1. Знакомство @

$ - приглашение к вводу НЕ привилегированного пользователя

# - приглашение к вводу для привилегированного пользователя

Поиск используются следующие метасимволы:

(\*) ноль и более символов

(?) один символ

(~) начальный каталог текущего пользователя

~ *пользователь* - начальный каталог указанного *пользователя*

Авторизация

Log root

Pass 123456

sudo su – root # стать привилегированным пользователем

sudo su – ruakaim # стать собой

passwd # сменить пароль пользователя

1.7 Man страницы (Справочные страницы – это краткое описание отдельных команд , драйверов, форматов)

man *заголовок* # считывает онлайн страницы справочного руководства

man 1 command # эта запись говорит нам о том , что требуется найти команды в разделе один справочных страниц.

man passwd # выводит расширенное описание команды passwd

man –k *ключевое слово* или apropos *ключевое слово* # выводит список справочных страниц с строке пояснений к которым имеется ключевое слово. Круто

1.2 Поиск и установка ПО

ПО в linux поставляется в виде пакетов которые предварительно скомпилированы, хотя иногда ПО поставляется в виде исходного кода , которой потом компилируют в исполняемый код в соответствии с особенностями системы программы и включают в пакеты полученные бинарные файлы. Как правило проще инсталлировать бинарную версию пакета, предназначенную для данной системы, чем искать и компилировать исходный код.

**BSD** ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Berkeley Software Distribution*) — система распространения [программного обеспечения](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) в [исходных кодах](https://ru.wikipedia.org/wiki/%D0%98%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4), созданная для обмена опытом между учебными заведениями. Особенностью пакетов ПО BSD была специальная [лицензия BSD](https://ru.wikipedia.org/wiki/%D0%9B%D0%B8%D1%86%D0%B5%D0%BD%D0%B7%D0%B8%D1%8F_BSD), которую кратко можно охарактеризовать так: весь исходный код — собственность BSD, все правки — собственность их авторов. В данный момент термин BSD чаще всего употребляется как синоним BSD-UNIX — общего названия вариантов [UNIX](https://ru.wikipedia.org/wiki/UNIX), восходящих к дистрибутивам [университета Беркли](https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D0%BB%D0%B8%D1%84%D0%BE%D1%80%D0%BD%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%83%D0%BD%D0%B8%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%82%D0%B5%D1%82_%D0%B2_%D0%91%D0%B5%D1%80%D0%BA%D0%BB%D0%B8).

**Исхо́дный код** (также **исхо́дный текст**) — текст [компьютерной программы](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0) на каком-либо языке [программирования](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) или [языке разметки](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D1%80%D0%B0%D0%B7%D0%BC%D0%B5%D1%82%D0%BA%D0%B8), который может быть прочтён человеком. В обобщённом смысле — любые входные данные для [транслятора](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D1%81%D0%BB%D1%8F%D1%82%D0%BE%D1%80). Исходный код [транслируется](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D1%81%D0%BB%D1%8F%D1%82%D0%BE%D1%80) в *исполняемый код* целиком до запуска программы при помощи [компилятора](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%82%D0%BE%D1%80) или может исполняться сразу при помощи [интерпретатора](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BF%D1%80%D0%B5%D1%82%D0%B0%D1%82%D0%BE%D1%80).

ПО устанавливается в папку /usr/bin

Поиск можно осуществить следующими способами :

1. which screen # показывает полный путь к команде screen
2. whereis screen # ищет более широкий список каталогов и не зависит от пути
3. ls /usr/bin # ручной способ , зависит от пути
4. locate # Ищет файлы любого типа, но не ищет пакеты или команды для RedHat нужно ставить пакет mlocate в yum
5. rpm -q python # поиск пакета по имени (package manager – q query)
6. rpm –qf python #поиск пакета по имени сразу показывает путь до него
   1. Установка ПО (62стр.)

Yum – пакетный установщик для линукс для RHAT и centOS

Как пример

sudo yum install tcpdump # Установка tcpdump инструмента для захвата пакетов, позволяющего просматривать исходные пакеты отправляемы из системы в систему

sudo yum remove tcpdump #Удаление tcpdump

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1.4 Первым делом необходимо настроить доступ через putty по SSH (защищенное соединение) | | | | | | | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| для это проверяем есть демон SSH в системе | | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| service sshd status | |  | проверка демона SSH | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| system status firewalld | | | статус фаерволла | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| system stop firewalld | | | отсановка фаерволла | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| system disabled firewalld | | | выключение фервола из автозапуска | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| iptables -L | |  | проверка работоспособности firewall если нет ни каких правил то фаерволл отключен | | | | | | | | |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| еще желательно посмотреть | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| SELinux (англ. Security-Enhanced Linux — Linux с улучшенной безопасностью) — реализация системы принудительного контроля доступа, которая может работать параллельно с классической избирательной системой контроля доступа. | | | | | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| его можно выключить через текстовый редактор VI | | | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| vi etc/selinux/config | | | редактируем файл настроек SELinux | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| внутри файла в поле SELINUX меняем с enforcing(принудительно) на disabled | | | | | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| для применения нужен ребут | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| затем |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| далее немного обновим систему | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| yum update | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| ставим vim htop mc tree одной командой | | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| yum instal epel-release | | | набор интересных пакетов | | |  |  |  | EPEL (Extra Packages for Enterprise Linux) — это основанный сообществом открытый и бесплатный проект репозитория предоставляемый командой Fedora который содержит высококачественные пакеты дополнительного программного обеспечения для Linux дистрибутивов. | | | | | | | | | | | | |  |  |
| yum instal vim htop mc tree | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| reboot | перезагрузка системы | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| настраиваем сеть | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| yum instal net-tools | |  | набор сетевых иснтрументов ОБЯЗАТЕЛЬНО | | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| проверяем свой ip | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| ip addr | проверка ip | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| находим ip и создаем сессию через putty по этому ip | | | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| еще бонус надо поставить VirtualBox Guest Additions | | | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **VirtualBox Guest Additions** (дополнения гостевой операционной системы) – пакет расширений, который устанавливается в гостевую операционную систему и расширяет ее возможности по интеграции  и взаимодействию с хостовой (реальной) ОС. | | | | | | | | | | | | | | | |  |  |  |  |  |  |  |  |

