Вопросы для допуска к экзамену

Ответы отмечайте синим

Тема «История ОС GNU/Linux и UNIX-подобных ОС»

- В какое десятилетие была разработана первая версия ОС UNIX: 1960-1970, 1970-1980, 1980-1990, 1990-2000?
  - 1960-1970
  - 1969 год MINIX
- В какое десятилетие было разработано ядро ОС Linux: 1960-1970, 1970-1980, 1980-1990, 1990-2000?
  - 1990-2000
  - 1991 год Линусом Торвальдсом
- В какое десятилетие был основан проект GNU: 1960-1970, 1970-1980, 1980-1990, 1990-2000?
  - 1980-1990
  - 1985 год
- Укажите ключевые отличия между ОС UNIX и ОС семейства GNU/Linux.
  - UNIX – проприетарная, с закрытым кодом, GNU/Linux – свободные ОС
- На каких условиях можно применять ОС семейства GNU/Linux: бесплатно для ознакомления, бесплатно для некоммерческого условия, бесплатно для коммерческого использования.
  - Бесплатно для коммерческого использования.
  -
- **Какие ограничения существуют в применении ОС данного семейства?**
  - ????????

    По идее нет ограничений

- Какие ограничения накладывают лицензии BSD и GPL? Почему компаниям выгодно использовать данные лицензии при разработке ПО? Почему это может быть проблемой?
  - BSD разрешает закрывать код проектов под их лицензией, а GPL выступает за полную кодовую открытость.
  - BSD позволяет взять код под ее лицензией, изменить ее под свои нужды и закрыть, продавая ее. Так сделали Sony: взяли ядро FreeBSD и изменили ее под свои нужды, создав операционную систему для PlayStation, таким же образом поступили и Apple, чья система частично основана на FreeBSD или Nintendo. При этом, им вменяется и самими разработчиками, что они "берут, но не отдают обратно", тем самым паразитируя на продуктах лицензии BSD. GPL же относится порицательно к подобным действиям и будто бы "приобщает к общественности" все программы под своей лицензией, фактически лишая права собственности на свой код, но это на первый взгляд. На деле же она запрещает закрывать код, а те, кто берут и изменяют части куска кода под их лицензией, обязательно должны выложить модифицированную версию под той же лицензией.
- Являются ли ОС семейства GNU/Linux исключительно решением для использования на серверных устройствах? Можно ли использовать их на других устройствах?
  - Нет, их можно использовать на телефонах, ПК, игровых приставках.
- Каким образом можно получить доступ к ОС GNU/Linux в рамках других распространённых настольных ОС: Windows, macOS?
  - Виртуальная машина, WSL - windows subsystem for linux
- Каким образом происходит установка дистрибутива GNU/Linux Debian в рамках виртуальной машины VirtualBox?
  - Скачать ISO с официального сайта
  - Создать и настроить виртуальную машину: виртуальный жесткий диск, выделенное количество системных ресурсов (ЦП, RAM)
  - Установить Debian, следуя подсказкам установщика
- Как можно получить доступ к командному интерфейсу GNU/Linux?
  - Если установлен графический интерфейс, то получить доступ к командному интерфейсу Линукс можно через любой эмулятор терминала
  - Если графического интерфейса нет, то после входа в систему можно воспользоваться виртуальным терминалом
  - Можно подключиться через SSH

Тема «Базовые принципы использования командного интерфейса»

- В каком формате происходит взаимодействие пользователя и вычислительной системы при использовании командного интерфейса?
  - Пользователь вводит текстовые команды с клавиатуры, а приложение-интерпретатор их выполняет
- Что такое командный интерпретатор? Перечислите известные вам командные интерпретаторы.
  - Командный интерпретатор - это обычное приложение, которое запускается средствами ОС, следовательно под каждой ОС может быть установлено множество таких интерпретаторов (Windows - cmd, PowerShell, MacOS - bash, zsh, Linux - bash, zsh, fish, dash)
- Что такое эмулятор терминала? Перечислите известные вам эмуляторы терминала.
  - Для работы с командным интерпретатором необходимы другие приложения - эмуляторы терминала - графический интерфейс, в котором можно взаимодействовать с нужными командными интерпретаторами (Windows-приложения - ConEmu, Mintty, Windows Terminal; Linux-приложения - gnome-terminal, console, terminator, xfce4-terminal, xterm; macOS-приложения - Terminal, iTerm2)
- Что такое приглашение к вводу? Зачем оно применяется в командном интерфейсе? В каких приложениях данный подход рекомендуется использовать?
  - В качестве приглашения, как правило, указывается текущий каталог, с которым работает командная оболочка, и символ **$** (bash), означающий ожидание новой команды; основные сферы применения - операционные системы, чаты, компьютерные игры (Наиболее ярким примером игр, использующих интерфейс командной строки, могут быть названы текстовые квесты, а также сетевые многопользовательские ролевые игры. Команды в таких играх вводятся на так называемом псевдоестественном языке)
- Какую информацию можно извлечь из базового приглашения интерпретатора Bash?
  - имя пользователя, с правами которого будут выполняться команды
  - название компьютера
  - текущий рабочий каталог
  - уровень прав текущего пользователя ($ — обычный пользователь, # — суперпользователь)
- Опишите основной цикл взаимодействия с командным интерпретатором.
  - Командный интерпретатор показывает приглашение для ввода
  - Пользователь вводит текстовую команду и нажимает «Ввод»
  - Командный интерпретатор обрабатывает строку
    - Если команда была введена неверно, то интерпретатор сообщит об ошибке
    - Если команда была введена корректно, то задача будет выполнена
  - Пока команда выполняется, то ввод и вывод передаются команде
  - После завершения команды интерпретатор опять показывает приглашение

- Какой вывод следует ожидать от команды, которая успешно выполнили своё действие? Какой вывод следует ожидать от команды, которая завершила свою работу ошибочно? Как можно проверить результат работы последней выполненной команды?
  - При успешном выполнении команды вывод зависит от самой команды (date выведет информацию о текущей дате и времени, ls выдаст содержимое текущего каталога)
  - Если выполнение завершилось ошибочно, то командный интерпретатор сообщит об ошибке
  - Проверить результат работы последней команды можно введя $? данная команда выведет код, если все выполнено успешно то код будет равен 0
- Какие задачи можно решать с помощью командного интерфейса? Какие задачи невозможно решить с помощью данного интерфейса, почему?
  - Можно решать:
    - копировать и перемещать файлы, переименовывать их и создавать новые папки;
    - форматировать диски, подключать их и отключать;
    - запускать приложения и программы без интерфейса, давать им задания, получать результат;

Тема «Устройство файловой системы и основные команды для работы с файлами»

- Каким образом устроено дерево каталогов ОС GNU/Linux и ОС семейства Windows? Чем они схожи, в чём принципиальное отличие между ними?

В современных полноценных ОС (не мобильной iOS) для организации дискового пространства используется используется иерархическая структура каталогов:

- **Единицей организации данных является каталог, который может содержать другие файлы и каталоги**
- **Первый каталог файловой системы называется корневой директорией**
- **В результате формируется древовидная структура файловой системы**

В Windows существует множество файловых деревьев, для идентификации которых используются названия C, D и так далее.

В UNIX-подобных ОС существует только одно дерево каталогов, различные дисковые разделы подключаются к единому дереву каталогов.

- Что обозначает термин текущий рабочий каталог? К какой части вычислительной системы данный термин применим?

Для удобства построения путей к нужным файлам была введена концепция текущего рабочего каталога. С её помощью можно строить пути либо абсолютно, либо относительно текущего каталога. Текущий рабочий каталог — это каталог, в котором в настоящее время работает пользователь.

Текущий каталог есть у любой программы, в том числе и у командной оболочки (shell) пользователя. Поскольку взаимодействие пользователя с системой обязательно опосредовано командной оболочкой, можно говорить о том, что пользователь «находится» в том каталоге, который в данный момент является текущим каталогом его командной оболочки.

- Для чего используются специальные каталоги «..» и «.»? В какой момент времени они создаются?

Согласно древовидному представлению мы можем двигаться в трёх направлениях:

- **Вверх, к родительскому каталогу**
- **Остаться на месте в текущем каталоге**
- **Вниз, в один из подкаталогов**

Для движения «вверх» необходимо использовать специальное название: ..

Для указания текущего каталога используется специальное название: .

Создаются при создании каталога, .. – обращение к родительскому каталогу, . – ссылка на тот же самый каталог.

- Что такое относительный путь? Что такое абсолютный путь? Как эти понятия связаны с понятием текущего рабочего каталога?

Относительный путь к файлу от документа — это путь к файлу относительно текущего документа. Такой адрес зависит от расположения файла, в котором он записан.

Путь к файлу: текущий рабочий каталог + относительный путь.

Абсолютный путь — это путь, который указывает на одно и то же место в файловой системе, вне зависимости от текущего рабочего каталога или

других обстоятельств. Полный путь всегда начинается с корневого каталога.

- Какая команда может быть использована для получения текущего рабочего каталога?

pwd позволяет узнать текущий рабочий каталог.

Если текущий рабочий каталог содержит символические ссылки: можно использовать опцию –P, ссылка будет преобразована в исходный путь.

- Какие файлы считаются «скрытыми» в ОС GNU/Linux? Где такие файлы обычно находятся? Какое назначение у данных файлов?

Согласно соглашению об именах, файлы, которые начинаются с символа точки, считаются скрытыми, и их приложение ls в своём выводе по умолчанию не показывает.

Скрытые файлы в операционных системах Unix, - это файлы, используемые для выполнения некоторых сценариев или для хранения конфигурации некоторых служб на вашем хосте. Некоторым популярным примером скрытых файлов являются файлы, содержащиеся в домашнем каталоге пользователя: .bashrc, в котором хранятся сценарии инициализации пользователя, или .bash_logout, который выполняется всякий раз, когда вы выходите из сеанса bash.

Чтобы увидеть все файлы, включая скрытые необходимо приложению ls передать аргумент -a: ls –a

Многие находятся в домашнем каталоге.

- Какие команды можно использовать для изменения текущего рабочего каталога?

Для изменения текущего рабочего каталога используется команда cd

Для перехода в другой каталог необходимо в качестве аргументу команды cd передать либо абсолютный, либо относительный путь к целевому каталогу.

Команда cd может также принимать следующие специальные аргументы:

· **- — переход в предыдущий каталог**

- **без аргументов — переход в домашний каталог**
- **~bob — переход в домашний каталог пользователя bob**

Также можно воспользоваться ~ для указания пути к домашнему каталогу текущего пользователя. Данный путь будет работать везде, не только с командой cd.

- Как следует работать с файлами, содержащими пробелы в своих названиях?

В Linux для текстового интерфейса пробелы в названии файлов представляют определенную сложность.

Предположим, что мы хотим перейти в каталог с названием cool things и напишем команду cd cool things. Данная команда не будет успешна, т.к. для cd будет передано 2 аргумента: cool и things, а не один.

Для решения этой задачи надо либо экранировать пробел:

- **С помощью кавычек: cd 'cool things'**
- **С помощью слеша: cd cool\ things**

Либо по возможности избегать пробелов в названиях файлов.

Также имена файлов в Linux чувствительны к регистру.

- Умейте формировать относительные и абсолютные пути к файлам и каталогам файловой системы GNU/Linux.
- Умейте изменять текущий рабочий каталог с использованием любых путей.

Тема «Получение информации о командах из стандартных руководств»

- Каким образом интерпретатор Bash обрабатывает команды, переданные пользователем?

После ввода команды в эмулятор терминала и нажатия клавиши Enter строка передаётся в командный интерпретатор

Общая структура простого запроса интерпретатору

команда --аргумент1 аргумент2 аргумент3

Команда определяет действие, которое вы хотите выполнить, а аргументы уточняют поведение

- Какие типы команд может обрабатывать интерпретатор Bash? Как можно определить тип команды? В чём их принципиальное отличие друг от друга?

Типы:

Встроенные команды интерпретатора (builtin)

Функции интерпретатора (function)

Псевдонимы интерпретатора (alias)

Исполняемый файл (file)

Определить так:

$ type cd

cd — это встроенная команда bash

Описание:

builtin

Данные команды направлены для управления состоянием интерпретатора, а не на прямое взаимодействие с внешним окружением. Например, cd

function

Вызов функции

alias

Вызов приложения по псевдониму

file

Вызов исполняемого файла

- Как можно прочитать документацию для встроенных команд интерпретатора?

help <команда>

- Объясните типичные схемы кодирования различных вариантов вызова команды, принятой в стандартной документации. Приведите примеры.

$ help cd

cd: cd [-L|[-P [-e]] [-@]] [каталог]

Change the shell working directory.

Первая строка кратко описывает какие аргументы можно передавать команде

Аргументы описывают детали действия, которое мы хотим выполнить над приложением

[-@] указывает, что аргумент -@ является опциональным: его можно указывать, а можно и не указывать

| указывает на то, что есть 2 взаимоисключающих варианта

Порядок аргументов может быть важен! Нельзя указать каталог до других аргументов

У команды cd нет обязательных аргументов

После текстового описания команды идёт блок с описанием каждой возможной опции

```
Options:
-L force symbolic links to be followed: resolve symbolic
 links in DIR after processing instances of `..'
-P use the physical directory structure without following
symbolic links: resolve symbolic links in DIR before
processing instances of `..'
-e if the -P option is supplied, and the current working
directory cannot be determined successfully, exit with
a non-zero status
```

По аналогии, можно ещё привести примеров.

- Каким образом можно прочитать документацию, встроенную в приложение?

<alias приложения> --help или же <путь к исполняемому файлу> --help

Остального нет в вопросе, но 100% это спросит:

Для удаления псевдонима используется команда unalias

При завершении сессии, при старте новой сессии состояние сбрасывается на начальное и все команды, которые были определены с помощью alias сбросятся

Чтобы эти псевдонимы были доступны всегда, их надо описать внутри конфигурационного файла bash, который считывается при каждом старте интерпретатора:

~/.bashrc

- Для чего используется приложение man? Какова связь между приложениями man и приложением-пейджером less?

Приложение man используется для просмотра документации к приложению. man использует less для вывода информации в командный интерфейс.

- Какие существуют разделы man-руководств? Как открыть руководство из целевого раздела?

- 1 Исполняемые программы или команды оболочки (shell)
- 2 Системные вызовы (функции, предоставляемые ядром)
- 3 Библиотечные вызовы (функции, предоставляемые программными библиотеками)
- 4 Специальные файлы (обычно находящиеся в каталоге /dev)
- 5 Форматы файлов и соглашения, например о /etc/passwd
- 6 Игры
- 7 Разное (включает пакеты макросов и соглашения), например man(7), groff(7)
- 8 Команды администрирования системы (обычно, запускаемые только суперпользователем)
- 9 Процедуры ядра [нестандартный раздел]

man [параметры man] [[раздел] страница ...] ...

