имя анонимного пользователя;

• comment – необязательный комментарий;

• path – путь к разделяемому ресурсу, или к очереди печати принтера;

• read only – доступ только для чтения;

• guest ok – разрешение гостевого доступа;

• browseable – разрешение отображать ресурс в сетевом окружении.

Настройка SAMBA может быть осуществлена как редактированием файла smb.conf, так и с помощью утилит, например, SWAT.

Подключение через SAMBA

testparm -s | less – проверка правильности конфигурации smb.conf;

nmblookup <имя компьютера> – проверить разрешение NetBIOS-имен службой nmbd;

nmblookup -M -- - – (два плюс один минус в конце команды) определить, какой компьютер сети является мастер-браузером (с именем по умолчанию __MSBROWSE__). **Мастер-браузер** – компьютер, обслуживающий базу данных NetBIOS-имен компьютеров;

smbclient -L <имя-компьютера> – получить список smb-ресурсов на сервере;

smbclient //<имя-компьютера>/<имя-ресурса> – подключение к SAMBA-ресурсу (из ОС Linux)

smbclient //IKT-201/pub – аналог предыдущей команды;

net view \\IKT-201 – подключение к SAMBA-ресурсу (из ОС Windows)

/sbin/useradd -M -d /home/samba/stud1 stud1 – регистрация SAMBA-

пользователя;

smbpasswd -a stud1 – установка пароля;

smbclient //IKT-201/STUD -U stud1 – указание пользователя при доступе;

/etc/samba/smbpasswd – база данных учетных записей SAMBA.

Монтирование файловых ресурсов SAMBA

Команда **mount.cifs (smbmount)** – монтирование файловых ресурсов протокола SMB/CIFS (реализована как демон, контролирующий события, происходящие во время монтирования и использования файловых ресурсов).

smbmount //IKT-201/pub /mnt/winshare -o username=nobody,password=' '

Разберем синтаксис команды:

//IKT-201 – имя компьютера;

pub – имя ресурса;

/mnt/winshare – точка монтирования.

Команда **smbstatus** выводит список текущих подключений SAMBA.

Лекция 16. Сетевые средства Linux

Адресация IPv4 (стандарт RFC 791, 1981 год)

IP-адрес – адрес компьютера в сети. Состоит из 32 бит (версия 4 стандарта). Из записывают в виде четырех десятичных чисел от 0 до 255 (по 8 бит), разделенных точкой, A.B.C.D, где A,B,C,D ∈ [0,255]. Пример: 192.168.10.10.

В IP-адресе можно выделить две части, разделенные точкой: **[адрес сети].[адрес узла].** Длина адреса сети разнится для сетей разных классов. Сети класса A имеют первые 8 бит, начинающиеся с 0. Это соответствует A ∈ [0,127], всего 128 сетей (рис. 25). При этом, 24 бита – произвольны, и соответствуют адресу узла (хоста, компьютера). Каждая сеть, соответственно включает 2^{24} – 2 узлов в сети (16 777 214). Число «2» отнимается, поскольку каждая сеть всегда включает широковещательный адрес (последний в сети, 127.255.255.255) и нулевой адрес, который означает саму сеть по себе и используется для маршрутизации.