* 1. Установка PostgreSQL

<https://www.dmosk.ru/miniinstruktions.php?mini=postgresql-install#install>

* 1. Создание ПО

1 можно создавать ПО из исходного кода, например, скачав с GitHub

yum install git # cтавим git

git clone \*\*\*\*\* # Скачивает репозиторий вместе со всей его историей изменений

git checkout tags/tcpdump-4.7.4 -Ь tcpdump-4.7.4 -пример

git checkout # создать ветку и сразу же перейти на неё, вы можете выполнить команду git checkout с ключом -b:

yum install gcc # затем ставится инсталлятор

в скаченном репозитрии выполняем

./configure

make

sudo make install

+ надежно есть история поставки кода

- муторно ставить

2 установка с помощью веб сценария

curl -о /tmp/saltboot –sL <https://bootstrap.saltstack.com>

sudo sh /tmp/saltboot

**Bootstrap** — **это** открытый и бесплатный HTML, CSS и JS фреймворк, который используется веб-разработчиками для быстрой вёрстки адаптивных дизайнов сайтов и веб-приложений.

Сценарии bootstrap исследует локальную сеть , затем инсталирует загружает и ставит ПО.

+Удобен для ПО , где процесс установки сложен

-Небезопасен не оставляет должной записи об инсталяции

-Пакеты на уровне ОС устарели и возможно не получите самую свежую версию ПО

- Внимательно следим за URL он должен быть https , а не http то есть использоваться защищенный протокол передачи гипертекста. Иначе вы легко уязвимы для перехвата хаккреами, потому что они знают что вы будете работать из под рута.

- Оболочка инсталляция происходит в режиме он лайн при это оболочка linux работает если произойдет сбой сети

То не понятно как поведет себя система.

3.Пакетная установка

2. Загрузка системы и системные демоны

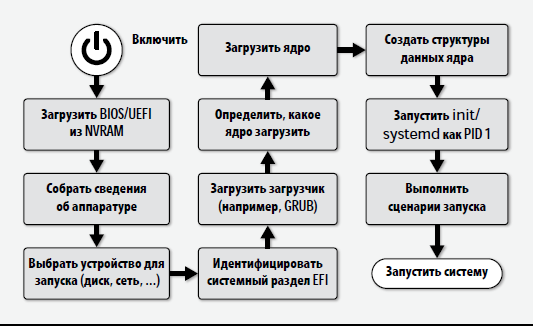
Bootstraping или самозагрузка системы:

Состоит из :

* Поиск, ввод в память и запуск кода начальной загрузки
* Поиск, ввод в память и запуск ядра ОС
* Активирование сценариев запуска и системных демонов
* Поддержание порядка управления переходами системы из одного состояния в другое

2.1 обзор процесса загрузки (стр 71)

Большинство дистрибутивов используют сеёчас system (демон системного менеджера)



* 1. Системные прошивки

BIOS (старая ) - базовое система ввода - вывода

UEFI новая – унифицированный интерфейс (способ взаимодействия ) расширения прошивки

efibootmgr # команда изменения ПОРЯДКА загрузки системы то есть сначала cd rom , потом HHD и тд

efibootmgr –v # посмотреть текущую конфигурацию порядка загрузки

sudo efibootmgr -o 0004 # изменить порядок загрузки на 0004

внимание systemd rm –rf # может быть достаточно чтобы стереть систему на уровне прошивки , ОПАСНО

* 1. Загрузчик GRUB универсальный загрузчик, который считывает конфигурацию системы из обычного текстового файла grub.cfg

1. основная задача загрузить ядро ОС , предоставляет пользователя выбор какое ядро или какую ОС запустить.

2.Маршалинг аргументов конфигурации ядра, передача специальных инструкций ядру во время начальной загрузки и/или используя загрузчик (например GRUB).

<https://itproffi.ru/konfigurirovanie-i-kompilyatsiya-yadra-linux/>

задание создать свою конфигурацию ядра linux c помощью утилиты grub2-mkconfig

3.Docker

Docker система контейнеризация. По сути это изолированная группа процессов , которые ограничены частной корневой файловой системой и пространством имен процессов.

Пространство имен – изолирует контейнерные процессы включая монтирование файловой системы, управление процессами и сетевое взаимодействие.

Контрольные группы – ограничивают использование системных ресурсов и определяют приоритет одних процессов над другими. Они препятствуют контейнерам вышедшим из под контроля использовать всю память и забирать всё процессорное время CPU

Функциональные возможности – позволяют выполнять определенные чувствительные операции с ядром. Например процесс может изменять права собственности на файл или менять системное время.

Режим защищенных вычислений – ограничивает доступ к системным вызовам.(обращение прикладной программы к ядру для выполнения какой либо операции).

docker – это запускаемая команда которая обрабатывает все задачи управления для системы Docker. Может работать с dockerd как удаленно пот tcp, так запускаться в одной системе

Dockerd – Это резидетный процесс демона который реализует операции с контейнерами и образами. Создает виртуальную сеть и поддерживает каталог данных в котором хранятся контенеры и образы (по умолчанию /var/lib/docker).