Просмотр информации о приложении passwd

$ man 1 passwd

passwd(1) Пользовательские команды

НАЗВАНИЕ

passwd - изменяет пароль пользователя

- Как можно выполнять поиск информации по доступным локально man-руководствам?

Для поиска руководств можно воспользоваться:

- приложением apropos: apropos passwd
- приложением man: man -k passwd

- Какие возможности предоставляют псевдонимы Bash? Как их использовать?

Возможность вызывать команды не через путь к исполняемому файлу, а через вызов заданного псевдонима. Использование: alias <имя>=”<команда>” После этого необходимо просто вызвать <имя>

- Умейте получать текст документации на все типы команд.
- Умейте получать знания из текстов документации на разные типы команд.

Тема «Использование текстовых редакторов Vim и GNU Emacs»

- Какое применение существует у текстовых редакторов в командном интерфейсе?
  - Предназначены для работы с текстовыми файлами. Они позволяют просматривать содержимое текстовых файлов и производить над ними различные действия: вставку, удаление и копирование текста и т.д. Консольные редакторы менее ресурсозатратные, чем редакторы с графической оболочкой, также более отказоустойчивы.
- Какие существуют текстовые редакторы?
  - Vi, Vim, GNU Emacs, Nano
- Опишите ключевые отличия между текстовыми редакторами emacs, nano и vim.
  - nano — текстовый редактор, нацеленный на предоставление понятного пользовательского интерфейса для пользователей графических текстовых редакторов. Содержит некоторое количество горячих клавиш.
  - vim — текстовый редактор, предоставляющий самобытный интерфейс. Интерфейс основан на введении понятия «режим». В разных режимах пользователь выполняет различные действия: просматривает файл, редактирует файл, работает с визуальным выделением. Приложение поддерживает плагины, позволяющие превратить его в хороший текстовый редактор для программистов. В некоторых компаниях является корпоративным стандартом.
  - emacs — текстовый редактор, поведение которого можно программировать на функциональном языке программирования, диалекте LISP. Также поддерживает расширение поведения путём установки дополнений. Каждый пользователь донастраивает его под свои задачи и требования.
- Умейте использовать один из текстовых редакторов emacs, nano или vim в рамках консольного интерфейса.

Тема «Перенаправление потоков ввода-вывода, построение конвейеров»

- Какие потоки данных доступны приложению после его запуска? Какое у них предназначение?

0, stdin — стандартный поток ввода

1, stdout — стандартный поток вывода

2, stderr — стандартный поток вывода сообщений об ошибках

Стандартные потоки ввода-вывода самый распространенный и удобный способ для обмена информацией потоков

- Где пользователь может увидеть или повлиять на содержимое данных потоков в командном интерпретаторе?

Можно увидеть с помощью команд ps, top, htop. Повлиять можно, например, удалив процесс

- Какие возможности у пользователя есть по перенаправлению стандартного потока вывода?

> - можем направить поток вывода команды в определенный файл, также можем создать с помощью данного потока вывода пустой файл

&> - все потоки в один файл

&>> - все потоки в конец файла

- Какие возможности у пользователя есть по перенаправлению стандартного потока вывода сообщений об ошибках?

2> /dev/null - избавимся от ошибок, отправив их вникуда

2>> - запишем ошибки в файл

- Какие возможности у пользователя есть по перенаправлению стандартного потока ввода?

Для перенаправления потока ввода у процесса надо использовать следующий синтаксис:

КОМАНДА < input.txt

В результате поток ввода будет команды будет получать данные из файла input.tx

Вместе с cat его можно использовать для чтения файла: $ cat < input.txt

- Как можно объединять выходные потоки данных? Как можно перенаправить потоки вывода в файл?

Данный механизм называется pipe, конвеер

В Bash для связи потоков двух процессов используется оператор |

- Каким образом следует использовать конвейеры для объединения работы нескольких приложений? Какие типичные задачи можно решать с помощью

КОМАНДА-1 | КОМАНДА-2

В результате будут запущены КОМАНДА-1 и КОМАНДА-2, причём стандартный вывод КОМАНДА-1 будет подключён ко входу в конвеер, а выход конвеера подключён к стандартному входу программы КОМАНДА-2

Пример : `ls -l / | grep tmp`. - команда `ls -l /` Выведет содержимое корневого каталога. Дальше данные поступают команде `grep tmp`, которая произведет поиск по входным данным (полученным из 1 команды). После чего команда `grep` выведет то, что нашла на экран

Сколько приложений можно объединить с помощью конвейеров? Есть ли какие-либо ограничения?

Их число бесконечно, зависит только от мощности ПК, у которого, конечно, будут какие-то ограничения

- Перечислите назначение типовых фильтров sort, uniq, wc, head, tail, grep, sed, awk.

sort -утилита для различной сортировки

uniq - предназначена для поиска одинаковых строк в массивах текста. При этом с найденными совпадениями пользователь может совершать множество действий — например, удалять их из вывода либо наоборот, выводить только их.

wc- команда для подсчета строк или слов в тексте

head - выводит начальные строки (по умолчанию — 10) из одного или нескольких документов

tail - она позволяет выводить заданное количество строк с конца файла

grep - Grep это утилита командной строки Linux, который даёт пользователям возможность вести поиск строки. С его помощью можно даже искать конкретные слова в файле. Также можно передать вывод любой команды в grep, что сильно упрощает работу во время поиска

sed -это потоковый редактор текста, работающий по принципу замены. Его можно использовать для поиска, вставки, замены и удаления фрагментов в файле. С помощью этой утилиты вы можете редактировать файлы не открывая их.

awk -Команда awk позволяет работать с утилитой для обработки строк. Например, эта утилита может извлечь лишь второй столбец из текстового файла с данными в табличном формате.

- Умейте перенаправлять потоки ввода-вывода в файлы.
- Умейте формировать конвейеры произвольной сложности.

Тема «Права доступа к файлам и каталогам»

- Что означают термины многозадачная и многопользовательская ОС? Объясните связь этих терминов друг с другом.

Многозадачная ОС

ОС позволяет выполнять или имитировать выполнение нескольких задач одновременно с помощью одного процессора

- Достигается за счёт переключения процессора между активными задачами быстрее чем человек может воспринять
- У задач может быть приоритет в зависимости от которого происходит выбор очередной задачи для исполнения
- Переключение контекста между задачами де-факто снижает эффективность работы вычислительной системы

Многопользовательская ОС

Позволяет запускать процессы нескольким пользователям одновременно

- Локальный пользователь с графическим интерфейсом
- Удалённые пользователи с текстовым интерфейсом
- Системные пользователи для работы служб

Рост производительности и усовершенствование технологий позволяют сейчас решать одновременно определенное число задач без потери скорости работы, однако в один момент времени только один пользователь может воспользоваться компьютером, поэтому часто мощности ПК простаивают.

- Какие требования предъявляются к многопользовательским ОС?

В многопользовательской среде необходимо решить следующие задачи

- Каждый пользователь должен управлять своими данными, другие пользователи по умолчанию не должны иметь возможность их изменять
- Данные пользователя в оперативной памяти также должны быть защищены от стороннего наблюдателя
- Обычный пользователь не должен иметь возможности влиять на настройку системы, т.к. это может повредить другим пользователям этой системы
- Пользователи должны иметь возможность работать над общими данными

Сфера разграничения прав

Требования необходимо обеспечить для:

- Файлов на файловой системе
- Данных различных процессов

- Опишите модель обеспечения безопасности файловой системы GNU/Linux: права пользователей и права каталогов файловой системы.

В UNIX-мире для формирования модели безопасности используются
- User ID, уникальный идентификатор пользователя
- Group ID, уникальный идентификатор группы

Пользователь

- Каждому пользователю обязательно назначается его уникальный идентификатор, который не может быть передан другим пользователям
- Каждому пользователю обязательно назначается «группа по умолчанию»
- Пользователь также может входить в произвольное количество групп

Файл на жёстком диске

- Файл принадлежит конкретному пользователю, его уникальному идентификатору
- Файл относится к конкретной группе, её уникальному идентификатору

- Каким образом можно получить информацию о правах текущего пользователя?

Для определения прав пользователя можно использовать приложение id
$ id --help
Использование: id [КЛЮЧ]… [ПОЛЬЗОВАТЕЛЬ]

Выводит информацию о пользователе и группе для заданного ПОЛЬЗОВАТЕЛЯ, или о текущем пользователе (если ничего не указано).

$ id
uid=1001(andrey) gid=1001(andrey) группы=1001(andrey),27(sudo)
- uid — уникальный идентификатор текущего пользователя и назначенное ему имя
- gid — уникальный идентификатор основной группы пользователя и её имя
- groups — список дополнительных групп, которым принадлежит пользователь

- Каким образом можно получить информацию о правах произвольного пользователя? Перечислите приложения и местоположение локальных конфигурационных файлов.
-

id [OPTIONS] [USERNAME]

Пример:
id mark
uid=1001(mark) gid=1001(mark) groups=1001(mark),998(docker)

Если имя пользователя не указано, команда id отображает информацию о текущем вошедшем в систему пользователе.

При вызове без какой-либо опции id печатает реальный идентификатор пользователя (uid), реальный идентификатор основной группы пользователя (gid) и реальные идентификаторы дополнительных групп (groups), к которым принадлежит пользователь. Действующий идентификатор пользователя, идентификатор группы и идентификаторы дополнительных групп печатаются только в том случае, если они отличаются от реальных.

$ passwd -S пользователь (дополнение, не знаю, надо ли это)

- **-S** - отображает информацию об аккаунте
- Первое поле - имя пользователя
- Второе поле показывает одно из значений: **P** - пароль установлен, **L** - пользователь заблокирован, **NP** - пароля нет.
- **07/21/2016** - дата последнего изменения пароля.
- **0** - минимальное время до смены пароля
- **99999** - максимальное время действия пароля
- **7** - за сколько дней нужно предупреждать об истечении срока действия пароля
- **-1** - через сколько дней пароль нужно деактивировать.
-

Информация о пользователях содержится в файлах

- /etc/passwd содержит описание локальных пользователей
- /etc/group содержит список групп и входящих в них пользователей
- /etc/shadow содержит пароли локальных пользователей

    - Как узнать права доступа к файлам в GNU/Linux?

ls -l

Пример:
$ ls -l test
-rw-r--r-- 1 andrey andrey 0 окт 5 14:55 test

- первый столбец описывает атрибуты файла
- второй столбец — количество физических ссылок
- третий столбец — владелец файла
- четвертый столбец — группа файла

Атрибуты файлов

- первый элемент описывает тип файлов
- далее группами по 3 символа описываются права доступа к файлу

Типы файлов

- - — обыкновенный файл

- b — специальный файл блочного устройства, например жёсткого диска
- c — специальный файл символьного устройства, например клавиатуры или терминала
- d — каталог
- l — символическая ссылка, не имеет собственных прав доступа
- n — сетевой файл
- p — файл FIFO для организации взаимодействия потоков
- s — файл socket для организации псевдосетевого взаимодействия

Права доступа
Последняя часть атрибутов описывает права доступа

Права для файлов
- r позволяет читать содержимое файлов
- w позволяет переписывать содержимое файлов, но не позволяет их удалять или переименовывать
- x позволяет выполнять запуск файла, скриптовые файлы должны быть также доступны на чтение

Права для каталогов
- r просматривать содержимое каталогов, если ещё установлен флаг x
- w позволяет создавать, удалять, переименовывать файлы в каталоге, если ещё установлен флаг x
- x позволяет пользователю заходить внутрь каталога

Порядок прав
Права записываются в порядке: владелец-группа-остальные
- -rw-r--r-- — владелец может читать и записывать в файл, группа может считывать из файла, все остальные могут считывать из файла
- -rwx------ — владелец может выполнять любые операции, все остальные, включая группу не могут ничего делать с файлом
- lrwxrwxrwx — файл является ссылкой, которую могут прочитать все, но работа с файлом будет происходит по его эффективным разрешениям

- Каким образом соотносятся текстовые права доступа с их представлением в восьмеричной системе исчисления?

Восьмеричное	Бинарное	Режим
0	000	—
1	001	–x
2	010	-w-
3	011	-wx
4	100	r–

5	101	r-x
6	110	rw-
7	111	rwx

- Какие есть подходы к изменению прав текущего пользователя?

Изменение прав доступа

- Для изменения прав доступа к файлу или каталогу применяется приложение chmod
- Изменить права на файл может только его владелец
- Для описания прав можно воспользоваться цифровой или буквенной нотацией

Пример:
$ > test
$ ls -l test
-rw-r--r-- 1 andrey andrey 0 окт 5 15:46 test
$ chmod 600 test
$ ls -l test
-rw------- 1 andrey andrey 0 окт 5 15:46 test

Так же права доступа можно задавать буквенно

Буквенная нотация

При описании буквенной нотации надо придерживаться следующего формата:
[ugoa...][[-+=][perms...]...

- На первом месте указывается для кого надо применить права. Если не указать, то применяется для всех (u - владелец файла; g - группа файла; o - все остальные пользователи)
- Затем указывается что мы хотим сделать с правами
    - - убрать права
    - + выдать права
    - = установить права
- В конце указывается список разрешений
- Можно указать различные разрешения через запятую

Изменение прав с буквенной нотацией

$ > words
$ ls -l words
-rw-r--r-- 1 andrey andrey 0 окт 5 16:03 words
$ chmod u+x words
$ ls -l words
-rwxr--r-- 1 andrey andrey 0 окт 5 16:03 words
$ chmod go-r words
$ ls -l words

-rwx------ 1 andrey andrey 0 окт 5 16:03 words

- Числовая нотация обычно короче
- Числовая нотация непонятна «неинициированным»
- Буквенная нотация позволяет изменять только лишь нужные части прав доступа, не меняя остальные

Изменение владельцев файлов

Для изменения владельца и группы для файлов используется приложение chown
> chown --help
Использование: chown [ПАРАМЕТР]… [ВЛАДЕЛЕЦ][:[ГРУППА]] ФАЙЛ…
или: chown [ПАРАМЕТР]… --reference=ОФАЙЛ ФАЙЛ…
Смена владельца и группы указанного ФАЙЛА на ВЛАДЕЛЬЦА и/или ГРУППУ.

Аргумент	Результат
`bob`	изменить только владельца у целевого файла на пользователя bob
`bob:users`	изменить владельца на bob, изменить группу на users
`:admins`	изменить только группу на admin, владелец не меняется
`bob:`	изменить владельца на bob, изменить группу на группу по умолчанию пользователя bob

- Какие есть подходы к запуску приложений с правами другого пользователя?