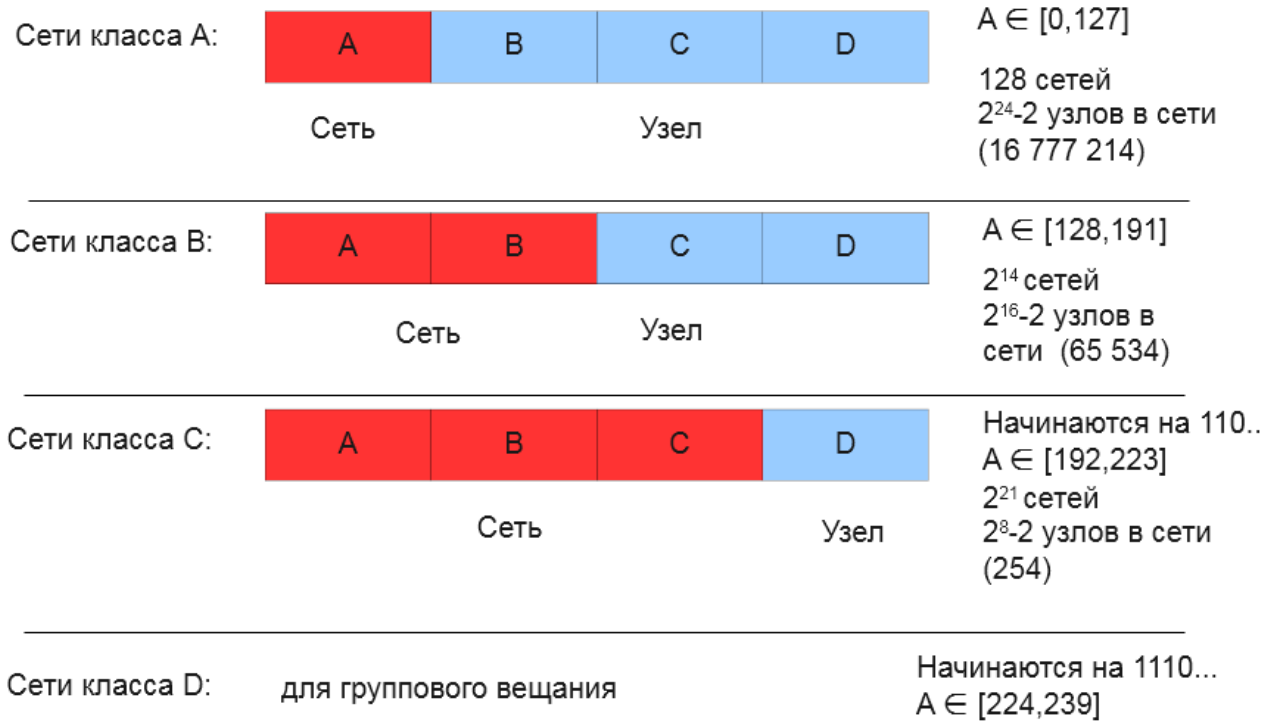


Рис. 25. Схема адресации IPv4

Маска подсети определяет то, какая часть адреса является адресом сети. Там, где биты относятся к адресу сети (а не узла), в маске содержатся единицы (табл. 9).

Таблица 9

Маска подсети

	В десятичном виде	В двоичном виде
IPадрес	192.168.111.253	1100 0000 1010 1000 0110 1111 1111 1101
Маска	255.255.255.0	1111 1111 1111 1111 1111 1111 0000 0000
Адрес сети	192.168.111.0	1100 0000 1010 1000 0110 1111 0000 0000

Маски для сетей различных классов:

Класс A: маска 255.0.0.0

Класс B: маска 255.255.0.0

Класс C: маска 255.255.255.0

Класс D: маска 255.255.255.255

Зарезервированные адреса

Адрес 127.0.0.1 сети класса A 127.0.0.0 – интерфейс «петля». Блоки адресов для локальных сетей:

Класс A: 10.0.0.0 – 10.255.255.255, одна сеть;

Класс B: 172.16.0.0 – 172.31.255.255, шестнадцать сетей;

Класс C: 192.168.0.0 –192.168.255.255, 256 сетей.

Адресация IPv6 (RFC 2373)

Возможности IPv6:

- расширенное адресное пространство (128 бит);
- нет понятия класса сети;
- упрощенный формат заголовка IP-пакета;
- встроенная поддержка IPSec – создание виртуальных частных сетей с шифрованными каналами;
- поддержка IPMobile – для мобильных пользователей.

fe80:0000:0000:0000:0250:8bff:fe5f:7ceb – IP-адрес IPv6;

fe80::0250:8bff:fe5f:7ceb – сокращение нулей;

00:50:8B:5F:7C:EB, fe80::250:8bff:fe5f:7ceb – MAC-адрес и автоматически сконфигурированный для него IPv6 адрес.