Docker run # создать контенер

Подкоманда

docker info Отображает сводную информацию о демоне

docker ps Отображает запущенные контейнеры

docker version Отображает обширную информацию о версии сервера и клиента

docker rm Удаляет контейнер

docker rm $(docker ps -a)

docker rmi Удаляет образ

docker images Отображает локальные образы

docker inspect Отображает конфигурацию контейнера (данные в формате JSON )

docker logs Отображает стандартный вывод из контейнера

docker ехес Выполняет команду в существующем контейнере

docker run Запускает новый контейнер

docker pull/push Загружает образы из удаленного реестра или загружает образы в удаленный реестр

docker s tart/ s top Запускает или останавливает существующий контейнер

docker top Отображает состояние контейнерного процесса

3.1 установка образа docker из artifactory(для банковской сети способ!) на примере postgresql

1. в артифактори находим образ праваливаемся и находим

Repository Path: ext-rbru-devimage-container-docker/postgres/12.4/manifest.json

1. Составляем команду установки образа:

sudo docker pull artifactory.raiffeisen.ru/ext-rbru-devimage-container-docker/postgres:12.4

artifactory.raiffeisen.ru/ext-rbru-devimage-container-docker/postgres путь

:12.4 версия ПО

вот еще статья

<https://confluence.raiffeisen.ru/pages/viewpage.action?pageId=218179246>

делаем проверку docker images видим наш образ

1. создание Контейнера на основе образа с параметрами

docker run --rm -e POSTGRES\_USER=root -e POSTGRES\_PASSWORD=123 -d --name testpost artifactory.raiffeisen.ru/ext-rbru-devimage-container-docker/postgres

На основе образа мы создаем свой экземпляр в контейнере

--rm автоматически удаляет контейнер при выходе из него. Если этого не сделать то командой docker мы увидим что контейнер находится в работе

-e POSTGRES\_USER=root -e POSTGRES\_PASSWORD=123 создаем пользователя и пароль при создании контейнера

-d Запустить контейнер в фоновом режиме и распечатать идентификатор контейнера

--name testpost задаем имя контейнера (если этого не сделать имя будет автоматическое)

1. Запустить сеанс баш или sh (это оболочки для исполнения команд )в работающем контейнере

docker exec -it testpost bash

docker exec -it testpost sh

1. Запуск postgresql

psql

1. Остановка контейнера (используя контейнер id или names эти параметры знаем из docker ps)

docker stop b81547375de7

docker stop happy\_gould

3.2 установка Docker из внешених репозиториев

1.1 sudo yum check-update #обновляем индекс пакетов

1.2 curl -fsSL https://get.docker.com/ | sh # установка

1.3 sudo systemctl start docker # запуск демона докера

1.4 docker search postgre #поиск образа

1.5 docker run --rm -it -e POSTGRES\_USER=root -e POSTGRES\_PASSWORD=123 -d --name postgreSQl postgres

Все аргументы команды описаны выше кроме

-it

-t tty Данная система предназначена для использования одного терминала несколькими процессами,

1.6 docker exec -it postgreSQl bash

1.7 su – postgres

1.8 psql

3.3 создание docker volume (том)

1. docker volume create my-vol #создает docker том локально

2. docker volume ls # смотрим список томов

3. docker volume inspect my-vol # параметры тома

4. docker run --rm -it -e POSTGRES\_USER=root -e POSTGRES\_PASSWORD=123 -d --name postgreSQl --mount type=volume,source=my-vol,destination=/root/test\_volumes postgres

5. docker exec -it postgreSQl bash

6. psql

7. bash

4.1 основные возможности командного интерпретатора

Bash , sh, tmux и др.

cat /etc/shells #посмотреть список установленных интерпретаторов

echo $SHELL # в переменную SHELL записан интерпретатор по умолчанию

cat /etc/sysconfig/network-scripts/ifcfg-lo # абсолютный путь

cat ifcfg-lo # относительный путь (то есть выполненный относительно директории в которой находимся

Type # узнать тип команды внутренняя, внешняя псевдоним или сценарий и т.д.

Ls # список файлов в директории:

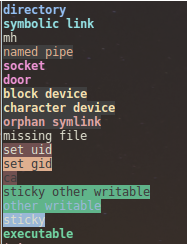
аргументы

-a все даже скрытые

-с –lt показывает время создания, сортирует по времени создания

-с –l показывает время создания, сортирует по алфавиту

* **Некорированный (белый)**: файл *или*текст без имени файла (например, разрешения на выходе ls -l )
* **Жирный синий**: каталог
* **Жирный голубой**: символическая ссылка
* **Жирный зеленый**: исполняемый файл
* **Полужирный красный**: архивный файл
* **Жирный пурпурный**: файл изображения, видео, графический файл и т. д. *или*дверь *или*.
* **Голубой**: аудиофайл
* **Желтый с черным фоном**: труба (AKA FIFO)
* **Полужирный желтый с черным фоном**: устройство-блок *или*.
* **Полужирный красный с черным фоном**: сиротская символьная ссылка *или*отсутствующий файл
* **Некорированный с красным фоном**: файл с идентификатором пользователя
* **Черный с желтым фоном**: файл с идентификатором группы
* **Черный с красным фоном**: файл с возможностью
* **Белый с синим фоном**: липкий каталог
* **Синий с зеленым фоном**: каталог с возможностью записи
* **Черный с зеленым фоном**: липкий и другой доступный каталог



mkdir #создать директорию

~ # всегда ссылается на домашний каталог

5. VIM

I #войти в режим редактирования

Esc # выйти

:q Выйти из редактора (можно использовать в случае, если текст

не менялся)