Приложение su

Приложение позволяет либо выполнить одну команду от имени другого пользователя, либо открыть интерактивную оболочку с его правами
$ su --help
Usage:
su [options] [-] [<user> [<argument>...]]

- - указывает, что необходимо загрузить параметры оболочки для интерактивного входа, чтобы окружение совпадало с целевым
- user имя целевого пользователя, root если не было указано
- С помощью флага -c можно передать нужную команду на исполнение
  $ su -c 'ls /root/*'
  Пароль:
  /root/empty-file
- Расширение пути будет выполнено с правами суперпользователя

Приложение sudo

Если вы хотите выполнить программу не от имени root, а от имени другого пользователя, то необходимо использовать опцию -u, например:
$ sudo -u sergiy whoami

- Опишите назначение приложений chmod, chown, su, sudo, passwd, adduser, deluser и т.д.

Как я понимаю, данные приложения чаще всего необходимы системному администратору для создания новых пользователей, управлением их правами доступа и редактированием этих же пользователей. (данное предложение является неподтвержденной мыслью, можете дальше прочесть описание программ и сделать свои выводы)

$ chmod опции права /путь/к/файлу

(подробное описание было в вопросе: “Какие есть подходы к изменению прав текущего пользователя”)

$ chown пользователь опции /путь/к/файлу

(подробное описание было в вопросе: “Какие есть подходы к изменению прав текущего пользователя”)

Приложение su

Приложение позволяет либо выполнить одну команду от имени другого пользователя, либо открыть интерактивную оболочку с его правами

$ su --help

Usage:

su [options] [-] [<user> [<argument>...]]

- - указывает, что необходимо загрузить параметры оболочки для интерактивного входа, чтобы окружение совпадало с целевым
- user имя целевого пользователя, root если не было указано

Получение прав суперпользователя:

$ su -

Пароль:

#

Необходимо интерактивно ввести пароль суперпользователя

Работа от имени суперпользователя

Работать с правами суперпользователя удобно: система не мешает своими сообщениями, что что-то пошло не так, все команды завершаются успешно

Однако при таком подходе снимаются все защитные механизмы, которые нужны, чтобы предотвратить изменения

Запуск одной команды

С помощью флага -c можно передать нужную команду на исполнение

$ su -c 'ls /root/*'

Пароль:

/root/empty-file

$

Расширение пути будет выполнено с правами суперпользователя

Использование sudo

Приложение sudo работает во многих аспектах как команда su, однако отличается наличием конфигурационного файла

- Для каждого пользователя может быть указан список команд, которые он может выполнить от имени другого пользователя
- Для части команд запрос пароля может быть отключён
- Для запуска команды может потребоваться как пароль текущего пользователя, так и пароль целевого пользователя
- Конфигурация находится в файле /etc/sudoers/ и каталоге /etc/sudoers.d
- Если для выполнения команд требуется ввод пароля, тогда его надо будет ввести 1 раз в 5 минут или до истечения сессии

Настройка sudo в GNU/Debian

В GNU/Debian создаваемый пользователь не указан в конфигурации sudo и не может ей пользоваться, однако там находится группа sudo
Для добавления вашего пользователя в группу sudo выполните

1. Зайдите в интерактивную сессию под пользователем root
2. Выполните добавление: usermod -a -G sudo USER, где USER — имя вашей учётной записи
3. Закройте сессии от имени суперпользователя и пользователя, затем подключитесь заново. При следующем входе пользователю присвоятся корректные группы

После такой конфигурации пользователь сможет с помощью sudo выполнять любые команды

Использование sudo

$ sudo ls /root
[sudo] пароль для andrey:
empty-file
$ sudo ls /root
empty-file

- sudo ls /root — запустить ls с правами root для просмотра каталога /root
- sudo -l — узнать какие привилегии доступны текущему пользователю
- sudo -i [-u user] — запустить интерактивную сессию под целевым пользователем
- sudo -K — завершить сессию

Приложение passwd

Для смены пароля пользователя можно воспользоваться приложением passwd
$ passwd --help
Использование: passwd [параметры] [ПОЛЬЗОВАТЕЛЬ]
Обычный пользователь может сменить только свой пароль
Суперпользователь может изменить пароль любого пользователя

Другие приложения

- useradd, низкоуровневая утилита для добавления пользователя в систему
- adduser, более удобное для использования приложение для добавления пользователей в систему

- groupadd, низкоуровневое приложение для добавления групп
- userdel
- deluser
- groupdel
- usermod
- groupmod

- Умение узнавать текущие права пользователя: названия, идентификаторы пользователя и групп.
- Умение узнавать текущие права файлов: владелец, группа, атрибуты доступа.
- Умение изменять права доступа к файлам и использованием строковых и числовых подходов.
- Умение изменять текущего пользователя без необходимости авторизации. Правила использования приложений su, sudo.

Тема «Написание скриптов на языке Bash»

- Опишите предназначение механизма shebang в GNU/Linux.
- Как запускать скрипты с явным и неявным указанием интерпретатора? В каких случаях следует применять каждый из способов?
- Какое основное назначение у написания скриптов на языке Bash?
- Какие есть альтернативы к написанию скриптов кроме Bash? В каких случаях следует использовать альтернативные языки программирования?
- Опишите процесс обработки строк в Bash. Опишите расширения путей и порядок их применения при обработке строк в Bash.
- Опишите синтаксис определения переменных, присваивания им значений и получение значений переменных.
- Опишите синтаксис условного оператора, циклических операторов Bash. Укажите ключевые особенности проверки на правдивость и ложность в рамках данных операторов.
- Опишите синтаксис для доступа к аргументам скриптов и функций на языке Bash?
- Каким образом можно проверить качество скриптов на языке Bash?
- Умение оформлять скрипты на языке Bash для решения произвольных задач.
- Умение отлаживать работу скриптов на языке Bash.

Тема «Управление процессами в GNU/Linux»

- Что такое процесс? Чем отличается процесс от приложения?
  - Процесс - это приложение на этапе выполнения, работающее под управлением операционной системы, выделенные ему ресурсы в определённый момент исполнения. Каждый процесс имеет уникальный

идентификатор процесса, PID. Процессы получают доступ к ресурсам системы (оперативной памяти, файлам, внешним устройствам и т. п.) и могут изменять их содержимое. Доступ регулируется с помощью идентификатора пользователя и идентификатора группы, которые система присваивает каждому процессу.
○ Приложение (примерно если спросит) - это последовательность инструкций для выполнения процессором.

- Откуда процессы берутся в операционной системе?

Во время загрузки ОС сначала запускается ядро ОС, которое подготавливает оборудование для функционирования и свои внутренние структуры для запуска приложений. Затем ядро запускает процесс под названием init, который уже ответственен за запуск системных служб. Многие из этих служб продолжают работу в фоновом режиме и выключаются только вместе с завершением операционной системы.
Одной из таких служб является служба входа (текстового или графического) в систему. Когда пользователь успешно авторизуется в системе, то служба уже запускает процессы для его работы.

В рассмотренной схеме всегда участвуют два процесса: один запускает, а второй запускается. В рамках этого взаимодействия первый называется процессом-родителем, а второй процессом-ребёнком.

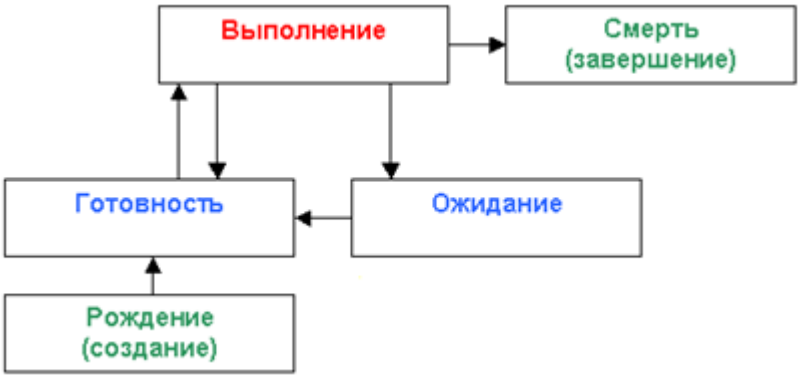
- Какими правами обладают запущенные процессы? Откуда берутся права у новых процессов?

Каждый процесс запускается от имени определённого пользователя и группы. Т.е. каждому процессу присваивается уникальный идентификатор пользователя и группы. В зависимости от этих значений процесс будет иметь соответствующие права по взаимодействию с файловой системой и другими частями ОС.

- Может ли процесс изменить (расширить или уменьшить) список доступных прав?

Процесс может сменить свой UID, если запустит вместо себя при помощи `exec()` другую программу из файла. Это происходит, например, при использовании приложения su для смены пользователя

- В каких состояниях может находиться процесс? Для каких целей используются разные состояния процессов? Как процесс переходит между этими состояниями?



Рождение процесса - это пассивное состояние, когда самого процесса еще нет, но уже готова структура для появления процесса. Смерть процесса - самого процесса уже нет, но может случиться, что его "место", то есть структура, осталась в списке процессов. Такие процессы называются «зомби».
Выполнение - это активное состояние, во время которого процесс обладает всеми необходимыми ему ресурсами. В этом состоянии процесс непосредственно выполняется процессором.
Ожидание - это пассивное состояние, во время которого процесс заблокирован, он не может быть выполнен, потому что ожидает какое-то событие, например, ввода данных или освобождения нужного ему устройства.
Готовность - это тоже пассивное состояние, когда процесс заблокирован по внешним причинам, например, когда он ожидает своей очереди на выполнение.

- Какими ресурсами может обладать процесс?

Процесс обладает ресурсами системы: память, время процессора, файлы, внешние устройства.

- Каким образом происходит выделение ресурсов процессам: кто инициирует, кто координирует?

Главный родительский процесс init. Все процессы создаются от него или от его потомков.

- Что такое сигнал? Кто может формировать сигналы процессам? Какие задачи решают с помощью сигналов процессам?

Сигнал— асинхронное уведомление процесса о каком-либо событии, один из основных способов взаимодействия между процессами. Когда сигнал послан процессу, операционная система прерывает выполнение процесса, при этом, если процесс установил собственный обработчик сигнала, операционная система запускает этот обработчик, передав ему информацию о сигнале, если процесс не установил обработчика, то выполняется обработчик по умолчанию.

Посылать сигналы могут: пользователь (Ctrl + C), ядро системы (при ошибках), другие процессы.
С помощью сигналов процессам их можно приостанавливать, продолжать и завершать (**Kill**).

- Опишите назначение приложений и типичные сценарии использования ps, top, kill, killall.

Приложение **ps** позволяет получить информацию о процессах, которые запущены в момент вызова приложения. Не интерактивное. Может быть очень полезным если система перегружена и необходимо срочно узнать запущенные процессы linux, чтобы освободить память или ресурсы процессора.
Интерактивные средства не всегда могут помочь, потому что они потребляют слишком много ресурсов. С другой стороны **ps** дает большую гибкость, поскольку утилита имеет множество опций и параметров.
В отличие от предыдущего приложения **top** является интерактивным, его можно сравнить с диспетчером задач в Windows. Он периодически опрашивает состояние системы и отображает активные процессы.
Приложение **kill** позволяет «убивать» процессы, в услугах которых мы больше не нуждаемся. Для его использования необходимо знать название процесса или его уникальный идентификатор.
Приложение **killall** позволяет отправлять сигналы процессам по имени.

- Какие возможности у Bash для работы с фоновыми процессами?

Bash предоставляет возможности для организации одновременной работы со множеством приложений в рамках одной сессии Bash-терминала. Это достигается благодаря помещению ряда процессов в фон и переводом некоторых процессов из фона на передний план. С точки зрения графического интерфейса это можно сравнить с работой за мобильным телефоном - только одно приложение может выводить информацию на экран, а остальные помещаются в фон. Чтобы запустить команду в фоновом режиме необходимо в конце команды поставить знак амперсанда **&.**

- Опишите назначение команд bg, fg, jobs.

Мы можем посмотреть список работающих в фоне команд с помощью встроенной команды **jobs**
Для возвращения процесса из фона на передний план, например для завершения, необходимо использовать команду **fg**

Для помещения текущего процесса из переднего плана в фон необходимо нажать сочетание клавиш **Ctrl+z**. После этого приложение **приостановит свою работу** и будет помещено в фон.

Для разрешения работы приложения в фоне можно воспользоваться командой **bg**. Она принимает такие же аргументы, как и **fg**. Это имеет смысл для приложений, выполняющих вычисления, чтобы они продолжили свою работу. Это также разумно для приложений с графическим интерфейсом.

- Умение просматривать список текущих процессов с помощью моментального просмотра ps и в интерактивной форме top.
- Умение посылать сигналы процессам по их названию, уникальному номеру и т.д.
- Умение приостанавливать работу процессов с помощью командного интерпретатора Bash. Умение восстанавливать работу процесса в фоновом и в основном процессах.

Тема «Управление пакетами с помощью инструментов dpkg и apt»

- Опишите типичные схемы доставки приложений в операционных системах семейства Windows, Android, iOS, macOS.

схема поставки приложений в Windows - файлы с расширением .exe

В Android основной канал передачи приложений - Google Play

iOS основной канал поставки - AppStore. В macOS встроенный в систему MacAppStore

- Какое назначение у подсистемы пакетного менеджера в рамках ОС GNU/Linux?

Она позволяет конечным пользователям управлять приложениями на своей системе: устанавливать, удалять, обновлять и т.д.

- Какие пакетные менеджеры существуют? Что между ними общее? Какие отличия?

Категории пакетных менеджеров

Высокоуровневые менеджеры. Применяются для поиска и скачивания пакетов из репозиториев. В процессе работы могут задействовать низкоуровневые менеджеры для инсталляции загруженных программ.

Низкоуровневые менеджеры. Используются для установки локальных пакетов, загруженных вручную пользователем, или высокоуровневым пакетным менеджером.

Общее у них - они позволяют управлять процессом установки, удаления, настройки и обновления различных компонентов программного обеспечения.????

Отличия-видно по определениям.

- Какое назначение несут репозитории? Кто может предоставлять репозиторий?

Репозиторий представляет собой хранилище пакетов для дистрибутива.

Производители дистрибутивов предоставляют для каждого дистрибутива свои репозитории, которые включают тысячи собранных пакетов

Производители ПО могут предоставлять свои репозитории с новыми версиями

Пакетный менеджер может работать со множеством репозиториев одновременно

Пакетный менеджер может обновлять пакеты, если в репозитории появились более свежие версии

Официальные репозитории могут содержать подразделы:

Ввиду патентных ограничений

Ввиду ограничений на лицензию ПО

- Из каких частей состоит пакет? Что такое зависимости пакета, для чего они применяются?

