



---

# LINUX. УРОВЕНЬ 1. ОСНОВЫ АДМИНИСТРИРОВАНИЯ И БЕЗОПАСНОСТИ

---

Конспект и лабораторные работы по курсу

Автор: Соколов М.Ю.



## **Программа курса**

### **Глава 1. Введение**

Базовые понятия Linux

Дистрибутивы Linux

### **Глава 2. Установка и подключение к системе.**

### **Глава 3. Структура ФС в Linux.**

Устройство ФС в Linux

Работа с файлами

Введение в регулярные выражения

Основы работы в текстовых редакторах (vim и sed)

### **Глава 4. Настройка ОС.**

Настройка сетевых интерфейсов

Настройка локализации

Управление модулями ядра

### **Глава 5. Управление подсистемой хранения данных**

Подключение дисков и создание разделов

Управление файловыми системами

Менеджер логических томов LVM. Создание и управление логическими томами

### **Глава 6. Управление сервисами**

Системы инициализации SystemV, systemd

Процессы. Мониторинг процессов. Межпроцессное взаимодействие. Сигналы

Переменные окружения

### **Глава 7. Введение в использование сценариев оболочки**

### **Глава 8. Система безопасности Linux**

Управление пользователями и группами

Аутентификация и авторизация.

Использование криптографических ключей для аутентификации

Модули PAM

Управлением доступом и привилегиями

Управление привилегиями с помощью настройки sudo

Управление режимом доступа

Использование Posix ACL

## **Глава 9. Управление ПО**

Системы управления пакетами

Установка пакетов

Репозитории ПО

Сборка и установка пакетов из исходников

Графический пользовательский интерфейс в Linux

## **Глава 10. Резервное копирование и восстановление**

Утилиты резервного копирования

Управление периодическими заданиями

## **Глава 11. Анализ производительности и оптимизация системы**

## **Глава 12. Использование веб консоли Cockerpit для удаленного администрирования**

## ГЛАВА 1. ВВЕДЕНИЕ

### Базовые понятия Linux

# Linux

1991 год. Linus Torvalds опубликовал исходный код ядра (версия 0.01)



1996 год. Выбран пингвин Тух (расшифровывается как Torvalds UniX) в качестве эмблемы

Исходники ядра Linux можно взять на:

<http://kernel.org/> и

<https://github.com/torvalds/linux>



УЦ

## Обзор дистрибутивов Linux

### RPM-based



### DPKG-based



## ГЛАВА 2. УСТАНОВКА И ПОДКЛЮЧЕНИЕ К СИСТЕМЕ.

### 2.1 Подключение и вход в систему.

Стандартным способом отправки команд является использование удаленной оболочки. Для этого используется протокол ssh

Ssh означает «Secure Shell» (безопасная оболочка) и является протоколом для предоставления безопасного подключения

Ssh обеспечивает безопасное подключение по протоколу secure shell (ssh), являясь его клиентом

В отличие от telnet при работе ssh соединение зашифровывается сразу после подключения клиента к серверу. Алгоритм шифрования RSA не требует секретности ключа для шифрования. Секретным должен быть ключ для расшифровки, а он всегда остается на стороне сервера и перехватить его невозможно.

Терминал, консоль и псевдотерминал.

Исторически, терминал – конечное устройство (относительно порта) для взаимодействия пользователя с системой.

Терминал (tty), либо аппаратно-реализованное устройство, подключенное к физическому порту (интерфейс RS-232), либо эмулируемое ядром – обеспечивающее ввод с клавиатуры.

Псевдотерминал (pts) – устройство, эмулируемое программой, например, ssh или xterm.

Консоль – устройство вывода (монитор), подключенное непосредственно к управляемому хосту.

### **Виртуальные терминалы (консоли)**

Поддерживаются 63 (1-63) виртуальных терминалов

По умолчанию предоставляются семь виртуальных консолей, переключаться между которыми можно комбинацией Ctrl + Alt + F1-7.

Виртуальные терминалы выше 7-го также существуют, но для них по умолчанию не предусмотрено ни текстового терминала, ни графического.

### **Командная оболочка (shell)**

Командная оболочка (командный интерпретатор) - это программа, задача которой состоит в том, чтобы передавать ваши команды операционной системе и прикладным программам

В командной оболочке можно писать и запускать небольшие программы для выполнения ряда последовательных операций с файлами и содержащимися в них данными — сценарии (скрипты)

Стандартом де-факто в Linux является bash (Bourne Again Shell)

Пользователи Linux могут свободно изменять свои оболочки

echo \$SHELL – отобразит используемую оболочку

Приглашение оболочки

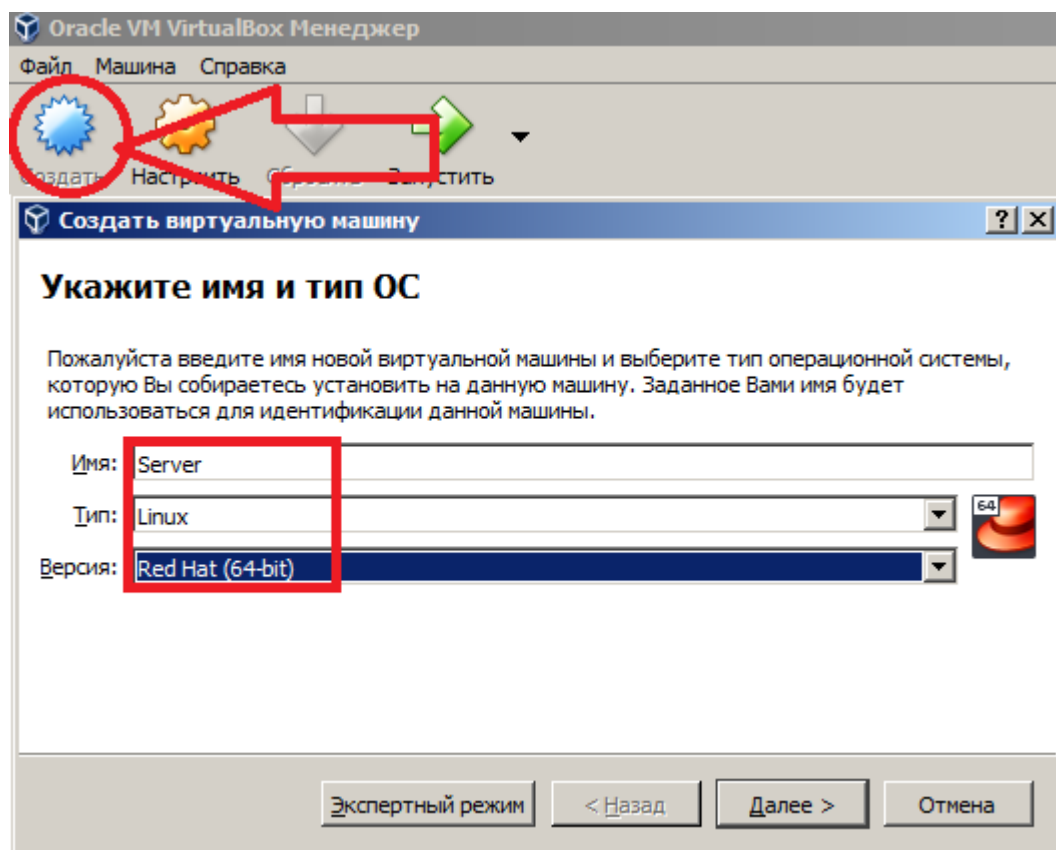
Вы должны увидеть символ \$, если вы вошли в систему как обычный пользователь. Это визуальный сигнал о том, что вы входите в систему как обычный пользователь.

Административный пользователь в linux называется root и символ # в приглашении указывает на то, что Вы вошли в систему как административный пользователь.

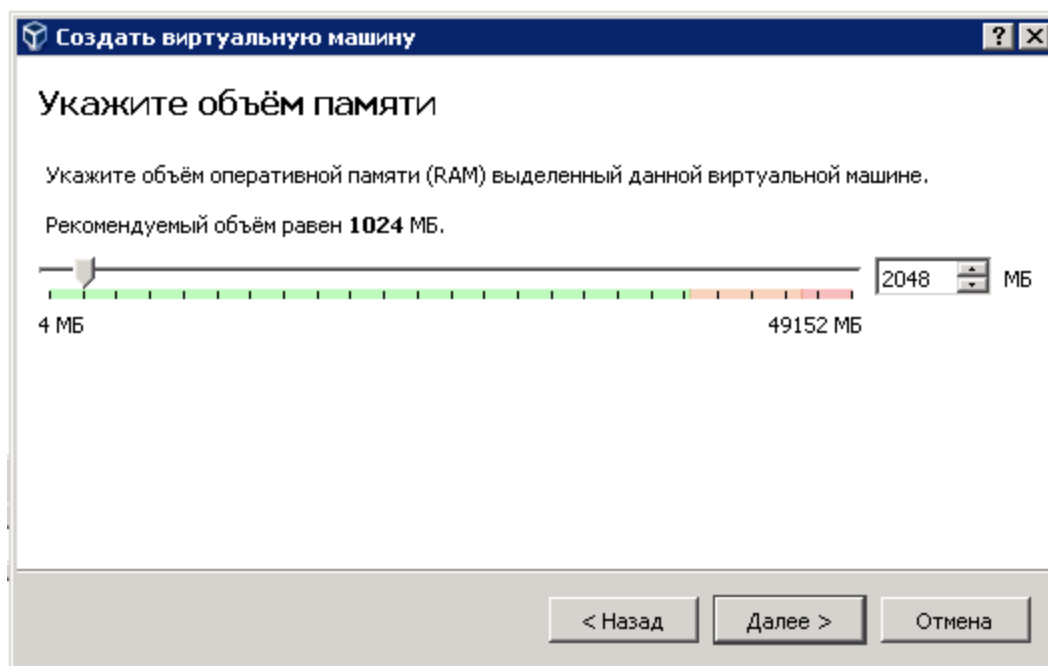
**ВНИМАНИЕ:** Для Linux как и для Unix характерна регистрозависимость.