:q! Выйти из редактора с потерей сделанных изменений

:w Сохранить файл на диск

:w имя\_файла Сохранить файл с указанным именем

:wq Сохранить файл и выйти

:x Сохранить файл и выйти (аналог wq)

:set number Включить нумерование строк (удобно использовать при отладке

сценариев, когда известен номер строки, вызывающей ошибку)

:set nonumber Отключить нумерование строк. У редактора есть целый набор

подобных переключателей, позволяющих влиять на его работу

**Команда Описание**

h,j,k,l Перемещение курсора влево, вниз,

вверх и вправо

dd Удалить строку

d$ Удалить символы до конца строки

gg Переместить курсор на первую строку

файла

G Переместить курсор на последнюю

строку

yy Скопировать строку в буфер

p Вставить строки из буфера ниже

курсора

P Вставить строки из буфера выше

курсора

u «Undo» отмена последней команды

. Повтор последней команды

/ Поиск по файлу вперёд

? Поиск по файлу назад

n Перемещение по найденному вперёд

N Перемещение по найден

1. Потоки и конвееры

Linux/UNIX системах разработчиками реализовано три стандартных потока:

стандартный ввод (stdin), стандартный вывод (stdout) и стандартный вывод

ошибок (stderr).

< # перенаправление ввода

> # перенаправление вывода

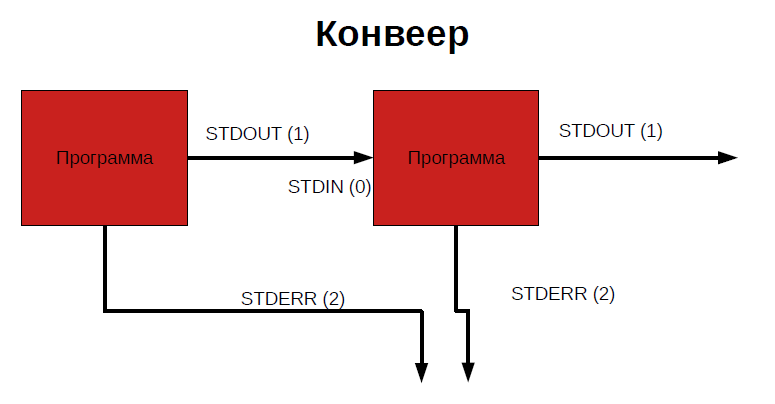
>> # перенаправление вывода без затирания старых данных если вывод идет в файл

2> # вывод ошибок

> /dev/null # перенаправление в никуда

Пример :

ls -la /tmp > ls.txt # перенаправит вывод исполнения команды ls в текстовый файлы ls.txt который сама и создасть



6.2 Стандартный вывод можно перенаправлять не только в файл, но и на ввод других

программ. Этот механизм называется конвейеризация и осуществляется он с

помощью символа «|»

rpm –qa | wc –l # считает кол-во установленных пакетов rpm

rpm #packet manager

-qa запросить все пакеты

wc #word count посчитать кол-во слов строк и тд

-l посчитать строки lines

cat /etc/passwd | cut -d: -f1 | sort | tr [a-z] [A-Z] # выводит список пользователей в системе

cat /etc/passwd показать файл со списком но из него еще нужно выбрать пользователей

cut # вырезать поле f1

-d # делиметр разделитель в данном пример :

-f1 оставляет поле 1 zabbix ( zabbix:x:998:211:Zabbix Monitoring System:/var/lib/zabbix:/sbin/nologin )

sort # сортировка

tr[a-z][A-Z] # translate преобразование к верхнему регистру

tee # команда ветвитель одновременно пишет в файл и выводит на экран

ls -c -lt /bin/ | tee testAllBinList.txt

* 1. Grep

grep # поиск по файлу

-n Выводить номера строк, в которых присутствует искомое значение

-i Отключить чувствительность к регистру символов

-A Х Выводить не только строку с найденным значением но и Х строк,

расположенных после неё (After)

-B Х Выводить не только строку с найденным значением но и Х строк,

расположенных до неё (Before)

-v Инвертировать результат

-c Вместо обнаруженных строк вывести их количество

-m Х Прекратить поиск после Х совпадений

ps -ef | tee ps.txt | grep 'postfix' # пример выводит список процессов | выводит данные в файл | фильтрует по тегу postfix

ps -ef | grep postfix # ищем процесс с название постфикс

-e отображает все активные процессы

-ef активные процессы в формате full

регулярные выражения

^ Начало строки grep '^root' /etc/passwd

$ Конец строки grep 'bash$' /etc/passwd

. Любой одиночный символ, за исключением перевода строки grep '^r...' /etc/passwd

\* Любое количество повторов предыдущего символа, включая ноль \* соответствует комбинации любых символов grep '^hel\*o' /usr/share/dict/words

[abc-f] Подмножество символов. Означает, что символ может принимать одно из указанных в скобках значений: a,b,c,d,e,f grep d[io]g /usr/share/dict/words

[^xyz] Инвертированное подмножество. Означает, что символ может принимать grep ^r[^o].\* /etc/passwd любое значение, кроме перечисленного в скобках

grep ‘^[^#]’ /etc/selinux/config – инвертирует вывод то есть выводит все значения где нет символа #

* 1. PS список процессов

ps -aux #выводит список всех процессов в системе

ps –lax # подробный формат вывода

ps -aux --sort –pcpu # сортировка по нагрузке на cpu

ps -aux --sort –pmem # сортировка по нагруженности памяти

ps -aux --sort=-time # сортировка по столбце time то есть время исполнения процесса

a # all

u # фильтровать по имени

x # увидеть процессы отсоединенные от управляющего терминала

l # длинный формат вывода данных

-o # пользовательские форматы вывода то есть кастомные в данном случае выводим pid,comm