Пакетный файл является архивом из файлов

- Большая часть из них — это файлы для установки в файловую систему
- Оставшиеся — это различные скрипты, выполняющиеся на различных этапах установки пакета: до установки, в разные моменты установки, после установки

Дополнительно пакет содержит мета-информацию, включающую название пакета, его версию, краткое описание и т.д.

Зависимости – список дополнительных пакетов и библиотек, участвующие в работе программы.

При установке пакета будут установлены все его зависимости, и зависимости зависимых пакетов, и …

Собственно зависимости используются для установки библиотек, интерпретаторов, возможно других исполняемые файлов,необходимых(зачастую) исполняемому файлу для работы приложения

- Укажите на способы поиска пакетов в пакетном менеджере APT и дистрибутиве Debian GNU/Linux.

Поиск пакета(в пакетном менеджере APT)

Для поиска пакета по имени или его краткому описанию можно использовать

apt-cache search TEXT

apt search TEXT

Вместо TEXT необходимо вводить искомое слово, обычно название пакета совпадает с названием приложения, которое надо установить.

Поиск пакета на packages.debian.org

Дистрибутив Debian предлагает графический инструментарий для поиска пакетов https:/packages.debian.org

Поиск пакетов по категориям

Поиск пакетов по имени

Поиск пакетов по файлам, входящих в пакеты

Поиск с помощью передачи аргумента

Помимо взаимодействия с графическим интерфейсом можно сформировать запросы в адресной строке:

По названию пакета: https://packages.debian.org/PACKAGE. Вместо PACKAGE надо написать название пакета

По файлам: https://packages.debian.org/file:PATH. Вместо PATH надо указать путь к файлу, который ищется

- Каким образом можно установить пакет?

Из репозитория

Для установки пакета из репозитория необходимо выполнить команду apt-get install PACKAGE или apt install PACKAGE. Вместо PACKAGE необходимо указать имя пакета для установки

Внимание если вы давно не обновляли локальный кеш пакетов, то его необходимо обновить с помощью apt update

Из файла

Если вы хотите поставить .deb-пакет не из репозитория, то необходимо

1. Скачать данный пакет на компьютер
2. Установить с помощью низкоуровнего инструмента dpkg: dpkg -i PATH, где вместо PATH указать путь к .deb-пакету

Если пакету не хватает зависимостей, то его установка завершится с ошибкой

- Каким образом происходит настройка пакетного менеджера APT?

Все файлы настроек APT расположены в директории /etc/apt.

apt.conf(единственная проблема-я его не нашел на своей виртуальной машине,однако все сайты,где есть хоть какая-то инфа про настройки apt говорили,что он должен быть.Самое упоротое- man apt.conf работает исправно и выдает нужную документацию.Такие дела…..)

/etc/apt/apt.conf – это основной файл настроек, который применяется всеми инструментами из состава APT. Документация содержит исчерпывающее описание всех доступных настроек и опций:

man apt.conf

apt.conf.d

Эта директория включает в себе файлы настроек, схожие по синтаксису с apt.conf. Именно эта директория позволяет легко и просто работать с настройками APT, добавлять или удалять файлы с настройками.

auth.conf

В этом файле содержатся ключи для авторизации в репозиториях. В том числе в него добавляются логины и пароли от репозиториев к приобретенным программам.

sources.list

Файл, содержащий список репозиториев.

sources.list.d

Директория, содержащая файлы репозиториев, схожих с sources.list по назначению. Каждый репозиторий описывается в своем файле.

- Какие разделы репозиториев есть в Debian GNU/Linux?

В случае с DEBIAN, у репозиториев есть 3 ветви(раздела):

1. **Главный(main):** Раздел, в котором хранятся все пакеты, включенные в официальный дистрибутив DEBIAN, которые бесплатны в соответствии с Руководством по свободному программному обеспечению DEBIAN. Официальное распространение DEBIAN полностью состоит из этого раздела.
2. **Вклад (вклад)(contrib):** Раздел, в котором хранятся пакеты, создатели которых предоставили им бесплатную лицензию, но у них есть зависимости от других несвободных программ, то есть программного обеспечения с открытым исходным кодом, которое не может работать без проприетарных элементов. Эти элементы могут быть программным обеспечением из раздела non-free или проприетарными файлами(несвободное программное обеспечение (англ. *proprietary software*; от *proprietary* — частное, патентованное, в составе собственности), такими как игровые ПЗУ, BIOS для консолей и т. Д.
3. **Несвободно(non-free):** Раздел, в котором хранятся пакеты с некоторыми обременительными условиями лицензии, ограничивающими их использование или распространение.

- Умение использовать пакетный менеджер apt для установки пакетов из репозитория.
- Умение использовать систему пакетов dpkg для установки и поиска пакетов, для поиска соответствия между файлами и пакетами.

Тема «Система инициализации и управления службами systemd»

Опишите процесс загрузки ОС GNU/Linux.

Основные этапы загрузки:

1. Системная прошивка компьютера выполняет первичную проверку и инициализацию аппаратного обеспечения.
2. В случае BIOS прошивка загружает в оперативную память и выполняет загрузочный код с одного из разделов заданного загрузочного устройства, который содержит фазу 1 загрузчика Linux. Фаза 1 загружает фазу 2 (значительный по размеру код загрузчика). Некоторые загрузчики могут использовать для этого промежуточный этап (под названием фаза 1,5), поскольку современные диски большого объёма могут некорректно считываться без дальнейшего кода. В случае UEFI запускается загрузчик загруженный со служебного раздела (EFS), который выбирается согласно настройкам приоритета загрузки определенного в энергонезависимой памяти компьютера. При этом возможна загрузка не только специализированного загрузчика, но можно загрузить и непосредственно ядро Linux (для этого ядро должно быть собрано с опцией EFI_STUB).
3. Загрузчик зачастую предлагает пользователю меню с доступными вариантами загрузки. После выбора или после заданного тайм-аута загрузчик загружает ядро.
4. Загруженное ядро распаковывается в памяти, настраивает системные функции, такие как работа необходимого оборудования и управление страницами памяти, после чего делает вызов start_kernel().
5. После этого start_kernel() выполняет основную настройку системы (прерывания, остальные функции управления памятью, инициализацию устройств, драйверов и т. д.), а потом порождает процесс бездействия, диспетчер и отдельно от них — процесс init (выполняющийся в пользовательском пространстве).
6. Планировщик начинает более эффективно управлять системой, в то время как ядро переходит к бездействию.
7. Процесс init выполняет необходимые сценарии, которые настраивают все службы и структуры, не относящиеся к уровню ядра, в результате чего будет создано пользовательское окружение, и пользователю будет предоставлен экран входа в систему.

Когда происходит завершение работы, init вызывается для управляемого закрытия программ пользовательского уровня, тоже согласно сценариям. После этого init закрывается, а ядро производит своё собственное завершение работы.

Какие существуют системы инициализации? Назовите их преимущества и недостатки.

1. SYSTEM V INIT
System V или SysV - это довольно старая, но до сих пор ещё популярная система инициализации Linux и Unix подобных операционных систем. Она была основой для создания многих других систем инициализации, а также первой коммерческой системой инициализации разработанной для Unix в AT&T. Она была разработана еще в 1983 году.
Почти все дистрибутивы Linux изначально использовали SysV. Исключением была только Gentoo, в которой использовалась собственная система инициализации и Slackware, с инициализацией в стиле BSD.
Основные возможности SysV:

- Написание файлов запуска служб на bash;
- Последовательный запуск служб;
- Сортировка порядка запуска с помощью номеров в именах файлов;
- Команды для запуска, остановки и проверки состояния служб.

Никакой параллельной загрузки, системы зависимостей, запуска по требованию и автоматического запуска здесь не было и в помине.

С момента ее разработки прошло много лет и из-за некоторых недостатков были разработаны другие системы для ее замены, они хоть и имели новые функции и были более эффективны, но они были по-прежнему совместимы с SysV.

2. OPENRC

OpenRC - это система инициализации Linux и Unix подобных операционных систем совместимая с Sys V Init и поддерживающая систему зависимостей во время запуска. Она приносит некоторые улучшения в SysV, и как и другие системы инициализации Linux, совместима с ней, но вы должны иметь в виду, что OpenRC не заменяет полностью файл /sbin/init. Эта система инициализации используется в Gentoo и дистрибутивах BSD.

Кроме стандартных возможностей SysV, здесь поддерживается также:

- Поддержка зависимостей служб;
- Поддержка параллельного запуска служб;
- Поддерживает настройку в одном отдельном файле;
- Работает как демон;

По сравнению с SysV тут появилось много новых возможностей, но все еще не все те, что нужны для оптимальной работы системы.

3. SYSTEMD

Systemd - это новая система инициализации Linux. Она была введена по умолчанию в Fedora 15, а сейчас используется почти во всех популярных Linux дистрибутивах. Это не только инициализирующий процесс, поддерживающий огромное количество возможностей, но и набор инструментов для управления службами и этими возможностями из системы. Основная цель - иметь полный контроль над всеми процессами во время их запуска и на протяжении всего выполнения.

Systemd очень сильно отличается от всех существующих систем инициализации, тем как она работает с сервисами, и даже конфигурационными файлами сервисов. Совместимости со скриптами SysV нет, их нужно преобразовать в linux systemd unit файлы.

Вот ее основные особенности:

- Понятный, простой и эффективный дизайн;
- Параллельная загрузка служб на основе зависимостей;
- Поддерживается завершение дополнительных процессов;
- Поддерживается собственный журнал с помощью journald;
- Поддерживается планирование заданий с помощью таймеров Systemd;
- Поддерживается управление сетью с помощью networkd;
- Для управления DNS используется systemd-resolved;
- Хранение журналов в бинарных файлах;
- Сохранение состояния сервисов linux systemd для возможного восстановления;
- Улучшенная интеграция с Gnome;
- Запуск сервисов по требованию;

4. RUNINIT

Runinit - это кроссплатформенная система инициализации, которая может работать в GNU Linux, Solaris, BSD и MacOS. Это отличная альтернатива для SysV с поддержкой мониторинга состояния служб.

Здесь есть некоторые интересные особенности, которых нет в других системах инициализации:

- Полный контроль сервисов, каждый сервис привязывается к своему каталогу;
- Надежное средство журналирования и ротации логов;
- Быстрая система загрузки и выключения;
- Портативность;
- Легкое создание файлов конфигурации служб;
- Небольшое количество кода системы инициализации.

5. UPSTART

Upstart - это система инициализации на основе событий, разработанная в Canonical и призванная заменять SysV. Она может запускать системные службы, выполнять над ними различные задачи, инспектировать их во время выполнения, а также выполнять нужные действия в ответ на события в системе.

Это гибридная система инициализации, она использует как SysV скрипты запуска, так и файлы служб Systemd.

Вот ее самые заметные особенности:

- Изначально разработанная для Ubuntu, но может использоваться и в других дистрибутивах;

- Запуск и остановка служб на основе событий;
- Генерация событий во время запуска и остановки служб;
- События могут быть отправлены обычными процессами;
- Связь с процессом инициализации через DBus;
- Пользователи могут запускать и останавливать свои процессы;
- Перезапуск служб, которые неожиданно завершились;
- Параллельная загрузка сервисов;
- Автоматический перезапуск служб;

Большинство ее возможностей работают благодаря интеграции с системой инициализации Systemd. В последнее время всё меньше используются скрипты SysV init и всё больше применяются юнит файлы Systemd. Рано или поздно Systemd вытеснит и полностью заменит Upstart в Ubuntu.

Какие типичные стадии инициализации присутствуют в systemd?

1.Первое, что делает компьютер при включении питания - это инициализация BIOS. BIOS считывает настройки загрузочного устройства, находит и передает права управления системой в MBR (при условии, что жесткий диск является первым загрузочным устройством).
2.MBR считывает соответствующую информацию из загрузочной программы Grub или LILO и инициализирует ядро. Затем Grub или LILO продолжат загрузку системы. Если вы укажете systemd в качестве программы управления загрузкой в файле конфигурации grub, последующий процесс загрузки будет завершен systemd. Systemd использует «цель» для обработки процесса загрузки и управления службами. Эти «целевые» файлы в systemd используются для группировки различных загрузочных модулей и запуска процесса синхронизации.
3.Первая цель, которую выполняет systemd, - это**default.target**. Но на самом деле default.target указывает на**graphical.target**Мягкая ссылка. Программная ссылка в Linux аналогична ярлыку в Windows. Фактическое расположение файла Graphical.target - /usr/lib/systemd/system/graphical.target. Содержимое файла graphical.target показано на скриншоте ниже.
4.На этом этапе начнется**multi-user.target**И эта цель помещает свои собственные субъединицы в каталог "/etc/systemd/system/multi-user.target.wants". Эта цель устанавливает системную среду для многопользовательской поддержки. Пользователи без полномочий root будут включены на этом этапе процесса загрузки. На этом этапе также будут запущены службы, связанные с межсетевым экраном.
"multi-user.target" передаст управление другому уровню "**basic.target**”。
5.Модуль «basic.target» используется для запуска общих служб, особенно служб управления графикой. Он использует каталог /etc/systemd/system/basic.target.wants, чтобы определить, какие службы будут запущены, а затем basic.target передаст управление**sysinit.target**.
6."Sysinit.target" запустит важные системные службы, такие как монтирование системы, пространство подкачки памяти и устройства, дополнительные параметры ядра и т. Д. sysinit.target будет передан в**local-fs.target**. Содержимое этого целевого блока показано на скриншоте ниже.
7.Local-fs.target, этот целевой модуль не будет запускать службы, связанные с пользователем, он только обрабатывает базовые базовые службы. Эта цель будет выполнять связанные операции в соответствии с / etc / fstab и / etc / inittab.

Какие команды systemcl используются для настройки стадий инициализации ОС? Как можно перейти между данными стадиями?

Ниже представлена Таблица 1, в которой идет сравнение всех таргетов systemd со старыми уровнями выполнения (runlevel) в SystemV. Псевдонимы таргета systemd предоставляются systemd для обратной совместимости. Псевдонимы таргета разрешают скриптам — и многим сисадминам, мне в том числе — использовать такие SystemV команды как init3 для изменения уровней выполнения. Конечно, команды SystemV направлены systemd для интерпретации и исполнения.

SystemV Runlevel	systemd target	systemd target aliases	Description
	halt.target		Приостанавливает систему без отключения питания
0	poweroff.target	runlevel0.target	Приостанавливает систему и отключает питание
S	emergency.target		Однопользовательский режим. Сервисы не запущены; файловые системы не смонтированы. Это самый базовый уровень оперирования. Для взаимодействия пользователя с системой в главной консоли запущена только аварийная оболочка.
1	rescue.target	runlevel1.target	Базовая система, включающая монтирование файловой системы с самым базовым набором сервисов и rescue оболочкой в главной консоли.
2		runlevel2.target	Многопользовательский режим, без NFS, но все сервисы, не относящиеся к GUI, запущены.
3	multi-user.target	runlevel3.target	Все сервисы запущены, но только через интерфейс командной строки (CLI).
4		runlevel4.target	Не используется.
5	graphical.target	runlevel5.target	Многопользовательский режим с GUI.