**ЛАБОРАТОРНАЯ РАБОТА 2.1. УСТАНОВКА И ВХОД В СИСТЕМУ.****Упражнение 1. Установка операционной системы**

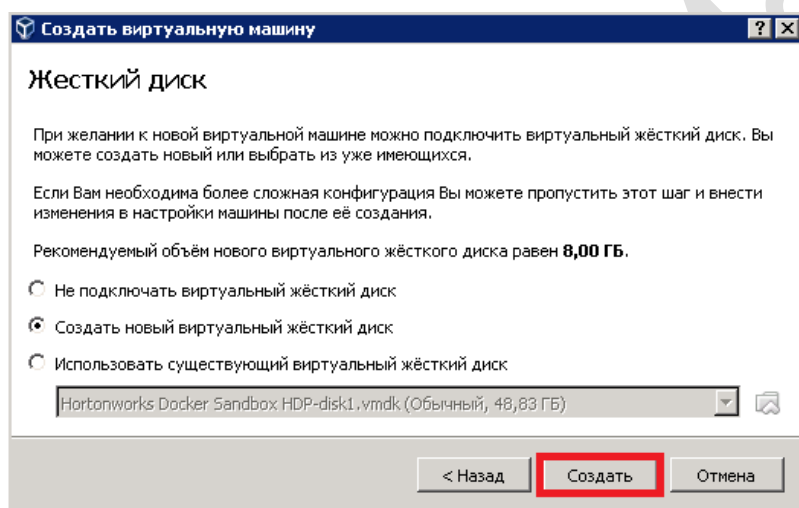
1. Создайте виртуальную машину в VirtualBox:



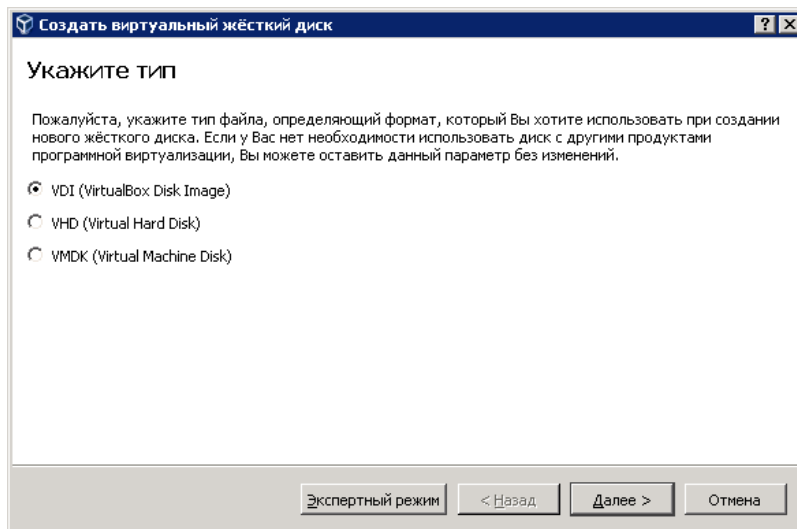
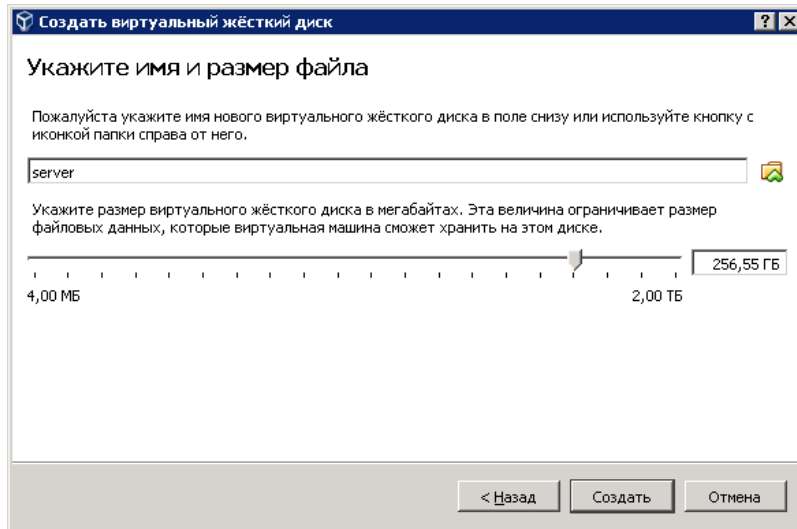
**Задайте объем памяти 2048 Mb**



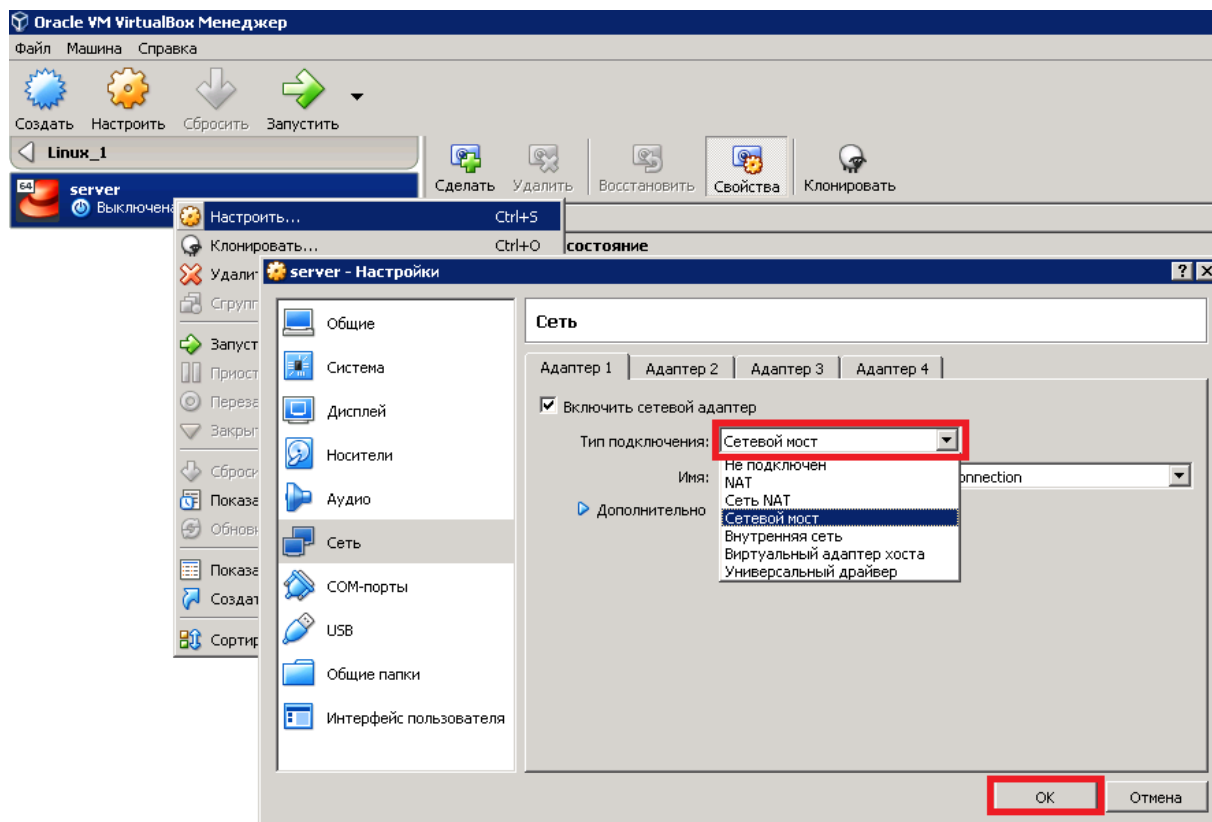
**Создать новый диск – тип VDI, формат: Динамический. Размер ~256 Гб**



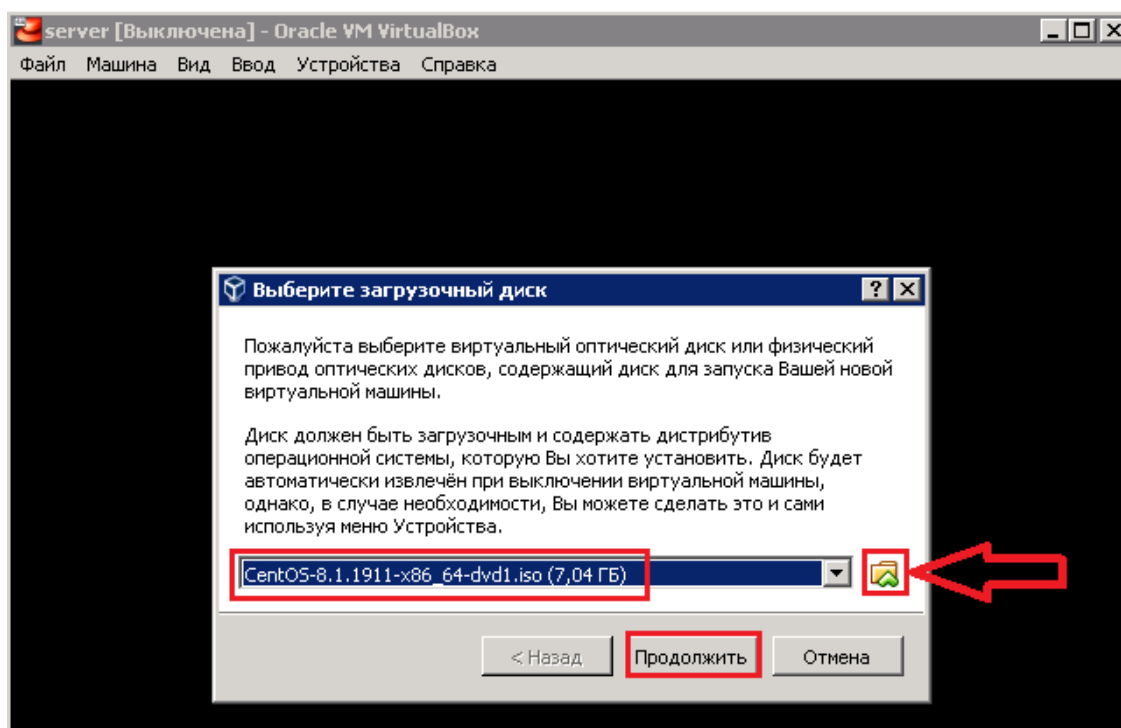




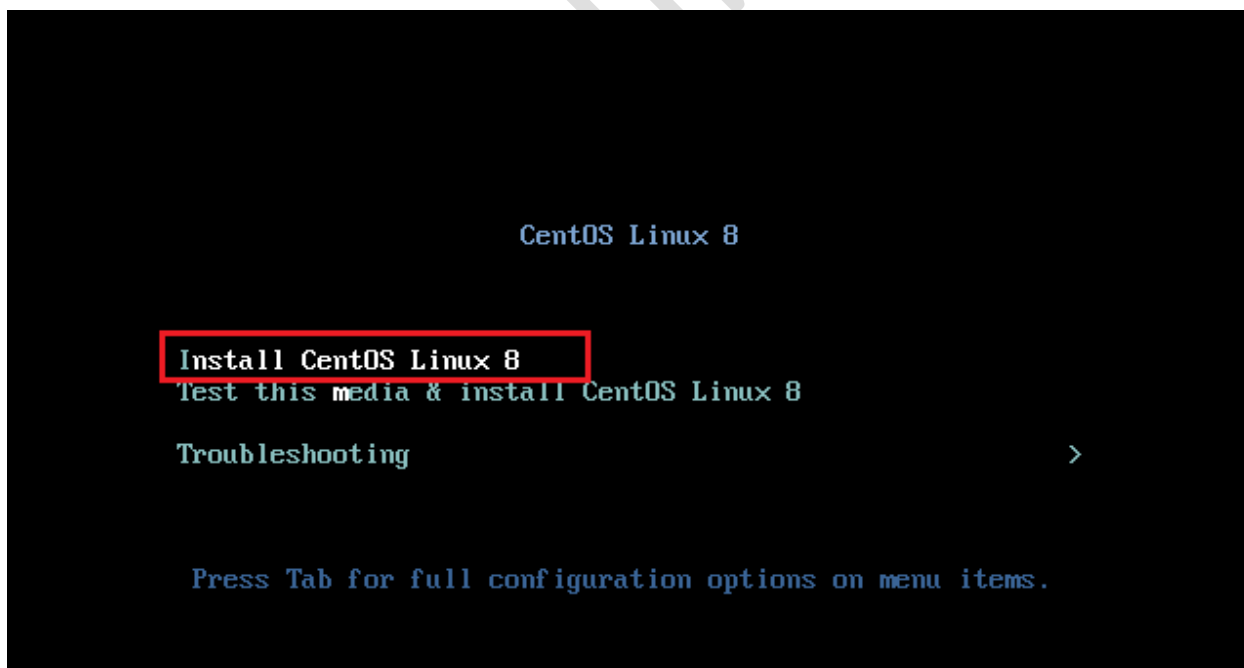
Настроить сетевой адаптер как Сетевой мост:



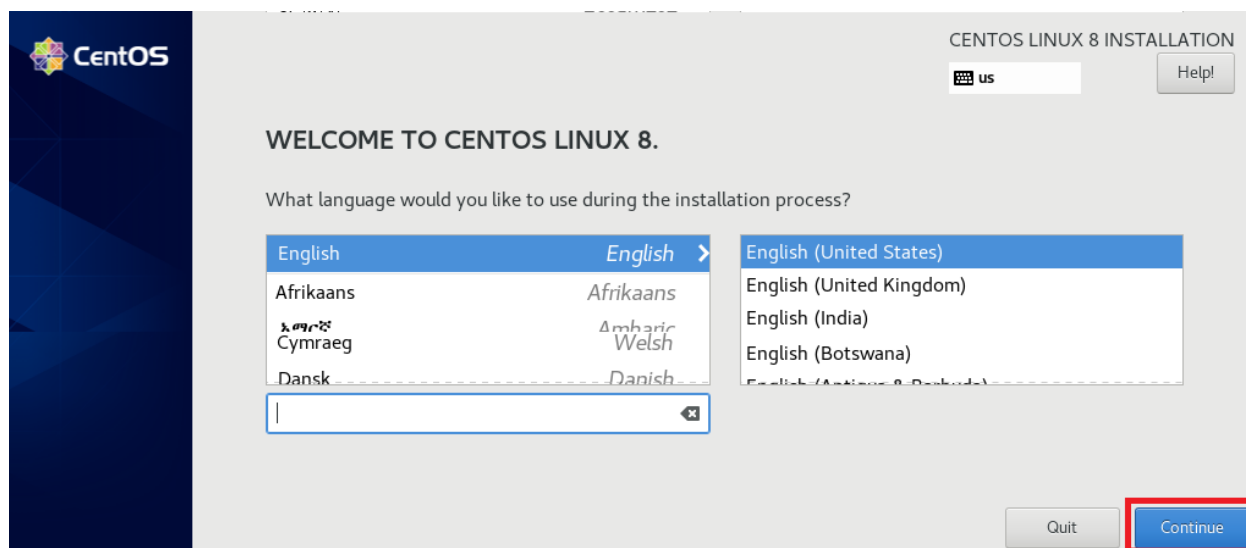
Запустить виртуальную машину и указать путь к файлу CentOS-8.iso:



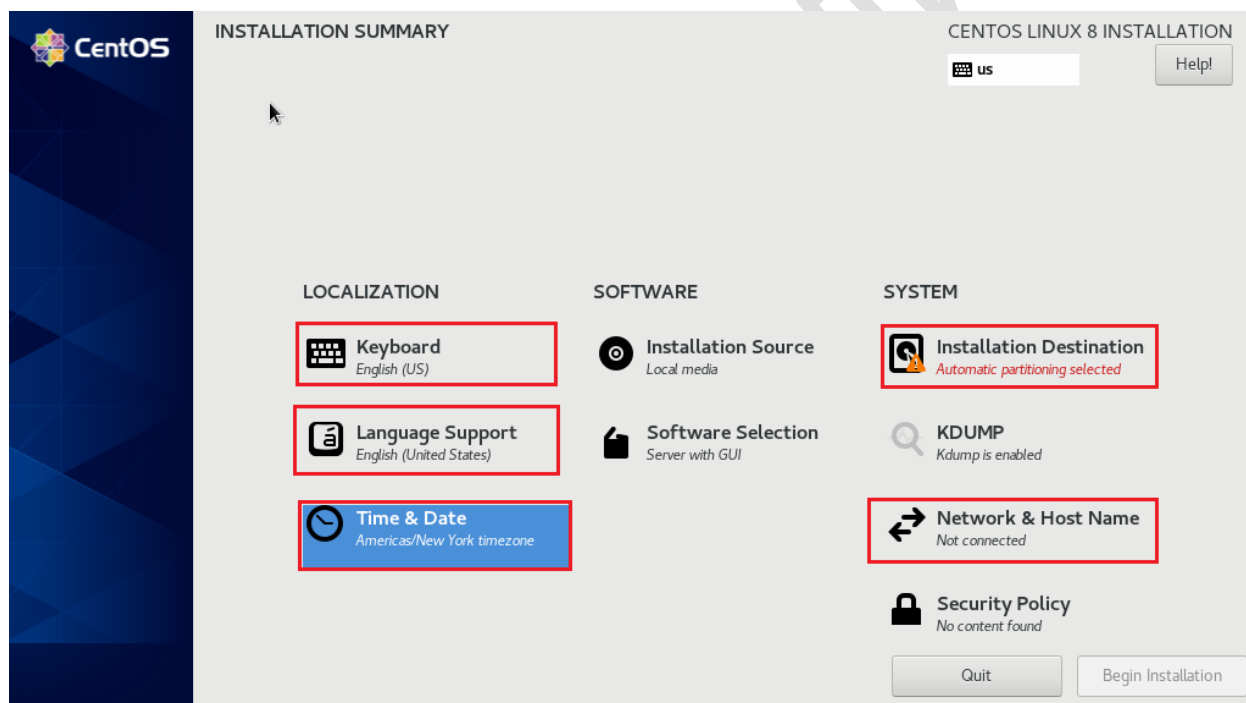
Выберите стрелками первую строку (Установка Centos Linux 8)



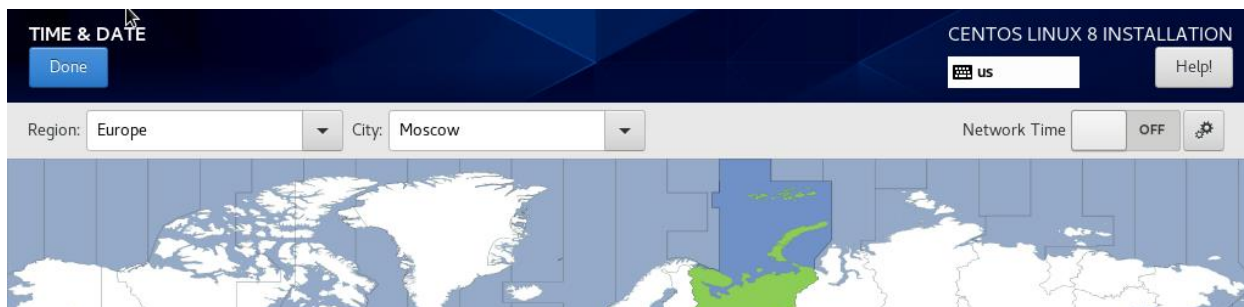
Опционально можно выбрать язык установки (по умолчанию английский) и нажать Продолжить(Continue):



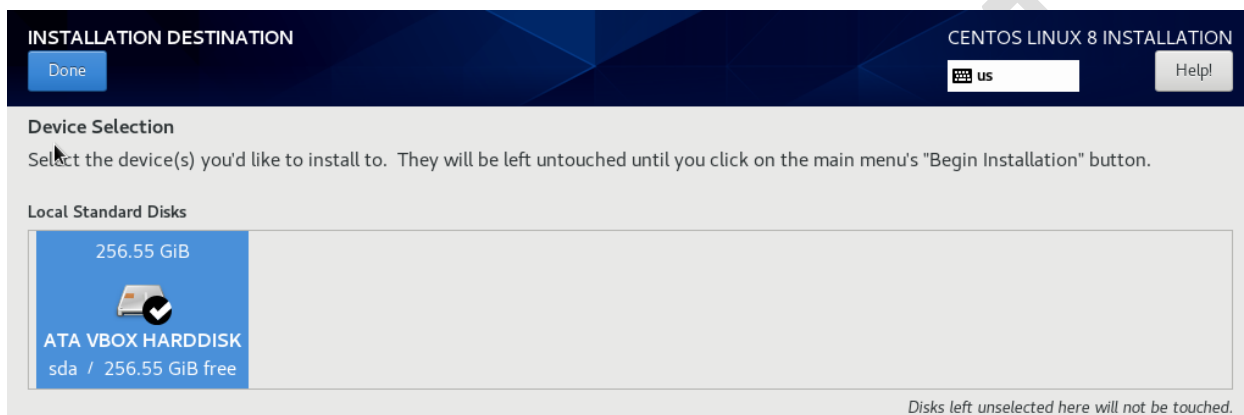
Принять настройки по умолчанию, кроме нижеследующих выделенных красным:



Выберите настройки времени и нажмите Done:



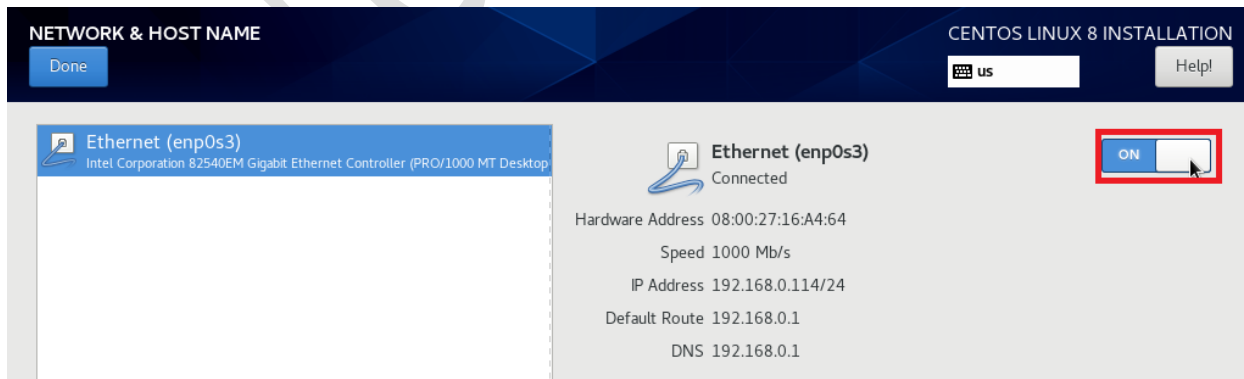
В меню **INSTALLATION DESTINATION** выберите диск (дважды щелкнув), на который будет установлена система:



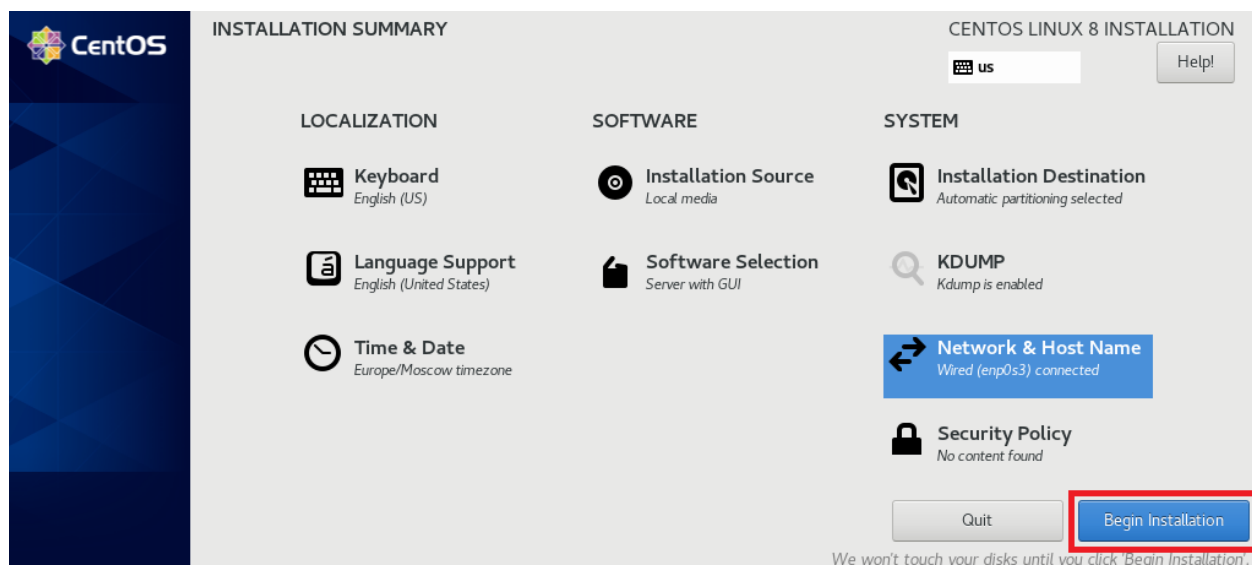
В меню **NETWORK & HOSTNAME**:

Включите сетевой адаптер (положение **ON**)

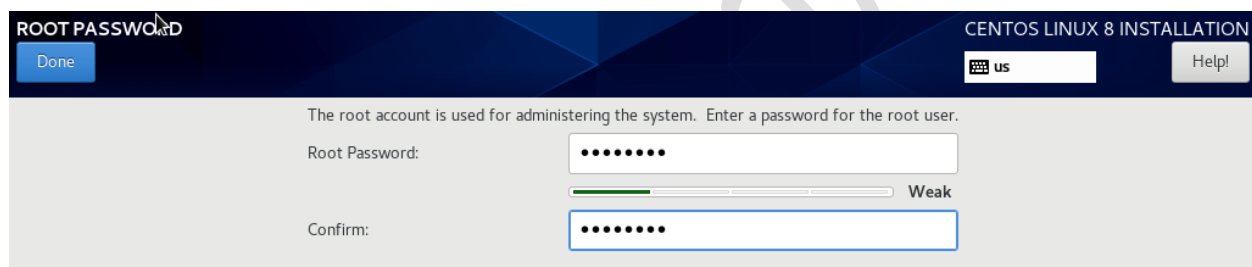
И можете записать назначенный IP-адрес для последующего подключения к системе по ssh.



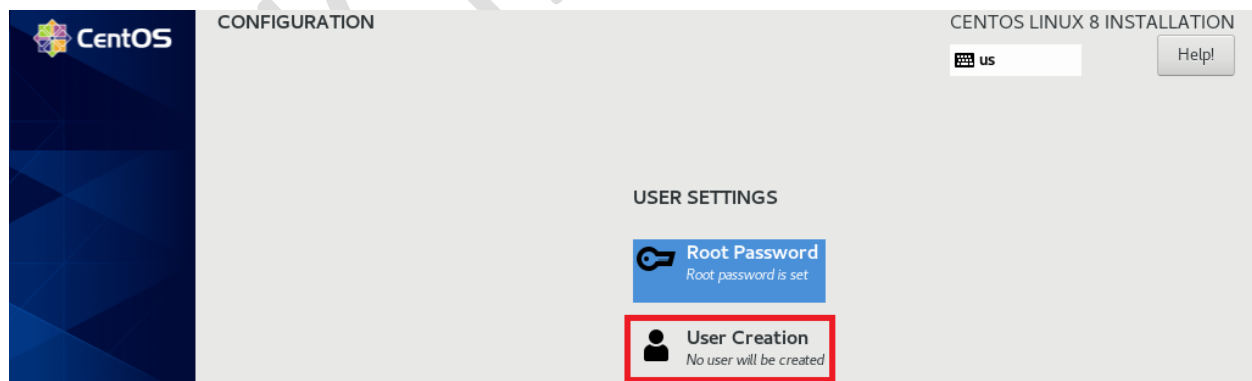
Нажать **Начать Установку (Begin Installation)**



Задать пароль для Root и нажать Done:



Затем выберите создать пользователя:



Создайте административного пользователя user1 и нажмите Done:

**CREATE USER** CENTOS LINUX 8 (CORE)

Done us Help!

Full name

User name

Tip: Keep your user name shorter than 32 characters and do not use spaces.

☒ Make this user administrator

☒ Require a password to use this account

Password  Too short

Confirm password

Advanced...

После перезагрузки войти в систему и принять лицензионное соглашение:

**License Information** CENTOS LINUX 8 (CORE)

Done us Help!

License Agreement:

CentOS 8 Linux EULA

CentOS 8 Linux comes with no guarantees or warranties of any sorts, either written or implied.

The Distribution is released as GPLv2. Individual packages in the distribution come with their own licences. A copy of the GPLv2 license is included with the distribution media.

☒ I accept the license agreement.

**Упражнение 2. Вход в систему.**

Цель работы: Получить практические навыки по основным моментам, связанным с работой в консоли.

1. Перейдите в текстовую консоль (виртуальный терминал)
2. Под каким пользователем вы находитесь в терминале: привилегированным или обычным?
3. Узнайте ip-адрес своей системы

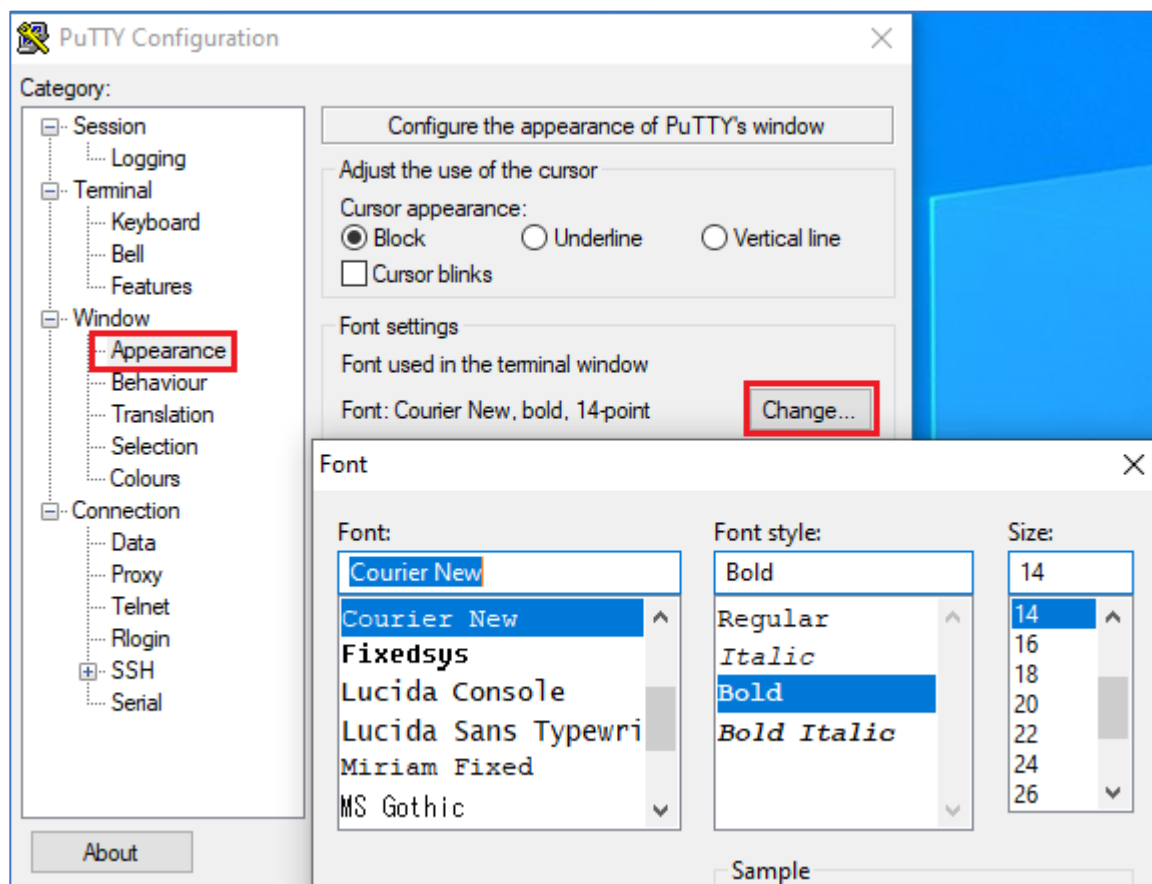
**Решение.**

1. Войдите в текстовую консоль (виртуальный терминал) нажатием комбинации клавиш Ctrl + Alt + F3
2. Обратите внимание на формат приглашения оболочки: \$ или #
3. Выполните команду ip a и запишите ip-адрес интерфейса