при выдаче в столбце command , бывают вот такие записи [kworker/0:2H] – это потомки ядра

* 1. найти \ узнать номер процесса

pidof /usr/sbin/sshd # команда помогает определить PID процесса

pgrep sshd # команда помогает определить PID процессаУДОБНО

* 1. TOP утилита которая показывает нагруженность в реальном времени УДОБНО

Внутри top можно нажать 1 чтобы переключиться между ядрами ЦПУ

topas # утилита на bfas1-10 для просмотра нагруженности системы

* 1. Поиск команда find

find / -name fstab # Найти во всей файловой системе файлы c именем fstab

find /usr -iname y\*.conf # Найти в директории /usr и всех лежащих ниже поддиректориях файлы, с именем,

начинающимся с y и расширением .conf без учёта регистра.

Параметры команды find:

-type # Тип искомого f - файл, d - директория, l - ссылка, s -сокет, p - канал, b - файл блочного устройства, с – файл символьного устройства find / -type f –name ps.txt

-user Владелец файла, заданный именем find / -user daemon

-group # Группа, связанная с файлом, заданная именем find /var -group mail

-o “или” (or) # Позволяет задать несколько критериев find / -name "\*.bak" -o -name "\*.tmp"

-not “не” (not) find /var -mmin -60 -not name messages

-uid Владелец файла, заданный

идентификаторомfind / -uid 1000 -not -path "/home/\*"

-gid Группа, связанная с файлом, заданная идентификатором find /home -gid 1000

-nouser Файлы, имеющие владельцем неопределенного пользователя (возможно он был удалён)

find / -nouser

-size Файлы, имеющие определенный размер. Размер можно указывать в байтах (с), килобайтах (k),

мегабайтах (М), гигабайтах (G) и 512байтных блоках (b)

-maxdepth Указать глубину поиска. -maxdepth 1 означает «искать

только в указанной директории»

-mmin n Файлы, изменённые n минут назад. - n означает найти файлы, изменённые в течении n минут.

-mtime n Файлы, изменение которых find /var -mtime -1 последний раз было 24\*n часов назад

-empty Искать пустые файлы (в случае директории подразумевается, что в ней ничего нет)

Еще несколько примеров использования find:

find /var/ -nouser -delete

Найти и удалить все файлы в директории /var и нижележащих, имеющие

неопределённого владельца

find /home -type f -ls find /etc -type f -mtime -7 ! -mtime -3

Найти все файлы в директории /etc, которые были модифицированы за последние

7 суток, но не позднее, чем трое суток назад.

find /home -type f -exec cp {} /data \;

Найти все файлы в директории /home и скопировать их в директорию /data

* 1. sort , uniq

ps –e –o pid,comm,user | sort -n

- e вывести все параметры

-o пользовательские форматы вывода то есть кастомные в данном случае выводим pid, comm ,user В таком виде она будет показывать владельца, идентификатор и имя команды,

породившей процесс.

–n отсортировать по идентификатору номера

ps --no-headers -e -o user | sort | uniq

7.Права доступа к папкам и директориям

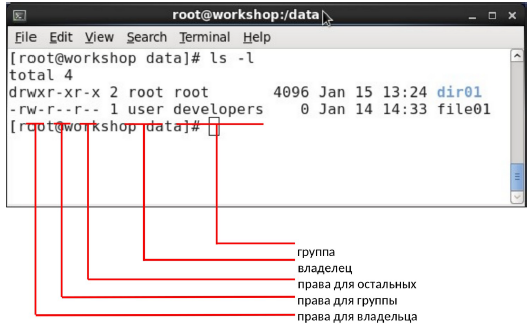
R - Чтение запись

W - Запись

X - Испольнение

Ls –l # Посмотреть права доступа можно при помощи команды

-l # лонг формат



chown # смена владельца файла

sudo chown ruakaim ps.txt # смена владельца файла

sudo chown ruakaim:users ps.txt # смена владельца и пользователя группы

sudo chown :users ps.txt # смена группы

chgrp # смена группы

id –Gn ruakaim список групп пользователя

-G выводит идентификаторы групп

-n имена групп

chmod # изменение прав доступа

Владелец файла обозначается буквой "u" (user), группа - буквой "g"

(group) и все остальные - буквой "o" (other). Если нужно применить какое-то

действие сразу ко всем трём категориям, то можно использовать букву "a" (all).

+ добавить прав к уже имеющимся

- отнять права от уже имеющихся

= присвоить права

chmod u+x,g-w script02.sh

И команда chown, и команда chmod поддерживают опцию "-R" которая позволяет

менять права доступа рекурсивно. Т.е. команда "chmod -R g+w /somedir"

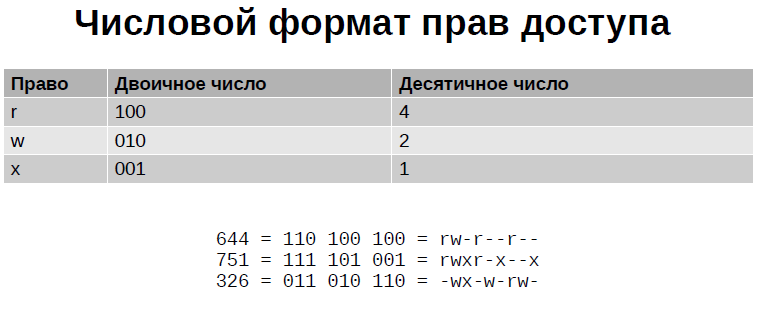
Установит право на чтение для группы не только на саму директорию но и на все

вложенные в неё файлы и поддиректории.