6	reboot.target	runlevel6.target	Перезагрузка.
	default.target		Этот таргет всегда имеет симлинк с multi-user.target или graphical.target. systemd всегда использует default.target для запуска системы. default.target никогда не должен быть связан с halt.target, poweroff.target или reboot.target.

Настройка состояния системы (уровень запуска) с помощью целей
Целями являются специальные файлы модулей, которые описывают состояние системы или точку синхронизации. Как и другие модули, файлы, которые определяют цели, могут быть идентифицированы по суффиксу, которым в данном случае является .target. Цели сами по себе немного значат, а используются для группировки других модулей.
Их можно использовать, чтобы привести систему в определенные состояния, подобно тому, как другие системы инициализации используют уровни запуска. Они используются в качестве справки, когда доступны определенные функции, позволяя вам указывать желаемое состояние вместо необходимости использования отдельных модулей для получения этого состояния.
Например, swap.target используется для указания того, что переключение готово к использованию. Модули, являющиеся частью этого процесса, могут синхронизироваться с этой целью путем указания в своей конфигурации, что они WantedBy= или RequiredBy= swap.target. Модули, которым требуется возможность переключения, могут указывать это состояние с помощью спецификаций Wants=, Requires= и After= для указания характера их отношений.

Каким образом происходит настройка служб в systemd?

Основополагающая цель системы инициализации заключается в инициализации компонентов, которые должны запускаться после загрузки ядра Linux (традиционно называются компоненты пользовательского пространства). Система инициализации также используется для управления службами и демонами для сервера и в любой момент времени работы системы. С учетом этого мы начнем с нескольких базовых операций по управлению службами.
В systemd целью большинства действий являются «модули», являющиеся ресурсами, которыми systemd знает, как управлять. Модули распределяются по категориям по типу ресурса, который они представляют, и определяются файлами, известными как файлы модулей. Тип каждого модуля можно вывести из суффикса в конце файла.
Для задач по управлению службами целевым модулем будут модули службы, которые имеют файлы модулей с суффиксом .service. Однако для большинства команд по управлению службами вы можете не использовать суффикс .service, поскольку systemd достаточно умна, чтобы знать, что вы, возможно, хотите работать со службой при использовании команд по управлению службами.

Какие типы конфигурационных файлов есть в systemd?

Systemd использует скомпилированные бинарные файлы. Все конфигурационные файлы открыты. Их можно менять через командную строку или GUI. При необходимости можно добавить свои конфигурационные файлы.

systemd-sysvcompat
Пакет **systemd-sysvcompat** (зависимость пакета **base**) содержит традиционный бинарный файл **init**. В системах под управлением systemd init — символическая ссылка на исполняемый файл systemd.
Кроме того, в этом пакете находятся 4 команды **SysVinit** — **halt(8)**, **poweroff(8)**, **reboot(8)** и **shutdown(8)**. Это символические ссылки на systemctl, и их работа обусловлена логикой systemd. Подробнее см. **#Управление питанием**.
В systemd-системах отказаться от совместимости с System V можно либо задав **параметр загрузки** init= (см. **BBS#233387**), либо с помощью собственных аргументов команды systemctl.
systemd-tmpfiles — временные файлы
Утилита *systemd-tmpfiles* создает, удаляет и очищает непостоянные и временные файлы и каталоги. Она читает конфигурационные файлы из /etc/tmpfiles.d/ и /usr/lib/tmpfiles.d/, чтобы понять, что необходимо делать. Конфигурационные файлы в первом каталоге имеют приоритет над теми, что расположены во втором. Конфигурационные файлы обычно предоставляются вместе с файлами служб и имеют названия вида /usr/lib/tmpfiles.d/*программа*.conf. Например, демон **Samba** предполагает, что существует каталог /run/samba с корректными правами доступа. Поэтому пакет **samba** поставляется в следующей конфигурации:
/usr/lib/tmpfiles.d/samba.conf
D /run/samba 0755 root root
Конфигурационные файлы также могут использоваться для записи значений при старте системы. Например, если вы используете /etc/rc.local для отключения пробуждения от устройств USB при помощи echo USBE > /proc/acpi/wakeup, вместо этого вы можете использовать следующий tmpfile:
/etc/tmpfiles.d/disable-usb-wake.conf
```
#    Path              Mode UID  GID  Age Argument
w    /proc/acpi/wakeup    -   -    -    -  USBE
```

Как происходит настройка выполнения задач по расписанию?

Чтобы SystemD принял во внимание то, что у него есть сервис, который он должен запускать по таймеру, у сервиса должен быть соответствующий unit-файл (файл конфигурации службы для SystemD), он должен лежать в правильном месте, иметь расширение ".service" и быть доступным на чтение, а также специальный одноименный unit-файл, но с расширением ".timer", описывающий, соответственно, когда указанный сервис должен быть запущен. Также желательно, чтобы в один и тот же момент времени у того же самого сервиса не было соответствующего ему init-скрипта - скрипта, который запускала служба init (когда еще была актуальна), для избежания конфликтов - многие системные администраторы все еще по привычке лезут в каталог /etc/init.d/ в поисках этих самых скриптов. Освоив systemd.timer, вы вполне сможете отказаться от CRON и ANACRON - синхронного и, соответственно, асинхронного планировщиков под Linux, освободив системные ресурсы для чего-то более важного.
Итак приступим. Для примера создадим простенький скрипт на языке командного интерпретатора bash, который выполнит архивирование со сжатием домашнего каталога пользователя, например, begemot, который располагается в /home, а затем отправит архив на бэкап сервер, используя утилиту ncat. Не будем его больше ничем перегружать - он нам нужен только для демонстрации:
```
#!/bin/bash
tar -cz /home/begemot | ncat 10.1.14.72 7000
```
Что мы здесь видим: запускается bash, вызывает tar, который создает архив папки /home, перенаправляя вывод gzip'у для сжатия данных, который уже в свою очередь передает уже сжатый поток данных ncat'у в направлении порта 7000 сервера с IP 10.1.14.72. Чтобы вся это кухня заработала, на той стороне должен стоят такой же ncat и принимать это всё с 7000-ого порта, но это тема для другой статьи. Сейчас нам нужно создать unit-файл описывающий данный сервис. Условимся, что скрипт мы положили в /opt, назвали его home_backup и, конечно, не забыли дать ему право на исполнение командой
chmod +x /opt/home_backup
А вот и содержание самого unit-файла, который нужно положить в /lib/systemd/system с названием, например, home_backup.service
```
[Unit]
Description=Backup script for /home/begemot

[Service]
ExecStart=/opt/home_backup
```

Обратить внимание здесь можно разве что на директиву ExecStart, которая описывает, что будем запускать, Description - это простое описание, не более. Наконец, чтобы активировать механизм systemd.timer осталось создать unit-файл типа timer и положить туда же с тем же названием, сменив только расширение с ".service" на ".timer"

```
[Unit]
Description=timer for home_backup script

[Timer]
OnCalendar=02:50

[Install]
WantedBy=timers.target
```

Здесь стоит обратить внимание на директиву "OnCalendar", которая показывает, что запускать сервис надо "по календарю" и, поскольку, ничего иного не указано, в 02:50 ночи. Ничего - не значит ничего, на самом деле SytemD просто остальные переменные сам заменит на "*", означающую "всегда" - точно так же, как и в терминологии CRON. Иными словами всегда, но только в 02:50, а 02:50 встречается в сутках только раз, поэтому это задание будет выполняться каждый день в 02:50. Какие еще вариации можно туда писать? Приведем несколько примеров, описанных в документации к SystemD:

Пн-Вт,СБ и ВС в 00:00:00
Sat,Thu,Mon-Wed,Sat-Sun
Все Пн и Вс в 2016-ом в 01:23 и в 02:23
Mon,Sun 16-*-* 2,1:23
1-ого числа каждого месяца, но только если это среда в 00:00:00
Wed *-1
То же самое
Wed-Wed,Wed *-1
По средам 17:48:00
Wed, 17:48
15-ого октября 2016-ого в 01:02:03, но только, если это Вт, Ср, Чт, Пт, Сб
Wed-Sat,Tue 12-10-16 1:2:3
7-ого числа каждого месяца в 00:00
--7 0:0:0
Каждый год 15-ого октября в 00:00
*-10-15 00:00:00
Каждой 5-ое марта в 08:05:40
03-05 08:05:40
Ежедневно в 08:05:40
08:05:40
Ежедневно в 5:40
--* 05:40:00
В 2016-ом 3-его мая в 00:00
2003-03-05
Ежегодно 3-его мая в 00:00
03-05
Ежечастно в 0 минут
hourly
Ежедневно в 00:00
daily
Ежемесяно 1-ого числа в 00:00
monthly
Еженедельно в понедельник в 00:00
weekly
Ежегодно 1-ого января в 00:00
yearly
Тоже самое
annually

Как видите, очень гибко. Практически любую вариацию можно легко настроить. "OnCalendar=" также можно комбинировать с другими типами таймеров, например "OnActiveSec=" задает время в секундах, которое должно пройти после активации timer файла, чтобы единовременно запустился указанный сервис. "OnBootSec=" делает то же самое, но отправной точкой является момент времени, когда компьютер был заружен. Осталось только понять зачем нужна директива "WantedBy=timers.target". На сам сервис она никак не влияет - она нужна для того, чтобы при следующей перезагрузке, наш таймер поднялся автоматически. Поэтому командой

systemctl daemon-reload
перечитываем конфигурацию SystemD. Командой
systemctl enable home_backup.timer
говорим SystemD не забыть его включить при следующей перезагрузке. И наконец, перезагрузившись, или руками включаем таймер
systemctl start home_backup.timer
Всё! Systemd.timer будет отслеживать системные часы и по наступлению нужного момента запустит нужный сервис.
Как просматривать и управлять состоянием служб systemd?
Управление службами
Основополагающая цель системы инициализации заключается в инициализации компонентов, которые должны запускаться после загрузки ядра Linux (традиционно называются компоненты пользовательского пространства). Система инициализации также используется для управления службами и демонами для сервера и в любой момент времени работы системы. С учетом этого мы начнем с нескольких базовых операций по управлению службами.
В systemd целью большинства действий являются «модули», являющиеся ресурсами, которыми systemd знает, как управлять. Модули распределяются по категориям по типу ресурса, который они представляют, и определяются файлами, известными как файлы модулей. Тип каждого модуля можно вывести из суффикса в конце файла.
Для задач по управлению службами целевым модулем будут модули службы, которые имеют файлы модулей с суффиксом .service. Однако для большинства команд по управлению службами вы можете не использовать суффикс .service, поскольку systemd достаточно умна, чтобы знать, что вы, возможно, хотите работать со службой при использовании команд по управлению службами.
Запуск и остановка служб
Чтобы запустить службу systemd, используя инструкции в файле модуля службы, используйте команду start. Если вы работаете как пользователь без прав root, вам потребуется использовать sudo, поскольку это влияет на состояние операционной системы:

- sudo systemctl start application.service

Copy
Как мы уже упомянули выше, systemd будет искать файлы *.service для команд управления службами, так что команду можно легко ввести следующим образом:

- sudo systemctl start application

Copy
Хотя вы можете использовать вышеуказанный формат для общего администрирования, для ясности мы будем использовать суффикс .service для остальных команд, чтобы предельно четко выражать цель, над которой мы работаем.
Чтобы остановить работающую в данный момент службу, можно использовать команду stop:

- sudo systemctl stop application.service

Copy
Перезапуск и перезагрузка
Чтобы перезапустить работающую службу, можно использовать команду restart:

- sudo systemctl restart application.service

Copy
Если данное приложение может перезагрузить файлы конфигурации (без перезапуска), вы можете выдать команду reload для инициализации этого процесса:

- sudo systemctl reload application.service

Copy
Если вы не уверены, есть ли у службы функция перезагрузки своей конфигурации, можно использовать команду reload-or-restart. Это перезагрузит необходимую конфигурацию при наличии. В противном случае будет перезагружена служба для выбора новой конфигурации:

- sudo systemctl reload-or-restart application.service

Copy
Включение и отключение служб
Указанные выше команды полезны для запуска или остановки служб во время текущего сеанса. Чтобы дать команду systemd автоматически запускать службы при загрузке, их необходимо включить.
Для запуска службы во время загрузки используйте команду enable:

- sudo systemctl enable application.service

Copy
При этом будет создана символическая ссылка из системной копии служебного файла (обычно в /lib/systemd/system или /etc/systemd/system) в месте на диске, где systemd ищет файлы для автозапуска (обычно /etc/systemd/system/some_target.target.wants; что такое цель, мы рассмотрим далее в этом руководстве).
Чтобы отключить автоматический запуск службы, можно ввести следующее:

- sudo systemctl disable application.service

Copy
При этом будет удалена символическая ссылка, что укажет на то, что служба не должна запускаться автоматически.
Помните, что включение службы не запустит ее в текущем сеансе. Если вы хотите запустить службу и включить ее при загрузке, необходимо дать обе команды, start и enable.
Проверка статуса служб
Чтобы проверить статус службы в вашей системе, можно использовать команду status:

- systemctl status application.service

Copy
При этом вы получите статус службы, иерархию контрольных групп и первые несколько строк журнала.
Например, при проверке статуса сервера Nginx вы можете видеть следующий вывод:
Output
```
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; vendor preset: disabled)
   Active: active (running) since Tue 2015-01-27 19:41:23 EST; 22h ago
 Main PID: 495 (nginx)
   CGroup: /system.slice/nginx.service
           ├─495 nginx: master process /usr/bin/nginx -g pid /run/nginx.pid; error_log stderr;
           └─496 nginx: worker process
Jan 27 19:41:23 desktop systemd[1]: Starting A high performance web server and a reverse proxy server...
Jan 27 19:41:23 desktop systemd[1]: Started A high performance web server and a reverse proxy server.
```
Это дает вам хороший обзор текущего статуса приложения и уведомляет о наличии каких-либо проблем или необходимости выполнения каких-либо действий.
Также есть методы для проверки определенных статусов. Например, чтобы проверить, активен ли (работает ли) модуль в данный момент, можно использовать команду is-active:

- systemctl is-active application.service

Copy
Это вернет текущий статус модуля, который обычно active или inactive. Код выхода будет «0», если он активен, и результат будет проще парсить в скрипты оболочки.
Чтобы увидеть, включен ли модуль, можно использовать команду is-enabled:

- systemctl is-enabled application.service

Copy
Это выведет информацию о том, что служба enabled или disabled, и снова установит код выхода на «0» или «1» в зависимости от вопроса команды.
Третья проверка заключается в проверке того, находится ли модуль в состоянии сбоя. Это означает, что была проблема, которая запустила данный модуль:

- systemctl is-failed application.service

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

Темы семинаров

SSH

- Опишите назначение технологии SSH: как она устроена, какие задачи позволяет решать?
  - SSH - (Secure Shell — «безопасная оболочка») - это протокол удаленного управления компьютером с операционной системой Linux. В основном ssh используется для удаленного управления серверами через терминал. SSH-клиенты и SSH-серверы доступны для большинства сетевых операционных систем. SSH позволяет безопасно передавать в незащищённой среде практически любой другой сетевой протокол. Таким образом, можно не только удалённо работать на компьютере через командную оболочку, но и передавать по шифрованному каналу звуковой поток или видео (например, с веб-камеры). Также SSH может использовать сжатие передаваемых данных для последующего их шифрования, что удобно, например, для удалённого запуска клиентов X Window System.
- Где находится конфигурационный файл сервера OpenSSH? Какие ключевые настройки можно сделать на стороне сервера?
  - Настройки сервера SSH находятся в файле /etc/ssh/sshd_config. Основные настройки – это настройки подключения непосредственно к серверу, Например, можно поменять порт подключения, установить доступ только для определенного пользователя или ограничить доступ к root.
- Где находится конфигурационный файл клиента OpenSSH? Какие настройки логично поместить в данный файл?
  - Для каждого пользователя можно сделать отдельные настройки клиента SSH. Для этого используется файл **.ssh/config**. Можно настроить пользователя или порт по умолчанию, настроить псевдоним подключения к серверу.
- Где находятся пользовательские ключи клиента OpenSSH? Для чего они используются?
  - Каждая пара ключей состоит из открытого и закрытого ключа. Секретный ключ сохраняется на стороне клиента и не должен быть

доступен кому-либо еще. По умолчанию ключи располагаются в папке ~/.ssh/. SSH-ключ — безопасный способ соединения с сервером. Подключение по SSH с помощью ключа исключает риск, который связан с подбором и взломом пароля.