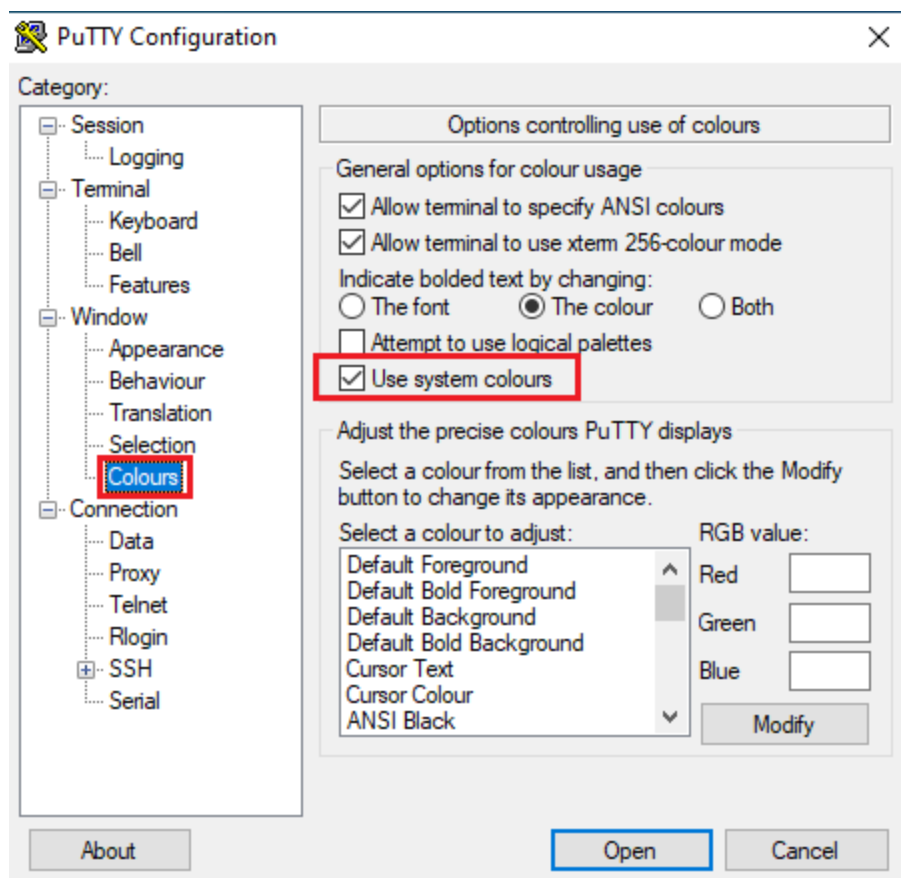


**ЛАБОРАТОРНАЯ РАБОТА 2.2. НАСТРОЙКА ПОДКЛЮЧЕНИЯ К СИСТЕМЕ****Упражнение 1. Настройка параметров PuTTY.**

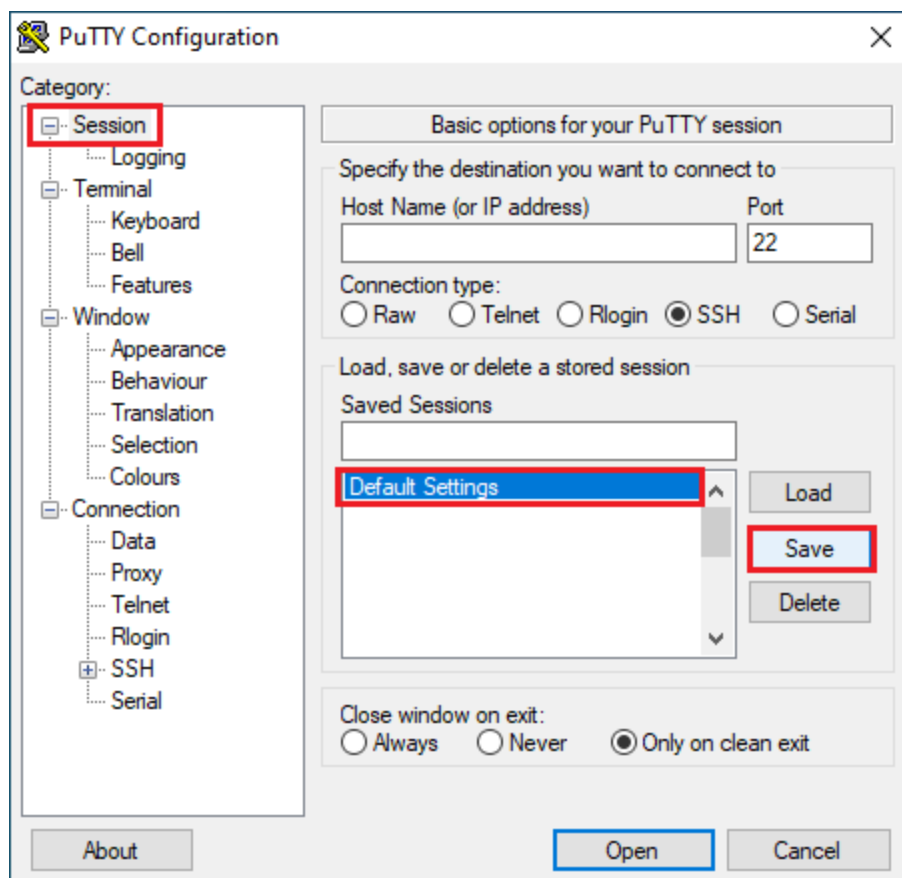
1. Запустите PuTTY и при желании настройте шрифт и фон терминала
2. Для этого в поле Category выберите Appearance и нажмите кнопку Change как показано на скриншоте ниже.



3. Выберите подходящий размер и начертание шрифта. Опционально можно выбрать и сам шрифт, но в данном примере в этом нет необходимости.
4. Далее, в поле Category перейдите на Colours и отметьте флажок Use system colours как показано на следующем скриншоте

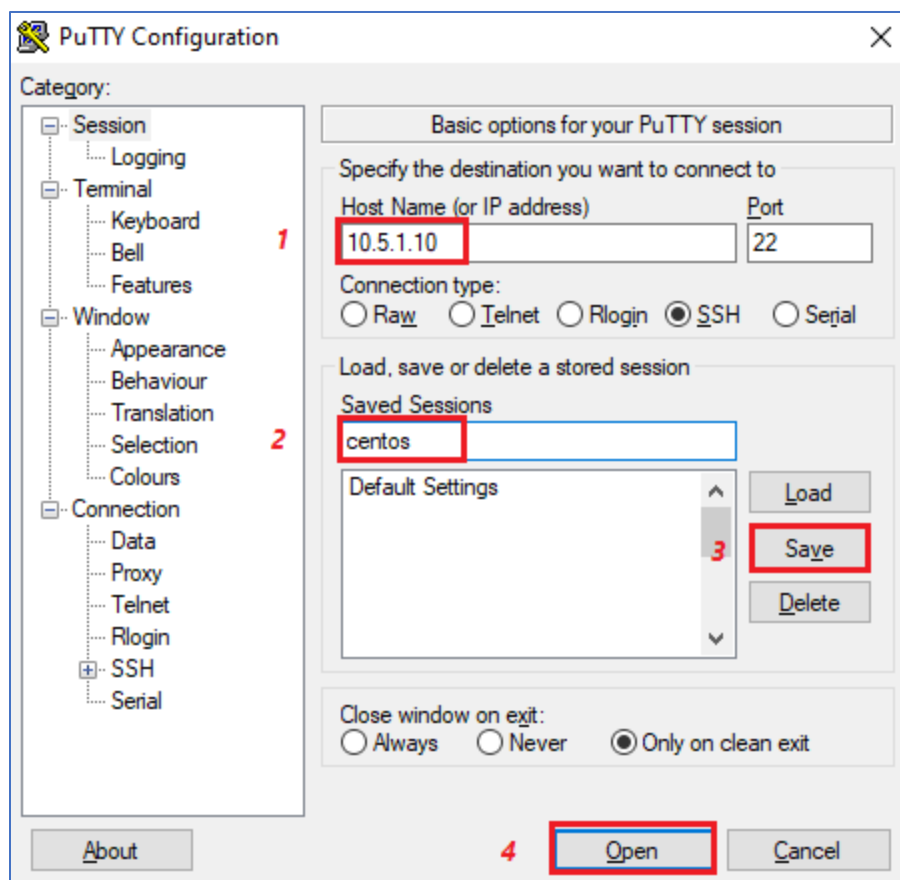


- Для сохранения сделанных настроек в качестве умолчательных в поле Category перейдите на Session и в поле Saved Sessions выделите курсором мыши Default Settings, затем нажмите кнопку Save (Сохранить) как показано ниже:



## Упражнение 2. Настройка подключения к системе.

1. В PuTTY поле Category убедитесь что выбрано Session
2. В поле Host Name (or IP address) введите ранее отображенный командой «ip a» адрес установленной системы и перейдя в поле Saved Sessions введите произвольное имя под которым Вы сохраните это подключение (например, **server**), затем нажмите кнопку Save Теперь, для подключения нажмите Open (или просто дважды щелкните на имени сохраненного подключения) как показано на скриншоте ниже.



3. Подтвердите в появившемся окне подлинность (значение ключа) подключения
4. Введите свои логин и пароль.

## 2.2. Структура команд и формат ключей.

Командами могут быть как встроенные директивы самой оболочки (cd, pwd, echo), так и вызываемые из под нее утилиты (ls, cat, grep, vi)

Командная строка в bash составляется из имени команды, за которым могут следовать ключи (опции) - указания, модифицирующие поведение команды. Ключи начинаются с символа - или --

Пример: ls --all и ls -a

Далее могут следовать аргументы (параметры) - имена объектов, над которыми должна быть выполнена команда

Команда –ключ (или опция) аргумент (над чем производится действие)

Короткий формат ключей: -n

Длинный формат: --option

Linux - регистрозависимая ОС.

включить репозиторий EPEL в вашей системе, а затем установить пакет автозаполнения bash и некоторые дополнительные функции, используя диспетчер пакетов YUM, таким способом:

```
yum install -y epel-release
```

включить автозаполнение команд в Bash

```
# yum install bash-completion bash-completion-extras
```

### 2.3. Работа со справочной информацией.

Документация

В случае минимальной установки документация на стандартные команды может отсутствовать.

Установит ее можно командой:

```
# yum install -y man man-pages-ru man-pages-overrides
```

man (от англ. manual — руководство) — команда, предназначенная для форматирования и вывода справочных страниц. Поставляется почти со всеми UNIX-подобными дистрибутивами.

Синтаксис: man [опции] [раздел] manpage

Пример: man -L ru 1 ls

Для чтения на русском добавить переменную в ~/.bashrc

```
export MANOPT="-L ru"
```

или

```
# man -L ru man
```

если есть соотв мануал на русском (см. в /usr/share/man/ru/man1)

**Примечание:** переменные окружения рассматриваются далее в курсе

Справочные страницы поделены на 8 стандартных разделов и один дополнительный. Каждый из разделов посвящен соответствующей тематике.

Раздел Описание

- 1 Исполняемые программы или команды оболочки (shell)
- 2 Системные вызовы (функции, предоставляемые ядром)
- 3 Библиотечные вызовы (функции, предоставляемые программными библиотеками)
- 4 Специальные файлы (находящиеся обычно в каталоге /dev)
- 5 Форматы файлов и соглашения
- 6 Игры
- 7 Различные описания, соглашения и прочее
- 8 Команды администрирования системы, которые обычно запускаются от имени суперпользователя
- 9 Процедуры ядра [нестандартный раздел]

Опция --help

Для получения краткой информации о программе, написанной сообществом GNU, можно использовать опцию --help

Пример: ls --help

Система помощи info

Также разработана сообществом GNU.

Пример: info ls

### ЛАБОРАТОРНАЯ РАБОТА 2.3. НАЧАЛО РАБОТЫ В ТЕРМИНАЛЬНОМ ПОДКЛЮЧЕНИИ

#### **Упражнение 3. Получение информации о системе и подключенных к ней пользователях**

1. Введите команду `who` и посмотрите результат

# `who`

2. Передайте вывод команды `who` другой команде (`wc`) для последующей обработки

# `who | wc -l`

# `wc` (от англ. word count — «подсчет слов»)

где `l` - число строк в файле;

`w` - число слов в файле;

`c` - число символов в файле

3. Получите сведения о вошедшем пользователе, выполнив следующие команды:

`id`

`whoami`

Команда для вывода всех встроенных инструкций `Bash`

4. Введите следующие команды:

\$ `date`

\$ `hostnamectl`

5. Введите команду:

\$ `help`

\$ `help | less`

Выведите справочную информацию по команде `man` на русском языке

\$ `man -L ru man`

### ГЛАВА 3. СТРУКТУРА ФС В LINUX.

#### **3.1. Типы файлов**

Для просмотра содержимого каталога и информации о файлах используется команда `ls`

Например, в выводе `ls -al`

- регулярный файл

d – директория (каталог)

. –текущий каталог

.. – родительский каталог

Файл каталога (d) - представляет собой определенный тип файла, который используется для связывания имен файлов и дескрипторов (inodes).

Файл устройств:

Символьные (c) и блочные устройства (b)

Файлы устройств предназначены для обращения к аппаратному обеспечению компьютера.

К символьным устройствам обращение происходит последовательно (символ за символом). Примером символьного устройства может служить терминал.

Считывать и записывать информацию на блочные устройства можно в произвольном порядке, блоками определенного размера. Пример: жесткий диск.

Сокеты (s) и каналы (p)

Каналы и сокеты организуют взаимодействие процессов.