Настройка сетевого интерфейса Ethernet

/sbin/ifconfig – посмотреть текущие настройки сети;

lsmod – убедиться, что загружен модуль ядра, /etc/modprobe.conf, связанный с сетью;

eth0, eth1, eth2... – имена сетевых интерфейсов;

ifconfig eth0 [ip-адрес] – просмотр базовой конфигурации устройства (интерфейса);

ifconfig -a – информация обо всех интерфейсах;

netstat -i – информация о трафике (пакетах) через все интерфейсы;

ping [ip-адрес] – проверка работоспособности сетевого интерфейса.

Настройка сетевого интерфейса в Ubuntu осуществляется через специальные программы, вызываемые либо нажатием на значок в панели значков в верхнем правом углу экрана (рис. 26), либо в верхнем меню Параметры системы – Сеть (рис. 27).

Настройка маршрутизатора по умолчанию

Маршрутизатор (*роутер*) – устройство, передающее IP-пакеты между различными сетями. Обычно при настройке подключения указывают маршрутизатор по умолчанию (default gateway).

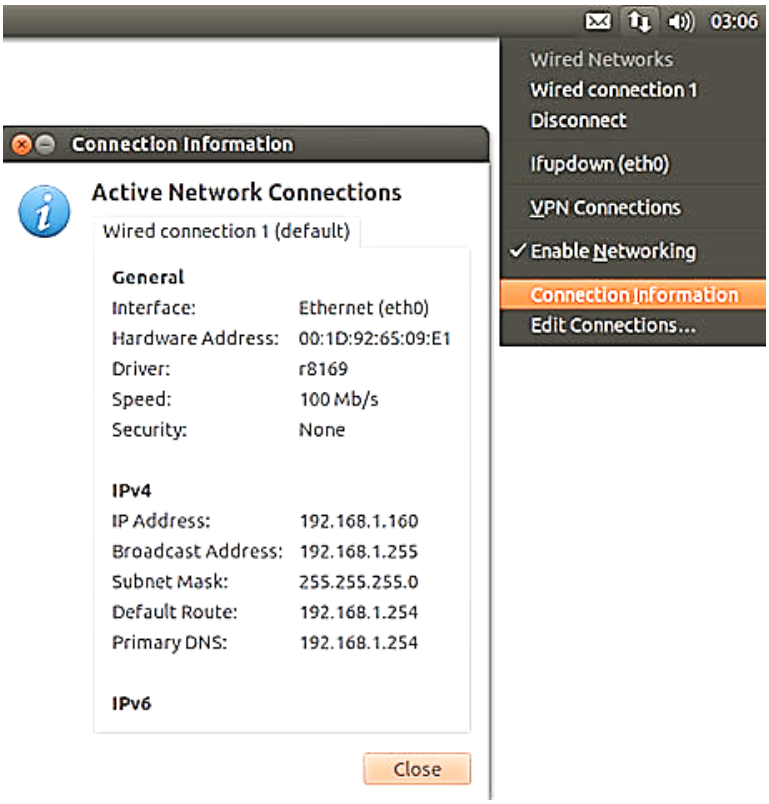


Рис. 26. Информация о сетевых соединениях в Ubuntu

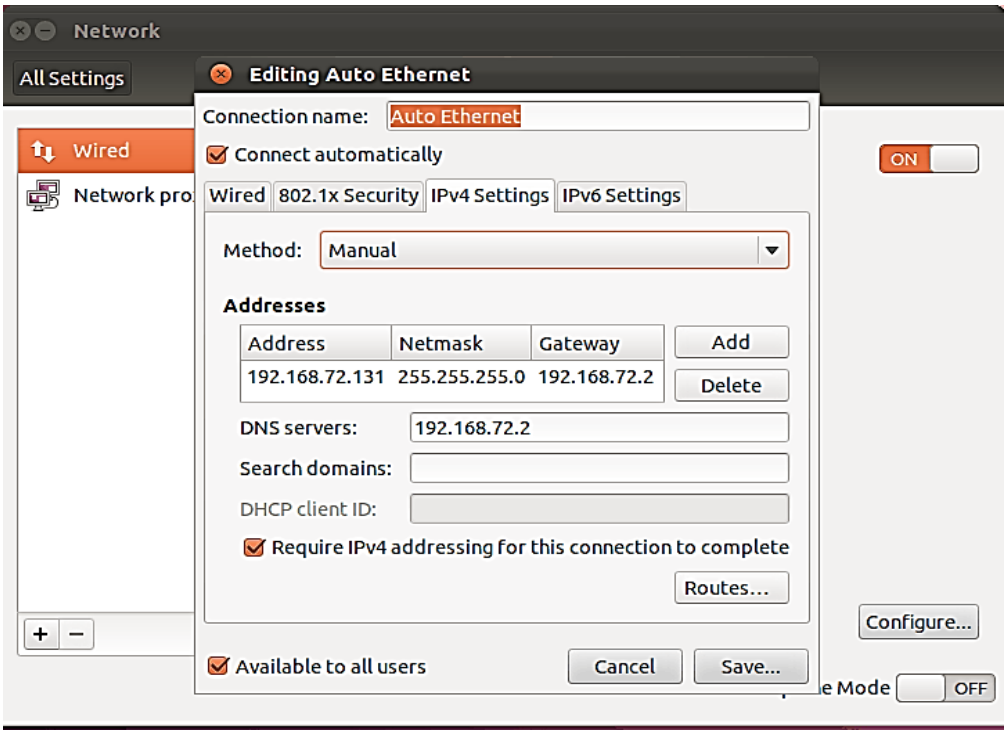


Рис. 27. Управление сетями Network в Ubuntu

В случае локальной сети или соединения точка-точка маршрутизацию настраивать не требуется. Для функционирования маршрутизации ядро должно поддерживать обмен пакетами между интерфейсами (forwarding).