- Умение устанавливать SSH-сервер OpenSSH, выполнять его базовую настройку. Умение устанавливать SSH-клиент OpenSSH.
- Умение ограничивать доступ к SSH-серверу с помощью указания имён, групп пользователей, ключей.
- Умение описывать стандартные параметры подключения к SSH-серверам в конфигурационном файле клиента.

Мультиплексер Zellij

- Опишите назначение технологии терминального мультиплексера? Какие альтернативы есть для решения данных задач?
  - Терминальный мультиплексер - программа, которая позволяет одновременно работать с несколькими терминальными сессиями в одном окне. Можно рассматривать его, как текстовую реализацию оконного режима работы с приложениями. Это может быть удобно, например, при разработке веб приложений - в одном терминале может быть открыт редактор с кодом, а в другом запущен разрабатываемый веб-сервер. Альтернативы - При использовании unix с графической оболочкой использовать вкладки в терминале???
- Опишите ключевые сценарии работы с мультиплексером Zellij: работа с вкладками, панелями, переключение, изменение размера.
  - Интерфейс мультиплексора содержит контекстные подсказки, позволяющие взаимодействовать с меню управления. При входе в конкретное меню, в интерфейсе отображаются также дополнительные подсказки, позволяющие совершать конкретные действия - создавать вкладки, редактировать панели, перемещать панели и пр.
  - Ctrl + t - перейти к меню управления вкладками
    Из меню можно переключаться между вкладками, создавать вкладки, закрывать вкладки, переименовывать их, включить отправление команд во все панели на текущей вкладке (sync), и переключаться между недавними вкладками.
  - Ctrl + p - перейти к меню управления панелями
    Из меню можно переключаться между панелями на текущей вкладке (если дойти до края экрана, произойдёт переключение на следующую вкладку, если она существует), создавать новые панели, разбивать уже имеющиеся панели по горизонтали или вертикали, закрывать панели, переключить одну из панелей в полноэкранный режим (не закрывая остальные панели), переключать режим отображения границ между панелями (минималистичный или полный), переименовывать панели,

выносить панели в отдельное “окно” перед остальными панелями, а также встраивать их обратно.
    - Ctrl + n - перейти к меню изменения размеров панели
      В этом меню нажатием стрелки в определённую сторону можно увеличивать панель в эту сторону. Также можно увеличивать/уменьшать размер панели по двум осям сразу, используя клавиши +-
      Также размеры панелей можно менять как нажатием Alt + +-, не используя меню изменения размера.
    - Переключение между панелями можно выполнять не только из меню панели, но и при работе с определённой панелью нажатием Alt + стрелки или Alt + hjkl. Также, дойдя до левого или правого края экрана, можно переключаться между вкладками, не открывая соответствующее меню.
- Умение использовать мультиплексер Zellij: управление панелями, вкладками, размерами элементов, перемещение между элементами.
- Умение помещать сессию работы в фоновый режим и восстановление работы сессии.

Работа с файловыми хранилищами

- Опишите логическое устройство жёсткого диска: блочный уровень, разделы, файловые системы.
    - Блочные устройства представляют абстрактный интерфейс к диску. Программы могут легко адресовать место на диске, как последовательность блоков по 512 байт с произвольным доступом. Пользовательские программы могут использовать эти блочные устройства для взаимодействия с диском, не беспокоясь о том, что у это за диски.
    - Дисковые блочные устройства разделяются на блочные устройства меньшего размера, называемые разделами. Разделы создаются с помощью средства под названием fdisk, которое используется для создания и редактирования таблиц разделов, расположенных на каждом диске. Таблица разделов определяет, как именно разбито пространство на целом диске.
    - Файловая система (ФС) — это архитектура хранения данных в системе, хранения данных в оперативной памяти и доступа к конфигурации ядра. Она устанавливает физическую и логическую структуру файлов, правила их создания и управления ими
- Опишите назначение виртуальной файловой системы в GNU/Linux.
    -
- Какие элементы виртуальной файловой системы GNU/Linux соответствуют устройствам частей жёсткого диска?
    -

- Каким образом происходит подключение файловой системы на разделе жёсткого диска к виртуальной файловой системе? Как сделать подключение постоянным?
  - Для монтирования файловых систем используется команда
    - *$ mount файл_устройства каталог_назначения*
  - Для размонтирования используется команда
    - *$ umount точка_монтирования*
  - или
    - *$ umount файл_устройства*
  - Для того чтобы подключение было постоянным, а не сбрасывалось после перезагрузки, можно настроить автоматическое монтирование дисков
- В чём отличие подключения дисковых разделов с помощью fstab с systemd?
- Как произвести форматирование раздела жёсткого диска в необходимую файловую систему, например ext4 или exfat.
- Умение просматривать список устройств хранения данных.
- Умение просматривать список подключённых дисковых разделов.
- Умение подключать дисковые разделы к каталогам файловой системы GNU/Linux.
- Умение формировать постоянные подключения дисковых разделов с помощью /etc/fstab и systemd.

Веб-сервер nginx

- Опишите назначение веб-серверов, их типичные схемы применения.
-
- Веб-сервера изначально предназначены для обслуживания статических страниц, например HTML и CSS. Позднее появилась возможность переадресовывать запросы к серверам приложений, отвечающих за генерацию динамического содержимого путём выполнения кода. Nginx позиционируется производителем как простой, быстрый и надёжный сервер, не перегруженный функциями. Применение nginx целесообразно прежде всего для статических веб-сайтов и как обратного прокси-сервера перед динамическими сайтами
-
- Опишите структуру каталогов конфигурации веб-сервера nginx. Как происходит подключение и отключение опциональных частей конфигурации?
- Администрирование сервера nginx в основном заключается в настройке и поддержке его файлов конфигурации, которые находятся в папке /etc/nginx.
- /etc/nginx/nginx.conf – главный файл конфигурации nginx.
- /etc/nginx/conf.d - общие конфигурационные параметры. Местоположение подключаемых файлов определяется с помощью директивы include. В качестве параметра передаётся путь к файлу конфигурации.

- /etc/nginx/modules-available – модули, доступные для подключения к основной конфигурации
- /etc/nginx/modules-enabled - модули, которые надо включить в конфигурацию nginx. Содержит символические ссылки на файлы из каталога modules-available
- /etc/nginx/sites-available – конфигурации виртуальных хостов, т.е. каждый файл, находящийся в этом каталоге, содержит информацию о конкретном сайте – его имени, IP адресе, рабочей директории и т.д.
- /etc/nginx/sites-enabled – ссылки на конфигурации сайтов из sites-available, которые будут обслуживаться nginx.
- Как выглядит типичный сценарий по добавлению поддержки нового виртуального сервера в рамках nginx?
- Для описания виртуального сервера используется директива server внутри контекста http. Контекст http может содержать несколько таких блоков для описания различных серверов.

```
http {
  server {
    # Server configuration
  }
}
```

- Внутри блока server может быть указан IP адрес и порт, по которым сервер слушает запрос. Поддерживается как IPv4, так и IPv6 формат. В примере показан сервер, настроенный на ipадрес 172.0.0.1 и порт 8080

```
server {
 listen 127.0.0.1:8080;
 # Additional server configuration
}
```

- Если порт не указан явно, то используется стандартный. При отсутствии адреса, обрабатывается любой адрес. Если директива listen не указана, используется стандартный порт 80 или 8000 в зависимости от привилегий пользователя.
- Как описывать раздачу статических файлов с файловой системы?
- Одна из важных задач конфигурации nginx — раздача файлов, таких как изображения или статические HTML-страницы. Директива root указывает, в каком каталоге должен осуществляться поиск файла. Nginx формирует путь к файлу, добавляя текущий URI к пути, указанному в root.

```
server {
  root /var/www/data;

  location / {
  }

  location /images/ {
  }

  location ~ \.mp3 {
    root /www/media;
  }
}
```

Например, на запрос http://localhost:80/images/cat.jpg будет предоставлен файл /var/www/data/images/cat.jpg

- Как описывать обратное проксирование запросов к веб-приложениям?
- Иногда бывает нужно чтобы различные url запросы обрабатывались на разных серверах, но первоначально приходили на один сервер. Например вы пробросили порт на своем роутере на один веб-сервер в вашей внутренней сети. Но хотите чтобы каталог /xxx открывался на втором веб-сервере, а /yyy открывался на третьем, и не хотите пробрасывать порты на каждый web-сервер. Или же если помимо статического контента у вас имеется сервер приложения, к которому тоже нужно предоставить доступ. Когда nginx проскирует запрос, он посылает его на указанный проксируемый сервер, получает ответ и пересылает обратно клиенту. В качестве адреса «внутреннего» сервера может быть указано как доменное имя, так и ip-адрес. Данный адрес может включать в себя и конкретный порт

```
location /some/path/ {
    proxy_pass http://www.example.com/link/;
}
```

  Например, запрос /some/path/page.html будет перенаправлен на http://www.example.com/link/page.html. Также nginx позволяет конфигурировать заголовок запроса при пересылке, обращаться к не-HTTP серверам, буферировать ответы и т.д.
- Как организовать одновременную раздачу данных и с файловой системы и веб-приложением?
- Уметь устанавливать веб-сервер nginx из репозитория.
- Уметь подключать и отключать описания виртуальных серверов nginx.
- Уметь формировать описания виртуальных серверов nginx с возможностью настройки: обслуживаемого доменного имени, способа предоставления данных (с файловой системы, проскирование).

VPN-сервер OpenVPN

- Опишите назначение технологии VPN. Какие типичные задачи можно решать с помощью технологии VPN
  - Сеть VPN может скрыть ваш IP-адрес и местоположение, зашифровать ваши интернет-данные и трафик, объединять ресурсы (серверы и рабочие станции) компании в единую безопасную виртуальную сеть, созданную на базе Интернета. И теперь сотрудники, работающие удаленно (из дома или из другой страны) будут находиться как бы в общей сети своей компании. Сеть VPN подходит и для консолидации территориально разделенных офисов компании.
- Опишите архитектуру приложения OpenVPN: выделенный сервер, клиенты, протокол взаимодействия.
  - Сервер OpenVPN ПО сервера OpenVPN создает туннель внутри незащищенной сети, например, Интернета. Этот туннель обеспечивает

безопасный зашифрованный трафик между узлами — участниками обмена данными в сети OpenVPN.
    - Клиент OpenVPN ПО клиента OpenVPN устанавливается на все узлы, которым необходим защищенный канал передачи данных с сервером OpenVPN. При соответствующей настройке сервера OpenVPN возможна защищенная передача данных между клиентами OpenVPN, а не только между клиентами и сервером OpenVPN.
    - Внутри OpenVPN используются протоколы безопасности SSL (Secure Socket Layer – уровень защищенных сокетов) или TLS (Transport Layer Security – безопасность транспортного уровня), с помощью которых пользователи могут обмениваться ключами.
- Каким образом происходит настройка сервера OpenVPN? Какие ключевые параметры есть? Как настраивать список пользователей для сервера?
-
- Каким образом происходит настройка клиента OpenVPN?
    - scp имя_пользователя@айпи_адресс:путь_до_файла_на_сервера путь_до_файла_на_клиенте
    - sudo apt-get install openvpn
    - sudo openvpn –config /etc/openvpn/client1.ovpn
- Уметь устанавливать клиентские и серверные компоненты сервера OpenVPN.
    - Сервер: wget https://git.io/vpn -O openvpn-install.sh
    - Клиент: Описан выше
- Уметь настраивать серверную часть для обеспечения подключения ряда клиентских приложений.
    - Добавление:
    - sudo ./openvpn-install.sh
    - Добавить клиент
    - Удаление:
    - ./easy-rsa revoke имя_клиента
    - service openvpn restart
- Уметь настраивать клиенсткое приложение для подключения к серверной части. Умение автоматизировать подключение с помощью менеджера сетевых подключений NetworkManager.
    - https://www.tune-it.ru/web/leksa/blog/-/blogs/openvpn-klient-na-ubuntu-18--1