ТОЛЬКО для атрибута, дающего право на исполнение предусмотрен вариант,

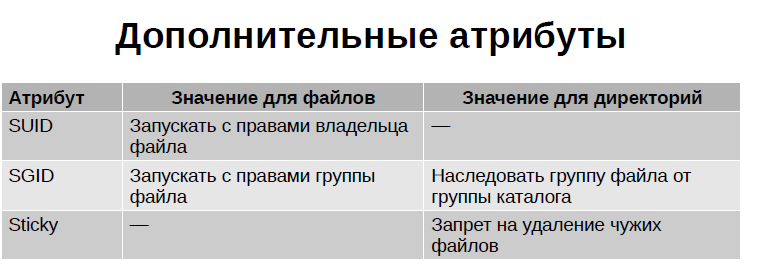
позволяющий указывать его как в нижнем (x), так и в верхнем (Х) регистре. !

7.1 Числовой формат права доступа



Пример

chmod 754 script04.sh # на исполняемый файл установлены следующие права rwx - r-x - r - -



если на каталог установлен атрибут sticky bit (-t), то удалить файл в этом каталоге может только его владелец.

7.3 Права по умолчанию

По умолчанию задаются права для файла 666 и директории 777

Изменить права по умолчанию можно задав параметр маски

umask # посмотреть какой сейчас параметр маски

umask 0027 # параметр маски присвоить 27 это число переводим в двоичную форму 11011 и отнимаем от параметра по умолчанию 666

110110110

000011011

Получаем 110101101 = 655 вот такое значение будет задано по умолчанию

Для root и пользователей маски по умолчанию РАЗНЫЕ!

8.1 Создание групп и пользователей

groupadd # добавить группу

useradd # добавить пользователя

sudo useradd -G developers chip # создать пользователя и добавить его группу

9.Смена пользователя

9.1 su # работа от лица другого пользователя

su # это запуск оболочки с привилегиями администратора

su –root # ДЕФИС говорит о там что мы переключаемся к учетной записи пользователя ВМЕСТЕ С ЕГО ОКРУЖЕНИЕМ , переменными и тд

su root # БЕЗ ДЕФИСА меняется только UID , но меняется окружение

ВАЖНО после завершения действия под пользователем выйти из под него !!! exit ctrl +D

МИНУСЫ su:

1. Нужно знать пароль пользователя

2.Запуск оболочки с повышенными привилегиями потенциально опасен

9.2 sudo

Sudo # позволяет запускать команды пользователя без пароля для чего это нужно

По умолчанию новый пользователь создается без права исполнения команд sudo, это сделано в целях безопасности. Чтобы пользователь имел возможность выполнять все команды необходимо добавить его в группу sudo:

usermod -a -G sudo <имя пользователя> #добавляет пользователя в группу sudo

Sudo cat /etc/passwd # выполнит команду чтения без пароля

Чтобы запретить выполнение определенных команд или утилит для всех кроме пользователя root, достаточно выполнить следующую команду:

Sudo chmod o-x $(which ls) #после выполнения этой команды пользователи кроме рута не смогут выполнять команды ls

Файл /etc/sudoers # определяет какие пользователи могут выполнять команды sudo

Редактируется при помощи visudoers

%webmasters ALL=/bin/systemctl restart httpd.service

кто где=(от имени кого) что

%webmasters # “Кто” имя пользователя или группы. Если подразумевается группа, то перед её

именем ставится знак "%"

ALL= #“где” имя хоста или IP-адрес узла, на котором допускается запускать команду.

Если этот параметр не принципиален, то можно использовать зарезервированное слово "ALL".

“от имени кого” - от имени кого будет выполнена команда. Если параметр не

указывается, то подразумевается пользователь root. (ЗДЕСЬ В ПРИМЕРЕ ПУСТО ЗНАЧИТ ОТ ROOT)

restart httpd.service # "что" - список команд, которые допустимо выполнять пользователю или группе

Таким образом, если мы хотим группе webmasters дать возможность перезапускать вебсервер

7.6 демон init

Первый демон который стартует после загрузки ядра это init

Основная задача запуск служб и демонов

• Установка имени компьютера.

• Установка часового пояса.

• Проверка дисков с помощью команды fsck.

• Монтирование файловых систем.

• Удаление старых файлов из каталога / tmp

• Настройка сетевых интерфейсов.

• Настройка фильтров пакетов.

• Запуск других демонов и сетевых служб.

Init старая версия демона , не поддерживает параллельную работу то есть более поздние действия не могут выполнятся пока не закончились предыдущие

Systemd поддержка параллельной работы процессов, управление сетевыми подключениями , авторизация и записи журнала ядра все это он подддреживает

Недостаток в том, что он контролирует кучу вещей а это противоречит философии unix, где сказано что программы должны быть простыми

Systemd управляет модулями (unit), поведение которых настраивается модульным файлом.

/usr/lib/system/system # основное место куда пакеты сохраняют модульные файлы (юниты) после инсталяции.

/lib/systemd/system. # Либо здесь для centOS лежат юниты

7.7 systemctl это универсальная команда изучения состояния системного менеджера system

Systemctl # выводит общий список всех загруженных активные и неактивные, сокеты диски устройства юниты list-units

systemctl list-units --type=service # только сервисы то есть службы

systemctl list-unit-files --type=service # список модульных файлов

systemctl list-units --type = target # Посмотреть список доступных целей

systemctl isolate [target] # Изменит текущую цель targe(берем из списка доступных целей) то есть режим работы системы

1. Управление доступом

8.1 root

Root имеет UID GID равные 0 что позволяет ему выполнять привилегированные операции , не рекомендуется назначать эти параметры при создании других пользователей , во избежание нарушения безопасности системы.