### 3.2. Стандарт Иерархии Файловой Системы

FHS Linux - стандарт иерархии файловой системы Linux, первоначально определенный Free Standards Group, и теперь Linux Foundation, определяет основные каталоги, которые должны быть представлены в системе и их цели. Он может быть загружен с <http://www.pathname.com/fhs/>

Стандартные каталоги:

Обязательное ПО

/bin - Содержит исполняемые программы и скрипты, необходимые как системным администраторам, так и непривилегированным пользователям, которые необходимы, когда никакие другие файловые системы еще не были смонтированы, например, при загрузке в пользователь sin-gle или режиме восстановления.

/sbin – системное ПО

Дополнительное ПО

/usr/bin и /usr/sbin - пользовательское ПО

/home/userX

**Демонстрация.** Утилита tree. Отобразить дерево директорий, начиная от **указанной**,

-L – уровень вложенности

```
# tree -L 1 /
```

### 3.3. Устройство файловой системы. Inode

В файловой системе UNIX-подобных ОС можно выделить следующие составляющие:

- суперблок (superblock) и его резервные копии;
- блоки хранения данных;
- информационный узел (information node, inode);

Для эффективного доступа пространство раздела диска разбивается на блоки фиксированного размера: 1024, 2048 или 4096 байт (кратными размеру сектора – 512 байт)

Суперблок – это основной элемент файловой системы, представляющий собой некоторую таблицу различных значений, определяющих параметры ФС и местоположение её составных частей. Если не считать смещение в 1024 байта от начала файловой системы, то он идёт самым первым

Суперблок считывается при монтировании файловой системы и до ее размонтирования размещается в оперативной памяти.

Блоки объединяются в группы.

В каждой группе, помимо информационных блоков, хранится информация о занятости блоков и inode группы в виде битовой карты (Block Bitmap и Inode Bitmap)

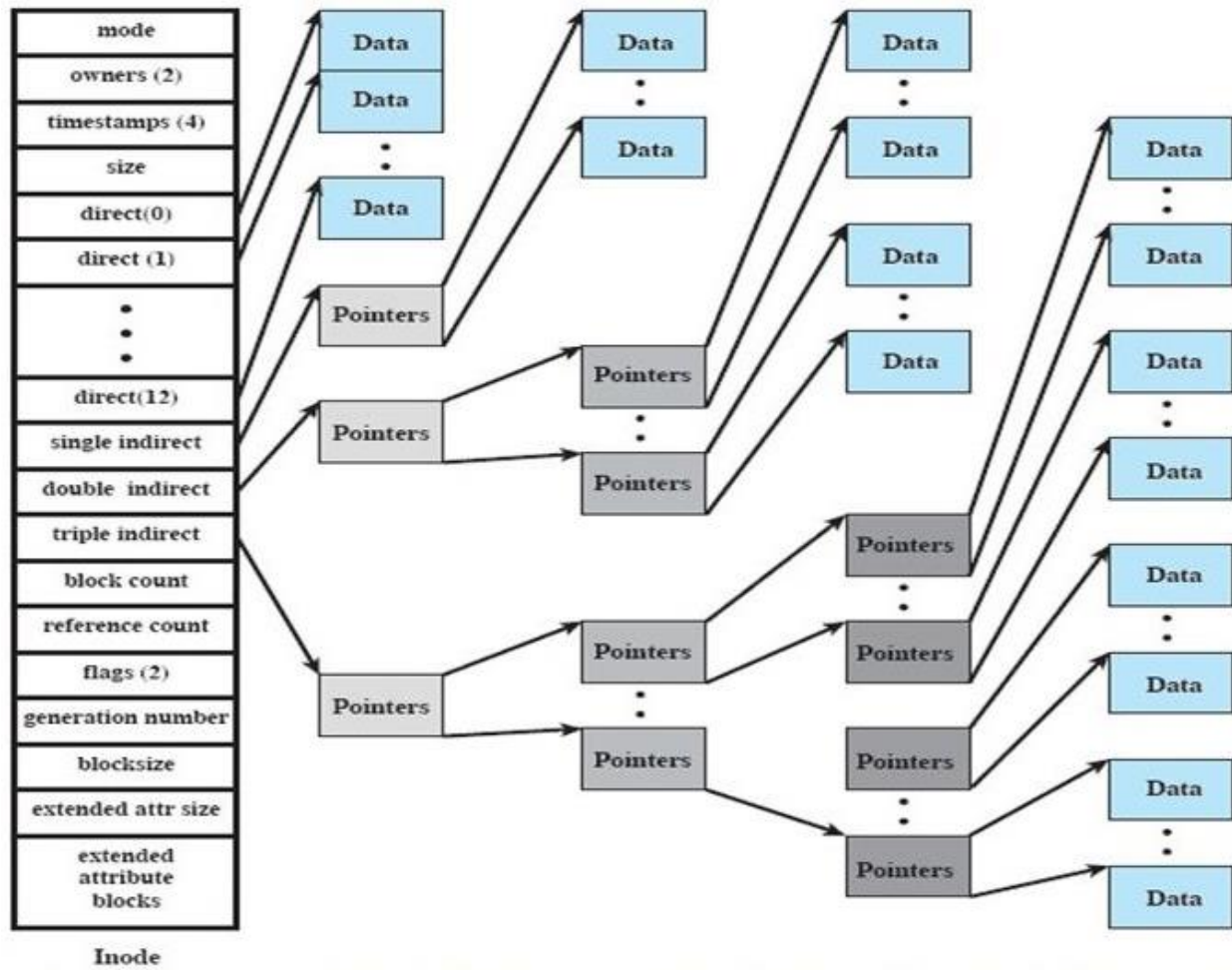
inode содержит информацию о «физическом» расположении данных. Связь логического имени файла с inode производится с помощью жёстких ссылок. Жёстких ссылок (как и логических имён) у одного файла может быть несколько. Например, тип файла – директория, в которую можно попасть как сверху, например по имени dir1 (cd dir1), так и снизу, например из низлежащих поддиректорий, переходом в директорию с именем «..» (cd ..).

Посмотреть информации по использованию инодов можно посредством df -Thi

Посмотреть номера inode у логических имён файлов (и директорий) можно с помощью команды ls -li либо с помощью утилиты stat.

Внутреннее устройство inode следующее: это специальная структура (таблица), которая содержит информацию об атрибутах и физическом расположении файла или метаданные

**Структура инода.**



### Базовые утилиты для работы с файлами

Кроме использования текстовых редакторов, для создания пустого файла можно воспользоваться командой `touch`, изначально предназначенной для установки времени последнего изменения файла или доступа в текущее время

Для чтения файла может использоваться утилита `cat` предназначенная для вывода указанного файла или файлов в поток вывода

Для больших файлов `cat` из-за отсутствия скроллинга использовать неудобно. Для этих задач подойдут утилиты `more` и `less`

`more` - скроллинг вниз (b – вверх)

`more /etc/passwd`

`less` - скроллинг вверх и вниз, PgUp / PgDwn работает

Выход из команды: q

Для просмотра логов используются утилиты `tail` и `head`.

`tail` выводит инфу на момент запуска программы. По умолчанию, выводит 10 последних строк. Можно переопределить параметром `-n`.

В терминальном подключении (PuTTY) выполните команду:

`tail -n 20 /var/log/secure`

Затем, выполните команду:

`tail -fn0 /var/log/secure`

Войдите в систему локально (в текстовую виртуальную консоль)

В терминальном подключении посмотрите появившиеся новые записи в журнале

Для выхода нажмите `Ctrl + C`

Противоположно предыдущей команде, `head` выводит начало файла

### 3.4. Использование ссылок в Linux

Есть два способа связать (два типа ссылок) имя файла с инодом:

Жесткие ссылки указывают на inode.

Символические ссылки (l) - указывают на имя файла, который имеет соответствующий индексный дескриптор.

Каждая ассоциация файла и инода известна как ссылка. Дополнительные ссылки могут быть созданы с помощью ln

Директория хранит карту сопоставлений в формате имя – inode.

Inode хранит информацию о дисковых блоках с контентом директории или файла, атрибут владельца и файловые разрешения. Hard link – еще одна запись вида: новое имя на существующий inode. Hard link должен быть в той же файловой системе, что и директория.

***Для директорий hard link не разрешены во избежание появления петель и циклов в дереве файловой системы***

Формат команды:

# ln файл ссылка

Пример вывода команды ln file1 file2

-rw-r--r-- 2 root root 0 марта 20 23:29 lnfile1 где 2 – число ссылок

Символьную ссылку можно создать при помощи команды ln с ключом -s (от "symbolic"). В качестве первого параметра пишется АБСОЛЮТНЫЙ АДРЕС и имя исходного файла, в качестве второго – адрес и имя ссылки. Например:

**ln -s /root/lnfile1 /var/softlfile1**

В отличие от жестких ссылок, символьные ссылки можно создавать и на каталоги.

### 3.5. Утилиты для поиска файлов

`which` – показывает путь к каталогу указанного исполняемого файла

`which vi`

`locate` - позв. найти файл, указанный по имени

`# yum install mlocate -y`

`locate hello`

`locate` работает с индексированной бд (отсюда и быстрый вывод результата) который периодически строится поверх `fs`.

команда `locate` может ускорить процесс используя внешнюю базу данных, генерируемую `updatedb`.

Команда `locate` ищет совпадения любой части пути, а не только самого файла. Во многих Linux системах есть «`cron job`» для периодического обновления базы. Вам необходимо запустить `updatedb` от `root`'а для генерации поисковой базы# `updatedb`

Работа программы `updatedb` может занять некоторое время.

`slocate`

Во многих Linux дистрибутивах, утилита `locate` была заменена на `slocate`. Как правило существует также ссылка на `locate`, так что вам не нужно запоминать, что именно имеется в системе. `slocate` означает «безопасный `locate`» (от англ. `secure locate` — прим. пер.). Он сохраняет информацию о правах доступа в поисковой базе, так что, обычные пользователи не смогут увидеть директории, которые они и так не смогли бы видеть

`which` – показывает информацию о программах (где находится. Ибо работает с переменной `PATH`, см. `# cat $PATH` ), а `locate` - показывает информацию о файлах

Команда `find`

`find / -name abcdef`

`find / -name abcdef 2>/dev/null`

### 3.6. Типы файловых систем

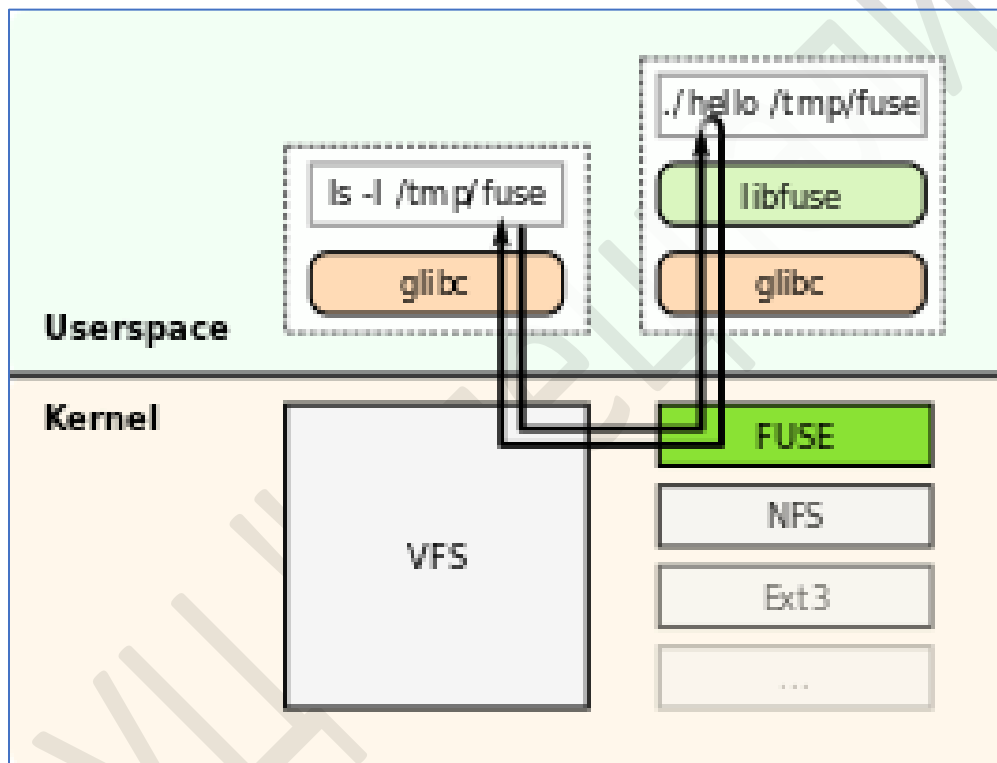
Кроме классических ФС в Linux поддерживаются временные (tmpfs) и виртуальные ФС

Процесс связывания файловой системы с устройством в Linux называется монтированием (mounting).