Синхронизация данных между компьютерами с помощью Syncthing

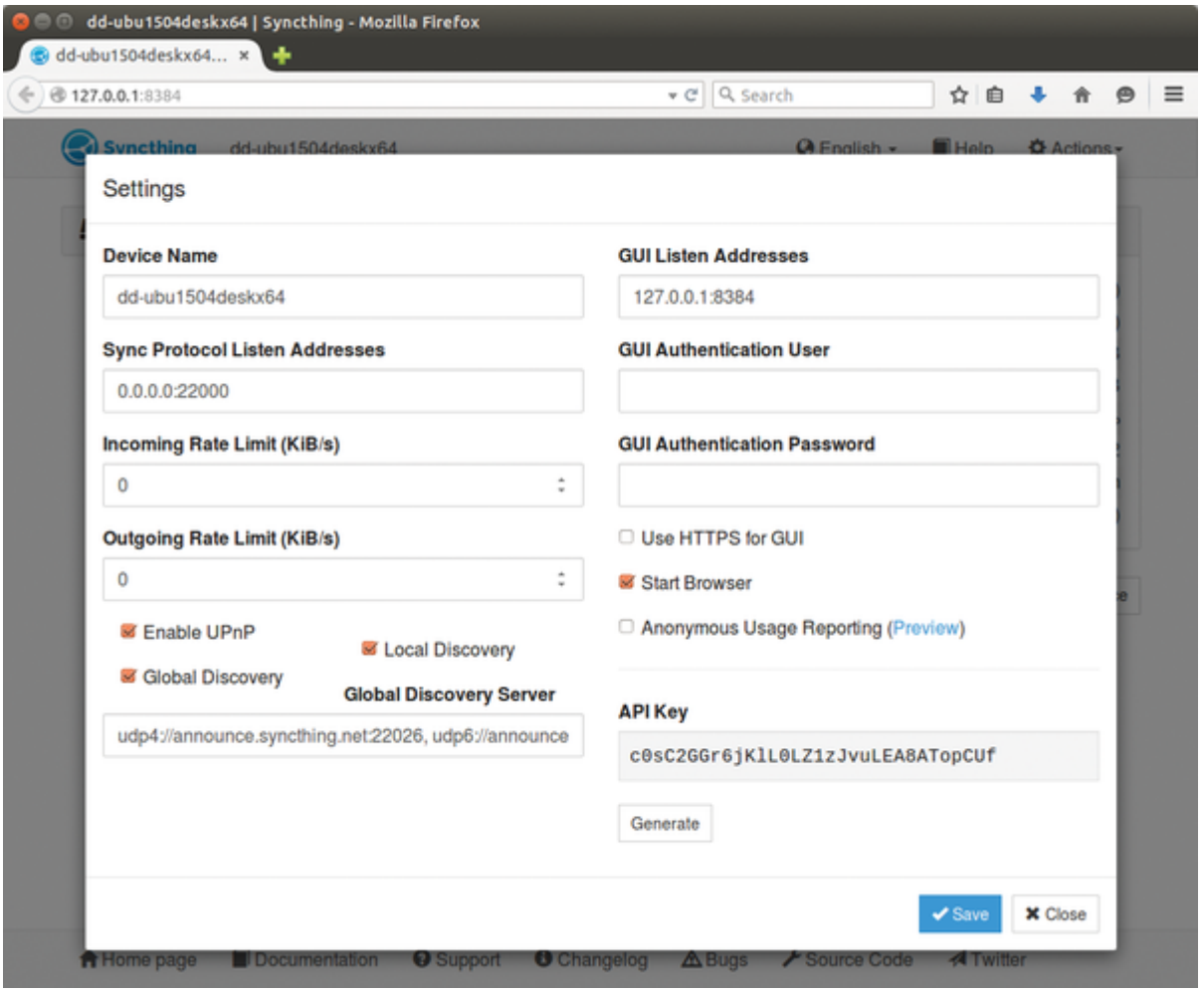
Почитать: http://rus-linux.net/MyLDP/admin/syncthing.html

- Опишите назначение технологии синхронизации данных syncthing.

  кроссплатформенное приложение, работающее по модели клиент-сервер и

предназначенное для синхронизации файлов между участниками (P2P) через протокол обмена блоками Block Exchange Protocol (BEP)

- Как происходит настройка приложения syncthing на компьютере? Как происходит связывание нескольких компьютеров друг с другом?
  Настройка через веб-интерфейс (`http://localhost:8384/`)
  При первом запуске Syncthing сгенерирует ключи и сертификаты, которые гарантируют безопасность обмена данными по сети. После этого веб-интерфейс приложения будет автоматически открыт с помощью используемого по умолчанию веб-браузера - Syncthing не имеет классического графического интерфейса
  На первом шаге нужно настроить приложение с помощью диалога настроек его веб-интерфейса, который доступен после нажатия на иконку шестеренки в правом верхнем углу веб-страницы



Далее нужно ввести уникальное имя устройства в поле ввода Имя устройства (Device Name); по умолчанию Syncthing использует в качестве имени устройства установленное имя узла. Также в данном подразделе вы можете ограничить пропускную способность канала, если вам нужно сохранить часть его пропускной способности для соединения с ресурсами сети Интернет. Используемое по умолчанию значение 0 позволяет использовать всю доступную пропускную способность канала. Если есть возможность, в данном разделе следует установить галочку Включить релеи (Enable UPnP), а также воспользоваться размещенными рядом с ней галочками для того, чтобы устройство отображалось на компьютерах из глобальной и локальной сетей. Дополнительно можете активировать поддержку протокола HTTPS и ввести имя пользователя с паролем

для ограничения доступа к веб-интерфейсу.

- Как происходит разрешение конфликтных ситуаций?
  Syncthing распознает конфликты. Когда файл был изменен одновременно на двух устройствах и его содержимое действительно различается, один из файлов будет переименован в <filename>.sync-conflict-<date>-<time>-<modifiedBy>.<ext>. Файл с более ранним временем модификации будет помечен как конфликтующий и, следовательно, будет переименован. Если время модификации одинаково, файл, исходящий от устройства, которое имеет большее значение первых 63 битов для своего идентификатора устройства, будет помечен как конфликтующий файл. Если возникает конфликт между модификацией и удалением файла, измененный файл всегда побеждает и возрождается без переименования на том устройстве, где он был удален. Имейте в виду, что файлы <filename>.sync-conflict-<date>-<time>-<modifiedBy>.<ext> обрабатываются как обычные файлы после их создания, поэтому они распространяются между устройствами. Мы делаем это, потому что конфликт обнаруживается и разрешается на одном устройстве, создавая файл sync-conflict, но такой же конфликт возникает везде, и мы не знаем, какой из конфликтующих файлов является «лучшим» от пользователя. точка зрения.
- Можно ли использовать данную систему как средство резервного копирования?
  Да. На нескольких серверах можно задать двустороннюю и одностороннюю синхронизацию, когда изменения на удаленных копиях не затронут оригинал.
- Уметь устанавливать приложение Syncthing на компьютер, формировать systemd-службу для его автоматического запуска.

```
  # Add the release PGP keys:
curl -s https://syncthing.net/release-key.txt | sudo apt-key add -
  # Add the "stable" channel to your APT sources:
echo "deb https://apt.syncthing.net/ syncthing stable" | sudo tee
/etc/apt/sources.list.d/syncthing.list
  # Update and install syncthing:
 sudo apt-get update
sudo apt-get install syncthing
  # формируем systemd
sudo nano /etc/systemd/system/syncthing@.service
```

- Уметь настраивать связь между рядом несколькими копиями Syncthing по ряду каталогов.
  (СМОТРИ ВОПРОС 2 ПРО НАСТРОЙКУ)

  После ввода настроек на всех используемых компьютерах следует связать эти компьютеры друг с другом. Индивидуальные идентификаторы устройств из 46 цифр, генерируемые на основе информации из сертификатов на каждом из устройств при первом запуске приложения, предназначены как раз для этой цели; они доступны посредством элемента меню настроек Показать ID (Show ID).Вы можете переместить этот идентификатор в форме текстового файла, расположенного на флеш-накопителе с интерфейсом USB, воспользоваться смартфоном с приложением для чтения QR-кодов (Рисунок 4), отправить ого в сообщении электронной почты или воспользоваться другими веб-сервисами. После этого придется ввести идентификатор интересующего устройства вместе с его именем, нажав на кнопку Добавить удаленное устройство (Add Device) в веб-интерфейсе приложения на другом устройстве. Функция Рекомендатель (Introducer) в диалоге добавления устройства позволяет указать, следует ли использовать все известные системе распределения данных узлы на новом

устройстве.

Вот пример из хабра, где юзер рассказывает как синхронизировал 3 отделения в городе https://habr.com/ru/post/350892/

Для начала вы должны запустить Syncthing на всех узлах (то есть, на всех компьютерах в сети)
на других отделениях Действия->Показать ID,копируем его
На основном сервере Добавить удаленное устройство (настраиваем там доступы к нужным каталогам)
на остальных отделениях разрешаем добавление устройства
Все синхронизировано

Видос : https://www.youtube.com/watch?v=OZw8ew8xMZM&ab_channel=Pulse

- Уметь решать конфликты синхронизации между разными системами.

Система резервного копирования kopia

- Опишите назначение технологии. Чем данная технология лучше или хуже ручного создания копий с помощью

Назначение kopia- Резервное копирование. Это процесс создания копии данных на носителе, предназначенном для восстановления данных.

Резервное копирование необходимо для возможности быстрого и недорогого восстановления информации в случае утери рабочей копии.

преимущества:

возможность подключения нескольких пользователей к репозиторию;

использование скользящего кэша;(хеш-функции , где ввод хешируется в окне, которое перемещается по вводу.Не знаю,нужно-ли,но всякий случай оставлю)

Дедупликация(специализированный метод сжатия массива данных, использующий в качестве алгоритма сжатия исключение дублирующих копий повторяющихся данных.);

после перемещения или переименования файлы не нужно загружать снова

- Каким образом происходит установка приложения на компьютер?

Добавляемключ к репозиторию:

$ curl -s https://kopia.io/signing-key | sudo gpg --dearmor -o /usr/share/keyrings/kopia-keyring.gpg

Добавляем репозиторий: $ echo "deb [signed-by=/usr/share/keyrings/kopia-keyring.gpg] http://packages.kopia.io/apt/ stable main" | sudo tee /etc/apt/sources.list.d/kopia.list

Обновляем список репозиториев:

$ sudo apt update

Устанавливаем приложения:

$ sudo apt install kopia

$ sudo apt install kopia-ui

- Как происходит создание хранилища для хранения резервных копий?

Создание хранилища:

$ kopia repository create filesystem --path /home/user/test/my-rep Необходимо придумать и запомнить пароль для репозитория. В случае утери пароля — доступ к хранилищу будет потерян!

Подключение к хранилищу:

$ kopia repository connect filesystem --path /home/user/test/my-rep Необходимо ввести пароль, придуманный во время создания репозитория.

- Как происходит добавление новых данных в хранилище резервных копий?

Создание резервной копии:

$ kopia snapshot create /home/user/test/files(с последующим добавлением в хранилище копии /home/user/test/files)

Просмотр списка резервных копий:

$ kopia snapshot list /home/user/test/files

Пример списка созданных резервных копий:

2022-05-14 17:39:58 MSK k0543d1e928522f175384b2ea1ecee3fe 0 B drwxrwxr-x files:2 dirs:1 (latest-3)

2022-05-14 17:43:53 MSK kb8f6e954269fe05ac37bc896114fa1b3 0 B drwxrwxr-x files:1 dirs:1 (latest-1..2,hourly-1,daily-1,weekly-1,monthly-1,annual-1)

Сравнение двух резервных копий:

$ kopia diff «id первой копии» «id второй копии»

- Как происходит извлечение данных из хранилища резервных копий?

Монтирование резервной копии в каталог файловой системы:

kb8f6e954269fe05ac37bc896114fa1b3 - id копии,которую мы извлекаем

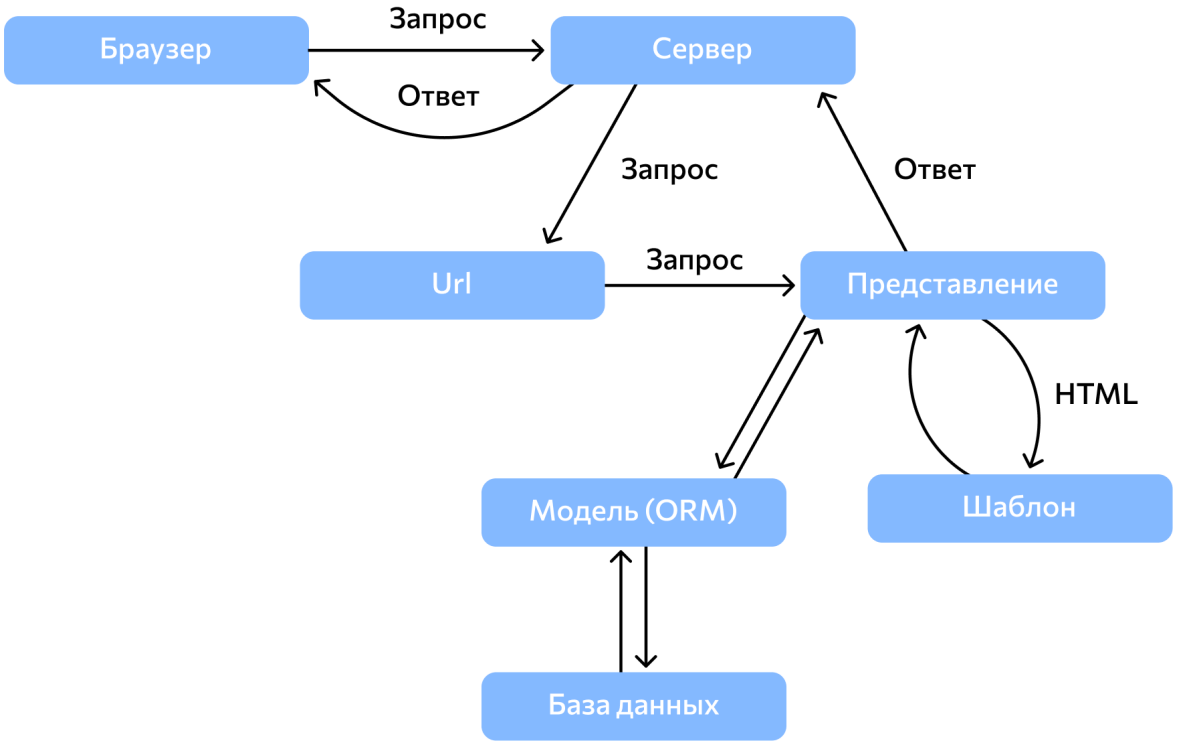
$ kopia mount kb8f6e954269fe05ac37bc896114fa1b3 /home/user/test/backs &

После монтирования резервной копии с файлами и каталогами станут доступны операции, доступные для файлов из файловой системы.

- Уметь устанавливать систему резервного копирования kopia.
- Уметь формировать хранилище для сохранения резервных копий данных.
- Уметь просматривать содержимое хранилище с резервными копиями, управлять резервными копиями.
- Уметь выполнять сохранение и восстановление из локальных хранилищ.
- Уметь выполнять сохранение и восстановление из хранилищ, доступных по протоколу SSH.

Развёртывание веб-приложения на языке Python

- Опишите принципиальную схему работы веб-приложений на языке Python.