netstat rn – получить текущую таблицу маршрутизации (содержит маршруты направления IP-пакетов в различные сети);

arp an – посмотреть кэш протокола ARP (преобразование IP-адресов в MAC-адреса). Для использования arp без указания пути директория /sbin должна быть в переменной $PATH;

route add default gw [ip-адрес] – назначить маршрутизатор по умолчанию;

traceroute [ip-адрес] или [имя_домена] – утилита для построения маршрута между текущим компьютером и целевым (покажет все узлы, через которые будет проходить пакет). Позволяет отследить процесс прохождения пакетов через маршрутизаторы. Существует аналог – **tracepath**.

tracert – аналог команды traceroute, но для Windows.

Просмотр и смена сетевого имени

hostname – получить сетевое имя компьютера;

/etc/sysconfig/network – настройка имени компьютера в Red-Hat-подобных системах;

Настройка имени компьютера в Ubuntu (13.04 и выше):

1) изменить имя хоста в файлах /etc/hosts и /etc/hostname

2) **systemctl restart systemd-logind.service**

Другой способ:

hostnamectl set-hostname new_host_name

Настройка разрешения имен

Разрешение имен – установление соответствия между парой «хост, домен» и IP-адресом хоста.

Способы разрешения имен:

- файл /etc/hosts – для небольших сетей;
- использование службы DNS (Domain Name System);
- использование служб NIS, NIS+, LDAP – для корпоративных сетей.