Для подключения файловой системы к существующей иерархии файловых систем (корню) используется команда mount (рассматривается далее в курсе).

Взаимодействие с различными, поддерживаемыми файловыми системами происходит через посредничество VFS.

Виртуальная файловая система (англ. virtual file system — VFS) или виртуальный коммутатор файловой системы (англ. virtual filesystem switch) — служит интерфейсом между ядром и конкретной файловой системой, позволяя легко добавлять поддержку новых типов файловых систем, внося изменения только в ядро операционной системы



**FUSE (англ. filesystem in userspace)** — модуль ядра, который позволяет создавать новые типы файловых систем, доступные для монтирования пользователями без привилегий за счёт запуска кода файловой системы в пользовательском пространстве при этом не прибегая к программированию на уровне ядра.

#### Файловая система /udev

Под udev понимается реализация файловой системы устройств devfs.

udev обеспечивает все необходимые средства для динамического создания и удаления файлов устройств и символических ссылок в каталоге /dev.

Содержимое каталога `/dev` хранится на виртуальной файловой системе, и все файлы, находящиеся в нём, создаются при каждом запуске системы.

### Файловая система `/proc`

Файловая система `/proc` содержит каталоги и виртуальные файлы. Виртуальный файл предоставляет полученную из ядра информацию, а также может использоваться для передачи в ядро пользовательской информации. Большинство программ получают информацию из файлов в `/proc`, например, **top** и **ps**.

`/proc` также источник информации об аппаратных средствах.

### `/proc/cpuinfo`

Здесь вы можете получить подробную информацию о процессорах

### `/proc/[pid]`

Имя каждого подкаталога соответствует *PID* (идентификатору процесса) работающего в системе процесса.

### `/proc/filesystems`

Этот файл содержит перечень всех поддерживаемых ядром типов файловых систем. Строки, начинающиеся с *'nodev'*, указывают на то, что ФС не является «физической».

### `/proc/cmdline`

В этом файле содержатся все аргументы, переданные ядру в момент старта системы.

### `/proc/loadavg`

Хранит информацию о загрузке системы. Первые три поля хранят ту же информацию, которую вы получаете о средней загрузке системы при помощи команды *uptime*.

Четвёртое поле состоит из двух значений, разделённых слэшем. Первая часть значения поля показывает количество выполняющихся в данный момент процессов/поток. Это значение не может быть больше количества присутствующих в вашей системе CPU. Вторая часть поля отображает количество процессов присутствующих в системе.

Значение пятого поля содержит ID последнего запущенного в системе процесса

### `/proc/meminfo`

Здесь хранится информация об использовании системой памяти

Подкаталог `/proc/sys` позволяет изменять некоторые параметры ядра в реальном режиме времени или отображать их.

Например `/proc/sys/kernel/hostname`



В этом файле содержится назначенное системе имя.

УЦ "Специалист"

### 3.7. Получение информации об устройствах хранения.

#### Формат идентификации дисков:

*sd [abcd] [12345] – диск – экземпляр – раздел*

Команды для вывода информации о доступных устройствах хранения:

`dmesg | grep sd`

`lshw -C disk`

Команда для вывода информации о разделах/томах на диске:

`fdisk -l /dev/sda`

Команда для вывода информации о файловых системах на разделах диска:

`file -s /dev/sda1`

Команда для вывода информации на какие каталоги смонтированы файловые системы и информация по ним.

`df -Th`

### 3.8. Регулярные выражения. Утилита `grep`

`# man 7 regex (man re_format в Ubuntu)`

Регулярное выражение – специализированный синтаксис, используемый для операций поиска и замены в тексте.

Регулярное выражение может содержать в себе один или больше нижеперечисленных элементов:

- Набор символов. Это символы, сохраняющие свои буквальные значения. Простейший пример регулярного выражения состоит только из набора символов и не содержит метасимволов.
- Якорь. Он (якорь) обозначает позицию текста в строке, которая совпадает с регулярным выражением. Например, символы `^` и `$` являются якорями.
- Модификаторы. Они расширяют или сужают (модифицируют) блок текста, совпадающего с регулярным выражением. Модификаторы включают в себя, например, звездочку (asterisk), квадратные скобки и обратную косую черту.

POSIX делит регулярные выражения на две категории:

BRE (Basic Regular Expressions) и ERE (Extended Regular Expressions).

Символы базовых регулярных выражений:

символ "точка" (`.`) соответствует любому одиночному символу;

\* – любое количество повторов предшествующего знаку «\*» символа/набора символов, в том числе и нулевое;

^ - соответствует началу строки, но, иногда, в зависимости от контекста, инвертирует символьный класс в регулярных выражениях.

[] - любой символ из числа заключенных в скобки (символы могут быть разделены запятыми, указание диапазона - [0-9]);

Пример: [012345789] – соответствует одному цифровому символу из заданного набора;

Предназначены для задания подмножества символов. Квадратные скобки, внутри регулярного выражения, считаются одним символом, который может принимать значения, перечисленные внутри этих скобок. Метасимвол ^ означает отрицание множества:

[^] - любой символ кроме тех что указаны в скобках;

Пример: `grep '[^rs]' /etc/passwd`

Классы символов:

[[:upper:]] - [A-Z];

[[:lower:]] - [a-z];

[[:digit:]] - [0-9];

[[:alnum:]] - [0-9a-zA-Z];

[[:space:]] - символ пробела;

[[:alpha:]] - [A-Za-z];

Пример: `# grep [[:upper:]] /etc/passwd`

\ - Служит для экранирования специальных символов, т.е. отменяет спец. значение следующего за ним метасимвола;

Пример: `grep 'bin\sh' /etc/passwd`

-- \<...\> -- Экранированные угловые скобки - задают границы слова (не работает в sed).

Пример: `grep -R '\<sed\>' /usr/share`

-- \( \) --Экранированные круглые скобки

Предназначены для выделения групп регулярных выражений. Они полезны при использовании с оператором «\|» и при извлечении подстроки.

-- \{ \} -- Экранированные фигурные скобки

Задают число вхождений предыдущего выражения. Для уточнения количества повторений наборов символов применяется модификатор `\{min,max\}`.

Пример: `grep '\(ro.*\)\{2\}' /etc/passwd`

`\{n\}` - указывает на то что предыдущий шаблон встречается ровно `n` раз;

`\{n,\}` - указывает на то что предыдущий шаблон встречается не менее `n` раз;

`\{,m\}` - указывает на то что предыдущий шаблон встречается не более `m` раз;

`\{n,m\}` - указывает на то что предыдущий шаблон встречается не менее `n` раз и не более `m` раз;

Символы расширенных регулярных выражений

Многие символы, экранируемые в базовых выражениях – `() {} |` – но не `- <>` – используются без экранирования.

Знак вопроса `-- ? --`

Означает, что предыдущий символ или регулярное выражение встречается 0 или 1 раз.

`$ grep -E '^r?o' /etc/passwd`

Знак "плюс" `-- + --`

Указывает на то, что предыдущий символ или выражение встречается 1 или более раз

### Утилита `grep`

Утилита `grep` позволяет вывести подстроки с нужными вхождениями и подсветить их из указанного файла. Кроме того, с помощью конвейера можно обрабатывать результаты вывода другой команды. `grep` умеет использовать регулярные выражения. Ключ `-v` позволяет исключить нужные подстроки. `grep` позволяет осуществлять выборку не только из входящего потока, но и из файла.

### Редактор `sed`

`sed` является потоковым редактором и умеет делать с файлами многое, как это делается в обычных текстовых редакторах, но по заданной команде и в скриптах. `sed` берет данные из потока или из файла, делает замены, и выводит данные в поток. Ключ `-i` позволяет перезаписать исходный файл. Редактор `sed` поддерживает работу с регулярными выражениями.

### Редакторы `vi` и `vim`

`vi` – штатный редактор для Unix и Linux систем. Поставляется с любым дистрибутивом.

`vim` – улучшенная версия `vi`

`vimtutor` – встроенный учебник по `vim`

J - вниз

K – вверх

L – вправо

G – в начало на первый символ

Редактирование текста

Режим вставки

i - ввод текста с текущей позиции

O - ввод текста с новой строки

Командный режим

J - склеить строки

x - удалить текст (DEL)

X - удалить текст (BACKSPACE)

yy - копировать строку в буфер

dd - вырезать строку в буфер

p - вставить строку из буфера под выбранной строкой

P - вставить строку из буфера над выбранной строкой

u - отменить последнее действие

: - командный режим

Например, во всем тексте заменить значение X на 1

:%s/X/1/g

**ЛАБОРАТОРНАЯ РАБОТА 3. СРЕДСТВА ПОЛУЧЕНИЯ ИНФОРМАЦИИ О ФАЙЛОВЫХ СИСТЕМАХ.**

**Цель работы.** Научиться применять базовые команды командной строки.

**Упражнение 1. Утилиты для получения информации о файловых системах.**

1. **Утилита file.** Чтобы определить, какая файловая система на разделе **/dev/sda1**, наберите в командной строке команду file с ключом -s:

```
file -s /dev/sda1
```

2. Утилита **df**. Наберите в командной строке следующие команды:

**# df -T** - показывает тип файловой системы.

**df -h** отображает информацию о смонтированных разделах с отображением общего, доступного и используемого пространства **/dev/sda1**

**Упражнение 2. Средства получения информации об использовании дискового пространства**

1. Выполните перечисленные ниже команды и посмотрите вывод:

**du /var** - подсчитывает и выводит размер, занимаемый директорией

**du -h /var** (- h оптимизирует отображение единиц измерения в выводе)

**du -h -s/var** (суммарный показатель объема занимаемого каталогом)

**du --max-depth=1 /usr/share/**

**Упражнение 3. Файлы и каталоги.**

1. Выполните нижеследующую команду, чтобы определить текущую директорию:

```
pwd
```

Выполните команду **ls** с предложенными ниже опциями, чтобы отобразить содержимое указанной директории (без параметров – текущей)

```
# ls -a
```

.имяфайла - скрытые файлы

```
# ls -l
```

```
# ls -al
```

2. Выполните предложенные ниже команды, чтобы проиллюстрировать переходы между директориями:

`cd ..` перейти в директорию уровнем выше

`cd ../../` перейти в директорию двумя уровнями выше

`cd` перейти в домашнюю директорию

`cd ~user` перейти в домашнюю директорию пользователя user

`cd` - перейти в директорию, в которой находились до перехода в текущую директорию

`cd /user/user1`

`cd ../../bin/bash`

`ls -a ~` (~ - домашний каталог)

### 3. Выполните предложенные ниже команды для создания и удаления каталогов.

`mkdir`

Создать дерево каталогов: `mkdir -p test1/test2/test2`

`ls -l`

`rm` – для удаления файла

`rm -r` - для удаления каталога рекурсивно, т.е. с содержимым.