- Что такое виртуальное окружение Python? Для чего оно используется при установке?

1. изолированное **окружение** среды, чтобы можно было для разных проектов использовать разные версии библиотек и их изменения друг друга не касались.

- Опишите процесс развёртывания веб-приложения Weblate с использованием python-пакетов из pypi.org.

1. установите Python, поставьте необходимые вам общие библиотеки(virtualenv, pip), создайте виртуальное окружение, запустите его, установите необходимые для проекта библиотеки (ставить в виртуальное окружение через `apt install` нельзя, только `pip install`), скопируйте настройки по умолчанию из `/weblate/settings_example.py` в `/weblate/settings.py`, настройте файл необходимым образом, далее команды по списку: `weblate migrate`, `weblate createadmin`, `weblate collectstatic`, `weblate compress`, `weblate/examples/celery start`, `weblate runserver`, подробнее тут `https://docs.weblate.org/ru/latest/admin/install/venv-debian.html`

- Уметь устанавливать пакеты для управления виртуальными окружениями Python.

1. apt install python3-virtualenv virtualenv
   - Уметь устанавливать пакеты для работы веб-приложения в виртуальное окружение Python.
1. Через pip install, например, pip install `pkgconfig, активируется venv командой` source **название_вашего_venv**/bin/activate

Уметь настраивать рабочие копии веб-приложений, зависимости которых установлены в виртуальное окружение.

- [ ] `в старой виртуальной среде - pip freeze > requirements.txt`
- [ ] `в новой виртуальной среде - pip install -r requirements.txt`
   - Уметь настраивать systemd-службы для автоматического запуска веб-приложений.

1. sudo nano /etc/systemd/system/weblate@.service
   - Уметь настраивать обратное проксирование nginx для работы веб-приложений.

1. https://routerus.com/how-to-install-nginx-on-ubuntu-18-04/
2. https://routerus.com/nginx-reverse-proxy/

Передача файлов по протоколу SFTP

- Опишите предназначение технологии. Какие технические средства обеспечивают взаимодействие по указанному протоколу?

интерактивная программа для передачи файлов, похожая на ftp, которая осуществляет все операции через зашифрованные средства передачи ssh. Предназначен для копирования и выполнения других операций с файлами поверх надёжного и безопасного соединения.

- Опишите преимущества и недостатки данной технологии по сравнению с другими средствами, например scp и syncthing.

-

- Каким образом можно ограничить доступ клиента к файловой системе на сервере на уровне указанного каталога?

весь пайплайн по ссылке - https://firstvds.ru/technology/ftp-sftp

- Как выполнять передачу отдельных файлов и каталогов с клиента на сервер и наоборот?

Копировать файл на устройство: put test.sh Downloads/
Выгрузить папку с устройство: get -r business-logs
Удалить файл на устройстве: rm archive-2009.png

- Как выполнять изменение рабочего каталога на стороне клиента и на стороне сервера?

cd - на удаленном
lcd - на локальном

- Как выполнить просмотр содержимого каталогов на стороне клиента и на стороне сервера?

ls - просмотр каталогов на удаленном
lls - просмотр каталогов на локальном

- Уметь устанавливать клиентские и серверные приложения из репозитория.

git clone ?

- Уметь ограничивать доступ клиентов к самой технологии передачи файлов на стороне сервера.

ограничить доступ по open-ssh

- Уметь ограничивать доступ клиентов к определённым частям файловой системы на стороне сервера.

весь пайплайн по ссылке - https://firstvds.ru/technology/ftp-sftp

- Уметь выполнять передачу отдельных файлов и каталогов с помощью технологий SFTP.

Копировать файл на устройство: put test.sh Downloads/
Выгрузить папку с устройство: get -r business-logs
Удалить файл на устройстве: rm archive-2009.png

Настройка Samba (Филиппов Александр ИВТ-31)

- Опишите предназначение технологии. В каких программных средах она реализована? Какие задачи позволяет решать?

Простыми словами, Samba это программное обеспечение, предназначенное для реализации файлового сервера, а начиная с 4-й версии и контроллера домена (аналог Active Directory). Файловый сервер работает по протоколу SMB/CIFS и позволяет предоставить доступ к общим ресурсам в локальной сети, например, текстовым файлам или сетевым принтерам. Пакет программ Samba имеет клиентскую и серверную части. В качестве клиентской части для подключения к файловому серверу может выступать программное обеспечение, работающее по протоколу SMB. В UNIX это samba-client, в Windows — проводник.

Samba так же позволяет реализовать множество кросс-платформенных служб, включая: • Аутентификацию и авторизацию • Сетевую печать • Преобразование имён • Анонсирование службы (просмотр ресурсов файлового сервера и общих принтеров)

SMB Protocol (Server Message Block Protocol) – сетевой протокол прикладного уровня, предназначенный для удаленного доступа к сетевым принтерам и файлам и другим сетевым ресурсам, а также для межпроцессорного взаимодействия. Первой версией протокола стал CIFS – Common Internet File System (единая файловая система интернета). В настоящее время протокол SMB тесно связан с операционными системами Microsoft Windows, где используется для реализации сетей Microsoft Windows и совместного использования файлов и принтеров.

Большинство функций Samba реализованы двумя демонами – smbd, nmbd. Демон smbd реализует обслуживание файлов и заданий на печать, а также аутентификацию и авторизацию. Демон nmbd обслуживает другие основные компоненты SMB: преобразование имён и анонсирование служб.

- Опишите преимущества данной технологии по сравнению с другими средствами передачи данных.

Главными отличиями от серверных версий Windows являются:

- отсутствие поддержки инфраструктуры узлов (sites) и репликации каталога в соответствии с настройками межузловых связей

Ещё одной особенностью следует считать то, что Samba работает только поверх TCP/IP, тогда как аналогичный сервис в Windows может предоставляться также поверх IPX и NetBEUI. Однако сама Microsoft в последних версиях Windows ориентируется на NBT, так что это отличие Samba неактуально.

По утверждениям ITLabs, в условиях многопользовательского доступа скорость работы в качестве ролей файлового и принт-сервера более чем в два раза выше по сравнению с Windows Server 2003 с теми же ролями

- Каким образом происходит настройка сервера Samba по раздаче файлов?

Samba настраивается в файле /etc/samba/smb.conf В этом файле указываются каталоги для совместного использования, их права доступа и общие рабочие параметры Samba. Для вывода списка всех параметров конфигурации Samba необходимо ввести команду testparm –v (данной командой также можно проверить правильность настройки файла smb.conf) После настроек конфигурационного файла и его проверки необходимо перезапустить демонов: sudo systemctl restart smbd sudo systemctl restart nmbd

- Каким образом происходит ограничение доступа к каталогам? Есть ли связь между Samba-пользователями или пользователями основной ОС?

Никогда не стоит забывать о безопасности использования файлов и других сетевых ресурсов. Чтобы обеспечить базовый уровень безопасности для типичной организации, необходимо сделать 2 вещи: • Явно указать, какие клиенты могут получить доступ к общим ресурсам SMBсервера. Эта часть конфигурации задаётся параметром host allow в файле smb.conf • Блокировать доступ к SMB-серверу за пределами организации. К сожалению, Samba использует только шифрование только для аутентификации и не использует шифрование для передачи данных. Во многих случаях придётся блокировать доступ извне, дабы пользователи случайно не загружали файлы в открытом виде, через интернет. Такая блокировка обычно осуществляется с помощью сил сетевого брандмауэра.

- Как выполнить просмотр состояния сетевых каталогов на Samba-сервере?

В пакет Samba и samba-client также входит утилита командной строки – smbclient, которая позволяет отображать список файлов конкретного сетевого ресурса. В нем также реализован FTP-подобный интерфейс для интерактивного доступа. К примеру, чтобы отобразить список доступных сетевых ресурсов для пользователя, необходимо ввести команду smbclient –L //server_name –U user_name а затем ввести пароль для данного пользователя. Чтобы подключиться к общему ресурсу и передать файлы, необходимо убрать флаг –L и указать имя этого ресурса smbclient //server_name/resource_name –U user_name

- Уметь устанавливать клиентские и серверные приложения из репозитория.

В Linux мы можем монтировать удаленный каталог с помощью команды mount, например:

```
mount -t cifs "//192.168.1.15/ad" /mnt -o user=dmosk
```

** где **192.168.1.15** — IP-адрес сервера; **mnt** — каталог, куда монтируем сетевую шару; **dmosk** — пользователь, под которым выполняем подключение к сетевому каталогу.*

*** в систему должен быть установлен пакет **cifs-utils**.*

SMB Browser

Также мы можем увидеть содержимое удаленных папок на samba при помощи клиента smb. Для начала установим данного клиента:

б) на Debian / Ubuntu / Mint:

```
apt-get install samba-client
```

После вводим команду:

```
smbclient -L 192.168.1.15 -U staff@dmosk.local
```

** где **192.168.1.15** — сервер samba, к которому мы пытаемся подключиться; **staff@dmosk.local** — учетная запись, под которой выполняется подключение.*

... мы получим список каталогов, которые расшарены на сервере.

Также мы можем подключиться к конкретной папке, например:

```
smbclient \\\\192.168.1.15\\ad -U staff@dmosk.local
```

Мы подключимся клиентом samba — можно выполнить запрос на показ содержимого:

```
smb: \> ls
```

Или полный список возможных команд:

```
smb: \> help
```

- Уметь настраивать список сетевых каталогов.

Установка и настройка Samba-сервер для Ubuntu выполняется следующими этапами.

Обновляем информацию о репозиториях и устанавливаем обновления для существующих пакетов в системе:

```
apt-get update && apt-get upgrade
```

Устанавливаем пакет Samba:

```
apt-get install -y samba samba-client
```

Создадим резервную копию файла конфигурации:

```
cp /etc/samba/smb.conf /etc/samba/smb.conf_sample
```

Создадим директории для файлов, например в каталоге /media:

```
mkdir /media/samba
```

Важно! По умолчанию, директория /media располагается в корне системы /, для нее редко создается свой раздел. По этой причине возможно переполнение корневого раздела. Во избежание этой неприятной ситуации, рекомендуем монтировать отдельный жесткий диск в /media/samba.

Создаем каталог для всех пользователей:

```
mkdir /media/samba/public
```

Изменим права доступа к каталогу:

```
chmod -R 0755 /media/samba/public
```

Также следует воспользоваться командой chown для смены владельца и/или группы.

Создаем директорию для ограниченного круга лиц:

```
mkdir /media/samba/private
```

С помощью системных инструментов создадим группу пользователей:

```
groupadd smbgrp
```

Добавляем пользователей Samba:

```
useradd user1
```

Созданных пользователей добавляем в группу:

```
usermod -aG smbgrp user1
```

Изменим группу, которой принадлежит приватная директория:

```
chgrp smbgrp /media/samba/private
```

С помощью инструментов Samba создадим пароль для добавленного пользователя:

```
smbpasswd -a user1
```

С помощью текстового редактора, например, nano, редактируем конфигурационный файл samba:

```
nano /etc/samba/smb.conf
```

Удаляем все строки из файла. Вставляем следующие:

```
[global]

workgroup = WORKGROUP

security = user

map to guest = bad user

wins support = no

dns proxy = no

[public]

path = /media/samba/public

guest ok = yes

force user = nobody

browsable = yes

writable = yes

[private]

path = /media/samba/private

valid users = @smbgrp

guest ok = no

browsable = yes

writable = yes
```

Сохраняем используя сочетание **Ctrl + X**, затем нажимаем Y и Enter.

Объясним значения строк. конфигурационный файл состоит из трех секций:

global - данная секция отвечает за общие настройки Samba-сервера;

public и **private** - секции описания настроек директорий общего доступа.

В секции **global** присутствуют пять параметров:

- **workgroup** - рабочая группа. Для упрощения работы пользователей WORKGROUP указывается, как группа по умолчанию. Если в вашей сети имя рабочей группы изменено, то следует изменить это значение и для Samba;
- **security** - уровень безопасности сервера. Значение user означает авторизацию по паре логин/пароль;
- **map to guest** - параметр определяет способ обработки запросов. Значение bad user означает, что запросы с неправильным паролем будут отклонены, даже если такое имя пользователя существует;
- **wins support** - включить или выключить поддержку WINS;
- **dns proxy** - возможность проксирования запросов к DNS.

Настройки директорий выполняются в соответствующих секциях:

path - полный путь до директории на жестком диске;

guest ok - возможность доступа к каталогу без пароля (гостевой);

browsable - показывать ли каталог (“шару”) на сервере среди прочих. Если параметр установлен как “no”, то доступ будет возможен по полному пути, например ip-addresshidden_directory;

force user - пользователь от которого ведется работа с каталогом. Для повышения безопасности сервера, обычно используют nobody. Главное, не использовать пользователя root - это небезопасно.

writable - установка значения как “yes” позволяет пользователю выполнять действия над файлами внутри каталога - переименование, добавление, удаление, перемещение в подкаталог и копирование;

valid users - список пользователей у которых есть доступ к каталогу. Если пользователей несколько, их имена указываются через запятую. Если необходим доступ для пользователей принадлежащих группе, перед именем группы устанавливается символ ”at” @ (“собака”).

Важно! Имя директории общего доступа, отображаемое пользователям, равно имени секции в которой оно описано.

Проверяем настройки с помощью команды:

```
testparm -s
```

Перезапускаем сервер:

```
service smbd restart
```

```
service nmbd restart
```

Настроим межсетевой экран. Для этого в правилах откроем TCP-порты 139 и 445, а также UDP-порты 137 и 138, но только для тех подсетей, которым доверяете. Для указания собственного диапазона адресов, замените значение после ключа “-s”:

```
iptables -A INPUT -p tcp -m tcp --dport 445 -s 10.0.0.0/24 -j ACCEPT
```

```
iptables -A INPUT -p tcp -m tcp --dport 139 -s 10.0.0.0/24 -j ACCEPT
```

```
iptables -A INPUT -p udp -m udp --dport 137 -s 10.0.0.0/24 -j ACCEPT
```

```
iptables -A INPUT -p udp -m udp --dport 138 -s 10.0.0.0/24 -j ACCEPT
```

Для сохранения правил и применения их после перезагрузки сервера следует воспользоваться пакетом iptables-persistent. Установим его:

```
apt-get install iptables-persistent
```

Входе установки пакета, программа предложит запомнить существующие правила iptables. Подтверждаем это действие.

Для проверки существующих правил используем:

```
iptables -L
```

- Уметь настраивать права доступа к сетевым каталогам.
- Уметь подключаться с помощью клиента smbclient к серверу.
- Уметь устанавливать постоянные подключения через систему точек подключения.

в ответах на все вопросы типа “уметь” содержат хуеву тучу текста, я сделал один и заебался