Важные файлы для конфигурации разрешения имен:

/etc/resolv.conf – настройка системы разрешения имен. В файле встречаются термины nameserver – сервер DNS, domain – имя домена, search – список доменов, которые нужно подставлять при поиске;

/etc/host.conf – тонкая настройка системы разрешения имен;

/etc/nsswitch.conf – информация о доменах и связанных с ними базах данных.

Поиск и устранение проблем с сетью

Основные причины проблем в работе сети:

- Неисправности в сетевых кабелях и сетевом оборудовании;
- Отсутствует драйвер для сетевой карты или установлен не тот драйвер;
- Неверно настроены IP-адреса узлов сети;
- Засорен кэш ARP;
- Неверно указан маршрутизатор по умолчанию;
- Неверно настроена система разрешения имен;
- Избыточная блокировка фильтром IP-пакетов (сетевым экраном).

/sbin/lspci – проверить работоспособность сетевой карты;

/sbin/modinfo – получение информации о модуле ядра;

/sbin/lsmod – список загруженных модулей;

/sbin/ifconfig, а также **netstat rn** – проверить, правильно ли настроен IP-адрес и маршрутизатор по умолчанию.

Алгоритм проверки работоспособности сети при помощи команды ping

1. Проверить работоспособность сетевой подсистемы в ядре:

ping 127.0.0.1

2. Проверить работоспособность сетевого адаптера:

ping [ip-адрес, присвоенный адаптеру]

3. Проверить работоспособность локальной сети:

ping [ip-адрес локального компьютера]

4. Проверить прохождение пакетов к внешнему сетевому адаптеру маршрутизатора по умолчанию;

ping [ip-адрес маршрутизатора]

5. Проверить прохождение пакетов по любому внешнему IP-адресу

ping [ip-адрес удаленного компьютера]

6. Проверить работу подсистемы разрешения имен

ping [имя домена]

Утилита netstat – информация о соединениях, и процессах, их открывших

netstat -tulpn | grep :80 – вывести информацию об открытых портах и отфильтровать только информацию о порте 80;

Nmap – утилита для исследования состояния компьютеров в сети (проверка открытых портов и т.п.). Применяется для проверки безопасности и корректности настройки сети.

nmap -sV -p 22,80,110,143,8080 192.168.0-10.1-255 – исследование состояния портов 22,80,110,143,8080 с попыткой определения открывшего порт приложения для ряда хостов от 192.168.0.1 до 192.168.10.255.

nmap -sP 192.168.0.0/16 -oG file.txt – пинг-сканирование всей локальной сети класса B для ряда хостов от 192.168.0.1 до 192.168.255.255 с выводом результатов в файл file.txt в удобном для программы grep формате.

Проверка кэша ARP

Ethernet работает с MAC-адресами, а не IP-адресами. Поэтому, для установки соединения компьютер выполняет широковещательный запрос (broadcast) по протоколу ARP. Целевой компьютер получает запрос и посылает ответный запрос, содержащий MAC-адрес. Кэш arp позволяет не перезапрашивать IP-адрес заново в течение некоторого времени (не превышающего, обычно, несколько часов)

/sbin/arp a – вывести содержимое кэша ARP (должно быть правильное сопоставление MAC-адресов и IP-адресов);

/sbin/arp d – удаление неправильных записей в кэше;

/sbin/arp s – установить запись вручную.

DNS (Domain Name System) – компьютерная система для получения информации о доменах. Основное употребление – перевод доменного имени в IP-адрес.

Для сайта disk.yandex.ru уровни доменов: III – disk, II – yandex, I – ru (доменная зона).

host – утилита для тестирования клиента;

dig – утилита для тестирования DNS-сервера;

nslookup – позволяет тестировать и сервер DNS, и клиента – подсистему разрешения имен.

Утилита IPTraf – мониторинг сетевого трафика.

/var/log/iptraf – директория для хранения лог-файлов по умолчанию.