Изначально UID GID равные 0 принадлежат процессу login и как только пользователь root входит в систему UID GID равные 0 передаются вам.

root нужно аккуратно нужно пользоваться очень аккуратно:

1. Уз root не содержат информации какие записи выполнялись с правами root.
2. Сценарий входа в систему не содержит информации кто же все таки входил в систему то есть это мог быть кто угодно если у нескольких пользователей есть доступ уз рута.

Создание групп и пользователей

groupadd # добавить группу

useradd # добавить пользователя

sudo useradd -G developers chip # создать пользователя и добавить его группу

* 1. SU

лучше получать доступ к учетной записи root с помощью команды su

При вызове без аргументов она выдает приглашение на ввод пароля пользователя , затем запускает оболочку с привилегированными правами до самого выхода из под рута.

Su не фиксирует действия пользователя под рутом но зато пишет в журнал кто и когда время входа и выхода вошел в систему.

Зная чьи - либо пароли можно заходить из-под себя под ними попытаться помочь

* 1. SU –

Это вход с РЕГИСТРАЦИЕЙЫ обычно в этом режиме меняются количество и имена файлов запуска, считываемых интерпретатором. Например, в режиме регистрации оболочка bash считывает файл ~!/ . bash\_profile, а вне этого режима - файл " / . bashrc. При диагностике проблем конкретного пользователя режим

регистрации позволяет максимально точно воспроизвести среду, в которой он работал.

В одних системах нужно стать su root , а потом например su rza и после этих действий переключене пользователя идет без ввода пароля. В других же достаточно просто зайти под root и su будет без пароля.

ВАЖНО сделать правилом использовать от абсолютный путь /bin/su для команды, чтобы избежать вредоносных команды su сделанных злоумыщенниками для сбора чужих паролей

Иногда для использования su нужно быть членом группы wheel

su # работа от лица другого пользователя

su # это запуск оболочки с привилегиями администратора

su –root # ДЕФИС говорит о там что мы переключаемся к учетной записи пользователя ВМЕСТЕ С ЕГО ОКРУЖЕНИЕМ , переменными и тд

su root # БЕЗ ДЕФИСА меняется только UID , но меняется окружение

ВАЖНО после завершения действия под пользователем выйти из под него !!! exit ctrl +D

МИНУСЫ su:

1. Нужно знать пароль пользователя

2.Запуск оболочки с повышенными привилегиями потенциально опасен

* 1. SUDO это ограниченый вариант команды su

Именно sudo рекомендуется использовать для доступа к учетной записи root:

1.sudo проверяет файл /etc/sudoers в котором перечислены люди которые, имеющие право использовать sudo и команды которые разрешено им использовать на данном хосте. Если команда разрешена к исполнению sudo запрашивает у пользователя его пароль для исполнения.

2.sudo может вести журналирование , для того что бы лог писался его нужно куда то определить

sudo visudo

Defaults logfile=/var/log/sudolog прописываем в переменную путь для лога

Теперь проверяем должно работать

3.sudo лучше не настраивать на ввод без пароля

Ansible ALL=(ALL) NOPASSWD: ALL # так делать крайне не рекомендуется!

Если лень вводить пароли можно заменить ручную аутентификацию ssh-agent и перенаправление SSH – ключей. Установите пакте pam\_ssh\_agent\_auth

Группы etc/group описание

* **daemon** - от имени этой группы и пользователя daemon запускаютcя сервисы, которым необходима возможность записи файлов на диск.
* **sys** - группа открывает доступ к исходникам ядра и файлам include сохраненным в системе
* **sync** - позволяет выполнять команду /bin/sync
* **games** - разрешает играм записывать свои файлы настроек и историю в определенную папку
* **man** - позволяет добавлять страницы в директорию /var/cache/man
* **lp** - позволяет использовать устройства параллельных портов
* **mail** - позволяет записывать данные в почтовые ящики /var/mail/
* **proxy** - используется прокси серверами, нет доступа записи файлов на диск
* **www-data** - с этой группой запускается веб-сервер, она дает доступ на запись /var/www, где находятся файлы веб-документов
* **list** - позволяет просматривать сообщения в /var/mail
* **nogroup** - используется для процессов, которые не могут создавать файлов на жестком диске, а только читать, обычно применяется вместе с пользователем nobody.
* **adm** - позволяет читать логи из директории /var/log
* **tty** - все устройства /dev/vca разрешают доступ на чтение и запись пользователям из этой группы
* **disk** - открывает доступ к жестким дискам /dev/sd\* /dev/hd\*, можно сказать, что это аналог рут доступа.
* **dialout** - полный доступ к серийному порту
* **cdrom** - доступ к CD-ROM
* **wheel** - позволяет запускать утилиту sudo для повышения привилегий
* **audio** - управление аудиодрайвером
* **src** - полный доступ к исходникам в каталоге /usr/src/
* **shadow** - разрешает чтение файла /etc/shadow
* **utmp** - разрешает запись в файлы /var/log/utmp /var/log/wtmp
* **video** - позволяет работать с видеодрайвером
* **plugdev** - позволяет монтировать внешние устройства USB, CD и т д
* **staff** - разрешает запись в папку /usr/local

 groups rza # Посмотреть группы пользователя

Разрешить пользователю выполнять команду sudo

Если при создании пользователя не задать ему группу , то по умолчанию он по падает в одноименную группу например пользователь rza попадает в группу rza и удалить ее не получиться если не поменять его первичную группу

usermod -aG wheel,mydevops rza # данной командой мы добавил пользователя rza в группы wheel (для получения возможности выполнять судо ) , в группу mydevops( это мной созданная группа)

a – append добавить

G – group

Но этого не хватило пришлось еще дописать пользователя /etc/sudoers

Visudo

В конце файла

rza ALL=(ALL) ALL

статья [Как дать пользователю sudo права в CentOS 8 (merionet.ru)](https://wiki.merionet.ru/servernye-resheniya/37/kak-dat-polzovatelyu-sudo-prava-v-centos-8/)