Сетевой экран (*брандмауэр*, *firewall*) – программа, осуществляющая фильтрацию сетевого трафика на основе заданного набора правил. Популярный межсетевой экран, встроенный в ядро Linux с версии 2.4, – **netfilter**. Утилита конфигурирования правил в netfilter называется **iptables**.

Основы конфигурирования с помощью iptables

Обмен трафиком между компьютерами происходит пакетами (информация разбивается на части опредленного размера, упаковывается и пересылается). Пакеты пропускаются через **цепочки, то есть, списки правил.**

Правила сопоставления пакетов

Каждый пакет проходит цепочку, начиная от первого правила. Как только найдено соответствие правилу, осуществляется переход (-j = --jump) к указанному действию (обычно – REJECT, ACCEPT, DROP).

Цепочки

Пять основных типов цепочек:

PREROUTING – предварительная обработка входящих пакетов.

INPUT – цепочка входящих пакетов с конкретным процессом-адресатом.

FORWARD – для перенаправляемых пакетов ().

OUTPUT – цепочка исходящих пакетов от локальных процессов.

POSTROUTING – пост-обработка исходящих пакетов.

Переадресуемые пакеты проходят три цепочки: PREROUTING – FORWARD – POSTROUTING, а не одну.

Стандартные действия, доступные во всех цепочках – ACCEPT (разрешить прохождение пакета), DROP (отклонить), QUEUE (передать на анализ внешней программе) и RETURN (вернуть в предыдущую цепочку).

Цепочки организованы в таблицы.

filter – основная таблица, используется по умолчанию, если название таблицы не указано. Содержит цепочки INPUT, FORWARD и OUTPUT.

iptables-save > /etc/network/iptables.rules – сохранить изменения iptables в файл.

В файл /etc/network/interfaces следует для интерфейса (устройства), к которому применяется данная политика фильтрации пакетов, указать строку

pre-up iptables-restore < /etc/network/iptables.rules

В версии Ubuntu 16.04 появился специальный пакет, упрощающий сохранение конфигурации iptables:

sudo apt install iptables-persistent

Сохранение и перезагрузка (применение) конфигурации iptables может быть прозведена командами

sudo /etc/init.d/iptables-persistent save

sudo /etc/init.d/iptables-persistent reload

Примеры использования iptables

iptables -L -n -v --line-numbers – просмотреть таблицы;

iptables -I INPUT 3 -s 205.205.205.205 -j DROP – вставить в цепочку INPUT правило под номером 3, блокирующее (-j DROP) прием пакетов от источника (-s = --source) 205.205.205.205.

iptables -A INPUT --source 127.0.0.1 --jump ACCEPT

iptables -A INPUT --jump FURTHER_ANALYSIS

Эти строки добавляют в конец цепочки INPUT два правила: первое правило – «пропустить все пакеты от 127.0.0.1», а второе правило – отправить (оставшиеся пакеты) на анализ в цепочку FURTHER_ANALYSIS.

iptables -A INPUT -i lo -j ACCEPT

iptables -A OUTPUT -o lo -j ACCEPT – разрешить трафик для интерфейса lo (внутренний трафик интерфейса локальная петля, lo – стандартное название интерфейса).

iptables -A INPUT --i eth0 -p tcp --dport 9000 -j LOG --log-prefix "Port 9000 incoming blocked! "

iptables -A INPUT -i eth0 -p tcp --dport 9000 -j DROP – правило в предыдущей строке позволяет отправить информацию о пакетах, пришедших на порт 9000 через устройство eth0 в /var/log/messages, а последнее правило блокирует прием этих пакетов.

host -t a имя_домена – найти ip-адрес для домена.

whois ip-адрес | grep CIDR – найти диапазон ip-адресов, используемых доменом и поддоменами.