Sudo # позволяет запускать команды пользователя без пароля для чего это нужно

По умолчанию новый пользователь создается без права исполнения команд sudo, это сделано в целях безопасности. Чтобы пользователь имел возможность выполнять все команды необходимо добавить его в группу sudo:

usermod -a -G sudo <имя пользователя> #добавляет пользователя в группу sudo

Sudo cat /etc/passwd # выполнит команду чтения без пароля

Чтобы запретить выполнение определенных команд или утилит для всех кроме пользователя root, достаточно выполнить следующую команду:

Sudo chmod o-x $(which ls) #после выполнения этой команды пользователи кроме рута не смогут выполнять команды ls

Файл /etc/sudoers # определяет какие пользователи могут выполнять команды sudo

Редактируется при помощи visudoers

%webmasters ALL=/bin/systemctl restart httpd.service

кто где=(от имени кого) что

%webmasters # “Кто” имя пользователя или группы. Если подразумевается группа, то перед её

именем ставится знак "%"

ALL= #“где” имя хоста или IP-адрес узла, на котором допускается запускать команду.

Если этот параметр не принципиален, то можно использовать зарезервированное слово "ALL".

“от имени кого” - от имени кого будет выполнена команда. Если параметр не

указывается, то подразумевается пользователь root. (ЗДЕСЬ В ПРИМЕРЕ ПУСТО ЗНАЧИТ ОТ ROOT)

restart httpd.service # "что" - список команд, которые допустимо выполнять пользователю или группе

Таким образом, если мы хотим группе webmasters дать возможность перезапускать вебсервер

Windos vs linux

1.использование Windows или Linux зависит от требований и бюджета. Linux бесплатен кроме redhat. По сути линукс хорош для малых предприятий где экономят бюджет на лицензиях

2.Зависит от софта есть софта который есть под линукс и есть софт под windows.

3.Администрирование и политики Linux более тонкие, чем администрирование и политики Windows. На Linux даже для пользователя можно дать через /etc/sudoers возможность запускать драйвера, приложения.

4. На windows более дешевое администрирование, то есть работа сис админа стоит дешевле, то есть для крупных предприятий есть выгода .  Для малого бизнеса будет выгодней использовать Linux, так как потратить 140 000 только на операционную систему могут не все, хотя, эти деньги относительно быстрей окупятся за счет более дешевого системного администратора, софта и периферии. Для крупного предприятия, покупающего сотни, а то и тысячи лицензий, гораздо выгодней использовать Windows, так как при таких закупках стоимость лицензии может снизиться до 700 рублей за лицензию, так же, как и дешевле обойдется администрирование.

5.  линуксовые серверы используют для размещения веб-серверов на nginx или Apache (в противовес Microsoft IIS), для работы с PostgreSQL и MySQL или популярными сейчас скриптовыми языками разработки.

6. С серверами Linux вы можете сделать много изменений в ядре. Тем не менее, сервер Windows, не так гибок при выполнении задач. Вы не можете делать любые модификации ядра, а можете лишь приблизиться к выполнению ваших индивидуальных требований, выбрав один из доступных вариантов операционной системы Windows.

7. с сервером Windows, вы точно знаете, к кому обращаться – [Microsoft](https://www.allprogram.ru/basket/URBGRgkdHENGFkwDDghJRlxXQFMJGkpGFilbUxEKFglfEB9mQV1ZUVIVT1BSVQkaW0RfXg==/). С Linux и другими открытыми платформами, вы можете не знать правильные места, где искать поддержку в случае любых неполадок.

8.windos более требователен к ресурсам чем линукс

9. Основным минусом использования ОС Windows является то, что это Windows. По сравнению с Linux Windows предлагает очень мало возможностей для настройки, и это ограничение может быть недостатком с точки зрения DevOps. Когда дело доходит до основных задач, таких как работа с командной строкой, вам понадобятся дополнительные инструменты, которые предлагает Microsoft. А также вы не сможете сделать систему Windows максимально защищенной.

10. Зачастую инженеры DevOps занимаются поддержкой конвейеров доставки программного обеспечения, которые развертываются в серверных средах Linux. Возможность работать из собственной среды Linux может упростить выполнение таких задач, как тестирование и размещение этих приложений на локальном устройстве, когда необходимо. Вы можете, например, развернуть контейнер Linux на вашем локальном компьютере, чего нельзя сделать на ОС Windows или macOS.

Источник

[Linux vs windows servers - какой дистрибутив выбрать — компания FreeHost](https://freehost.com.ua/faq/articles/windows-server-ili-linux-distributivi-vibiraem-servernuju-os-servernie-operatsionnie-sistemi/#1)

[Какая операционная система лучше всего подойдет для работы DevOps? : Rebrain | Блог (rebrainme.com)](https://rebrainme.com/blog/devops/kakaya-operaczionnaya-sistema-luchshe-vsego-podojdet-dlya-raboty-devops/)

Postgre база данных

1.Большие размеры таблиц

2.открытый исходный код

3.Массивы

4.Коррдинаты

5.Json

[Чем PostgreSQL лучше других SQL баз данных с открытым исходным кодом. Часть 1 / Хабр (habr.com)](https://habr.com/ru/post/282764/)

Задачи для практики

1.Настроить Kerberos между серверами