На основе информации о диапазоне адресов, можно заблокировать выход на определенный домен и его поддомены

iptables -A OUTPUT -o eth0 -d 205.205.205.0/24 -j DROP – запретить исходящие соединения в заданную подсеть.

iptables -A INPUT -p icmp --icmp-type echo-request -j DROP – замаскировать компьютер, скрыв его от ping-запросов.

iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT – разрешить входящих трафик, если соединение с удаленным копьютером было инициализировано первично локальным компьютером.

iptables -A INPUT -p tcp -s 205.205.205.0/24 --dport 22 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT – разрешить соединения по порту 22 (стандартно, используется службой ssh) только с компьютеров подсети 205.205.205.0/24 (если политика по умолчанию для входящих пакетов – DROP).

Механизм NAT (Network Address Translation) – IP-адрес локального компьютера подменяется сетевым экраном или маршрутизатором подменяется на имеющийся внешний IP-адрес (например, маршрутизатора) для обмена пакетами с удаленным компьютером, для каждого соединения используется отдельный порт. Пакеты, поступившие от удаленного пакета по этому порту, переадресуются обратно инициировавшему сеанс связи локальному компьютеру.

Существуют **антивирусные программы для Linux**. Популярный антивирус – **ClamAV**. Управляется из командной строки.

Установка:

sudo apt install clamav clamav-daemon или **sudo yum install clamav**

Обновление базы данных вирусов

sudo freshclam

Сканирование без очистки с оповещением при нахождении вирусов

sudo clamscan -r /home/user --bell

Сканирование с автоматическим удалением инфицированных файлов

sudo clamscan --infected --remove -r /home/user

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Таненбаум Э., Бос Х. Современные операционные системы /4-е изд. СПб.: Питер, 2015. 1120 с.

2. Результаты опроса участников сайта LinuxQuestions.org, 2017 [Электронный ресурс]. Режим доступа: https://www.linuxquestions.org/questions/2017mca.php. Дата обращения: 07.03.2018.

3. Береснев А.Л. Администрирование GNU/Linux с нуля /2-е изд., перераб. и доп. СПб.: БХВ-Петербург, 2010. 576 с.

4. Список сигнатур файлов (англ.) Википедия – свободная энциклопедия [Электронный ресурс] Режим доступа: https://en.wikipedia.org/wiki/List_of_file_signatures. Дата обращения 05.03.2018.

5. BASH Programming - Introduction HOW-TO [Электронный ресурс]. Режим доступа http://tldp.org/HOWTO/Bash-Prog-Intro-HOWTO-11.html. Дата обращения 16.03.2018.

6. Cooper M. Advanced Bash-Scripting Guide [Электронный ресурс]. 2014. Режим доступа http://tldp.org/LDP/abs/html/index.html Дата обращения 08.02.2018.

7. Bash Guide for Beginners. Chapter 4. Regular expressions. 4.3. Pattern matching using Bash features [Электронный ресурс]. Режим доступа http://tldp.org/LDP/Bash-Beginners-Guide/html/sect_04_03.html. Дата обращения 08.02.2018.

8. Lee X. PowerShell as cmd.exe or Bash [Электронный ресурс]. Режим доступа: http://xahlee.info/powershell/commands.html. Дата обращения 01.02.2018.

9. Rusling D. A. The Linux Kernel. Chapter 4. Processes [Электронный ресурс]. Режим доступа: https://www.tldp.org/LDP/tlk/kernel/processes.html. Дата обращения 20.03.2018.

Учебное издание

МИТРИЧЕВ Иван Игоревич

АДМИНИСТРИРОВАНИЕ ОПЕРАЦИОННЫХ СИСТЕМ

Конспект лекций

Редактор: Е. В. Копасова

Подписано в печать 22.10.2018 г. Формат 60x84 1/16.
Усл. печ. л. 9,1. Уч.-изд. л. 9,6. Тираж 100 экз. Заказ

Российский химико-технологический университет имени Д. И. Менделеева
Издательский центр

Адрес университета и издательского центра:
125047 Москва, Миусская пл., 9