`rmdir` – подходит только для удаления пустого каталога

### 4. Выполните команды для работы с файлами.

Копирование – утилита `cp`

Синтаксис: `cp` что куда

`cp /etc/passwd .` – копирование файла `/etc/passwd` в текущий каталог

Проверка: `ls`

Переименование, перемещение, в том числе под другим именем.

Синтаксис: `mv` что куда

### Упражнение 4. Работа со ссылками

1. Создайте файл `testfile1` с текстом «my test file» в каталоге `/var`
2. `nano /var/testfile1`
3. В своем домашнем каталоге создайте жесткую ссылку на него под именем `htestfile1`
4. `ln /var/testfile1 htestfile1`
5. Выполните команду `ls -l ~` и посмотрите на количество ссылок у файла `htestfile1`

**Упражнение 5. Эффекты от копирования ссылок.**

1. Создайте директорию /test1

```
sudo mkdir /test1
```

2. Перейдите в созданную директорию

```
# cd /test1
```

3. Создайте файл и жесткую ссылку на него в каталоге /var/test1

```
sudo touch file1
```

```
sudo ln file1 hfile1
```

Посмотрите результат

```
# ls -l
```

4. Перейдите в домашний каталог

```
# cd ~
```

**Выполните операции копирования и перемещения в пределах одного раздела ФС:**

**Жесткие ссылки**

Подготовьте целевые директории

```
sudo mkdir /test2 /test3
```

```
sudo cp /test1/* /test2
```

создаст 2 новых файла. Проверьте командой `ls -li /test2 /test1` (или просто посмотрев на количество ссылок у файла в целевой директории)

```
sudo mv /test1/* /test3
```

переместит ссылки

Проверьте командой `ls -li /test3`

Выполните `sudo rm -rf /test2 /test3`

**Символические ссылки:**

Подготовьте целевые директории

```
sudo mkdir /test2 /test3
```



Создайте файл и символическую ссылку на него в каталоге /var/test1

```
sudo touch /test1/file1
```

Создайте символическую ссылку:

```
sudo ln -s /test1/file1 /test1/sfile1
```

Посмотрите результат

```
ls -l /test1
```

Выполните копирование

```
sudo cp /test1/* /test2
```

создаст 2 новых файла

Посмотрите результат

```
# ls -l /test2
```

```
mv /test1/* /test3
```

переместит файл и ссылку

Выполните `ls -l /test3` и обратите внимание, что ссылка «битая», поскольку целевой файл поменял месторасположения.

### **Выполните операции копирования и перемещения между разделами ФС:**

Подготовьте целевые директории в другом разделе:

```
sudo mkdir /var/test2 /var/test3
```

И в своем домашнем каталоге ~user1

```
mkdir test1
```

Создайте файл и жесткую ссылку на него в своем домашнем каталоге ~user1

```
touch test1/file1
```

```
ln test1/file1 test1/hfile1
```

Посмотрите результат

```
ls -l test1/ | grep file
```

### **Жесткие ссылки**

```
sudo cp test1/* /var/test2
```

создаст 1 новый файл и 1 ссылку на него

```
ls -l /var/test2
```

#### **Символические ссылки:**

Создайте символическую ссылку:

```
ln -s /home/user1/test1/file1 test1/sfile1
```

Посмотрите результат

```
ls -l test1
```

```
sudo cp test1/* /var/test2
```

создаст 2 новых файла

```
sudo mv test1/* /var/test3
```

переместит ссылки

```
ls -l /var/test3
```

Удалите директории

```
sudo rm -rf test* /var/test*
```

И созданные в этом упражнении файлы из домашнего каталога

```
rm -f *file*
```

#### **Упражнение 6. Поиск по файловой системе**

```
$ which vi
```

```
$ locate hello
```

```
$ sudo updatedb
```

```
$ locate hello
```

```
$ touch file1
```

```
$ find ~ -name file1
```

```
$ locate file1
```

```
$ sudo updated
```

```
$ locate file1
```

### Упражнение 7. Поиск всех жестких ссылок файла (по иноду)

Выполните под root, поскольку его домашний каталог и каталог /var/ в одной файловой системе:

```
sudo -i
```

```
touch file1
```

```
In file1 /var/file2
```

```
ls -li
```

```
# find /var -samefile file1
```

### Упражнение 8. Утилиты для чтения файлов

cat - используется для вывода на экран содержимого файла.

Выполните cat /etc/passwd

#### Выполните команду more /etc/ssh/sshd\_config

Используйте Enter для прокрутки вниз

И пока не вышли из файла, клавишу b для прокрутки вверх

Также выполните команду less /etc/ssh/sshd\_config

less - скроллинг вверх и вниз, PgUp / PgDwn работает

Выход из команды: q

В терминальном подключении (PuTTY) выполните команду:

```
tail -n 20 /var/log/secure
```

Затем, выполните команду:

```
tail -fn0 /var/log/secure
```

Войдите в систему локально (в текстовую виртуальную консоль)

В терминальном подключении посмотрите появившиеся новые записи в журнале

Для выхода нажмите Ctrl + C

### Упражнение 9. Регулярные выражения. Утилита grep

Отобразите строки, начинающиеся с r из файла /etc/passwd

```
grep '^r' /etc/passwd
```

Выведите всех пользователей с назначенной оболочкой из файла /etc/passwd

```
grep 'sh$' /etc/passwd
```

Отобразите строки, начинающиеся либо с r, либо с s из файла /etc/passwd

```
grep '^[rs]' /etc/passwd
```

Выведите всех пользователей с оболочкой /bin/bash

```
grep 'bin/bash' /etc/passwd
```

Выполните две нижеперечисленные команды и сравните результат

(полного вывода дожидаться не нужно, прервите нажатием Ctrl + C)

```
grep -R 'sed' /usr/share
```

```
grep -R '\<sed\>' /usr/share
```

В чем отличие этих конструкций?

Выведите строки. Содержащие заглавные буквы:

```
grep '[:upper:]' /etc/passwd
```

Выполните нижеперечисленные команды

```
grep -E '^r?o' /etc/passwd
```

```
grep -E '[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}' /etc/resolv.conf
```

Опишите результат применения этих команд

## Упражнение 10. Текстовые редакторы

### РЕДАКТОР SED

Формат команды

sed команды\_редактирования [имя\_файла]

-i – редактировать файл

Перейдите в домашний каталог:

cd ~

Скопируйте в домашний каталог для экспериментов файл /etc/ssh/sshd\_config

cp /etc/ssh/sshd\_config new\_sshd\_config

А также создайте файл example.txt

cat > example.txt

Введите string1

Нажмите Enter

Введите 1111:0000:0000:0000:2222

Нажмите Enter

И нажмите Ctrl + d для завершения

Выполните следующие команды:

sed -i 's/string1/string2/' example.txt      в файле example.txt заменить "string1" на "string2", результат вывести на стандартное устройство вывода.

Используйте опцию -e (добавить к команде скрипт), чтобы вывести результат применения регулярного выражения:

sed -e '1d' new\_sshd\_config      удалить первую строку из файла example.txt

sed -n '/root/p' /etc/passwd      отобразить только строки содержащие "root"

sed -e 's/ \*\$//' new\_sshd\_config      удалить пустые символы в конце каждой строки

sed -e 's/root/d' /etc/passwd      удалить строку "root" из текста, не изменяя всего остального

sed -n '1,8p;5q' /etc/passwd      взять из файла с первой по восьмую строки и из них вывести первые пять

sed -n '5p;5q' /etc/passwd      вывести пятую строку

sed -e 's/0\*/0/g' example.txt      заменить последовательность из любого количества нулей одним нулём

Теперь модифицируйте содержимое файла удалив пустые строки и комментарии из файла new\_sshd\_config с помощью sed -i:

sed -i '/ \*#/d; /^\$/d' new\_sshd\_config

**Редактор Vi**

**Создайте в домашнем каталоге пример конфигурационного файла для экспериментов:**

```
cat > sample_ifcfg-enp0s3
```

Введите:

```
TYPE=Ethernet
```

```
BOOTPROTO=static
```

```
IPADDR0=192.168.X.10
```

```
PREFIX0=24
```

```
GATEWAY0=192.168.X.254
```

```
DNS1=172.16.1.254
```

Нажмите Ctrl + d для сохранения

во всем тексте заменить значение X на 1

```
:%s/\.X/\.1/g
```

Проверьте результат

```
cat sample_ifcfg-enp0s3
```

**ГЛАВА 4. НАСТРОЙКА (КОНФИГУРИРОВАНИЕ) ОС.**

**Настройка адреса через конфиг (используйте вместо X номер своего стенда):**

**Прим.** Также можно править конфиг утилитой nmtui.

```
# sudo vi /etc/sysconfig/network-scripts/ifcfg-enp0s3
```

```
TYPE=Ethernet
```

```
BOOTPROTO=static
```

```
IPADDR=172.16.1.X
```

```
PREFIX=24
```

```
GATEWAY=172.16.1.254
```

```
DNS1=172.16.1.254
```

```
...
```

**Прим.** DNS серверы нумеруются с 1-цы

**Для перезагрузки сети выполните:**

```
$ sudo -i
```

```
# systemctl restart NetworkManager
```

```
# nmcli networking off && nmcli networking on
```

**Альтернативный способ:**

```
ifdown enp0s3 && ifdup enp0s3
```

**Примечание.** скрипты для настройки сети (network-scripts) объявлены устаревшими и могут быть не доступны по умолчанию

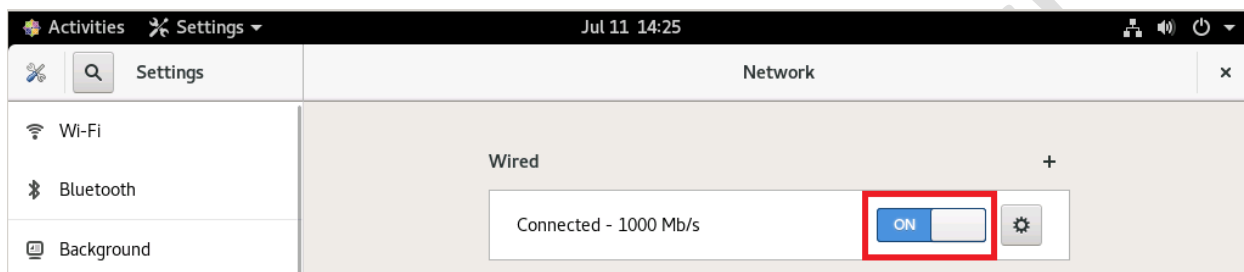
**Опционально. Настройка сети в графическом интерфейсе:**


Перейдите в графическую консоль (нажатием Ctrl + Alt+ F1)

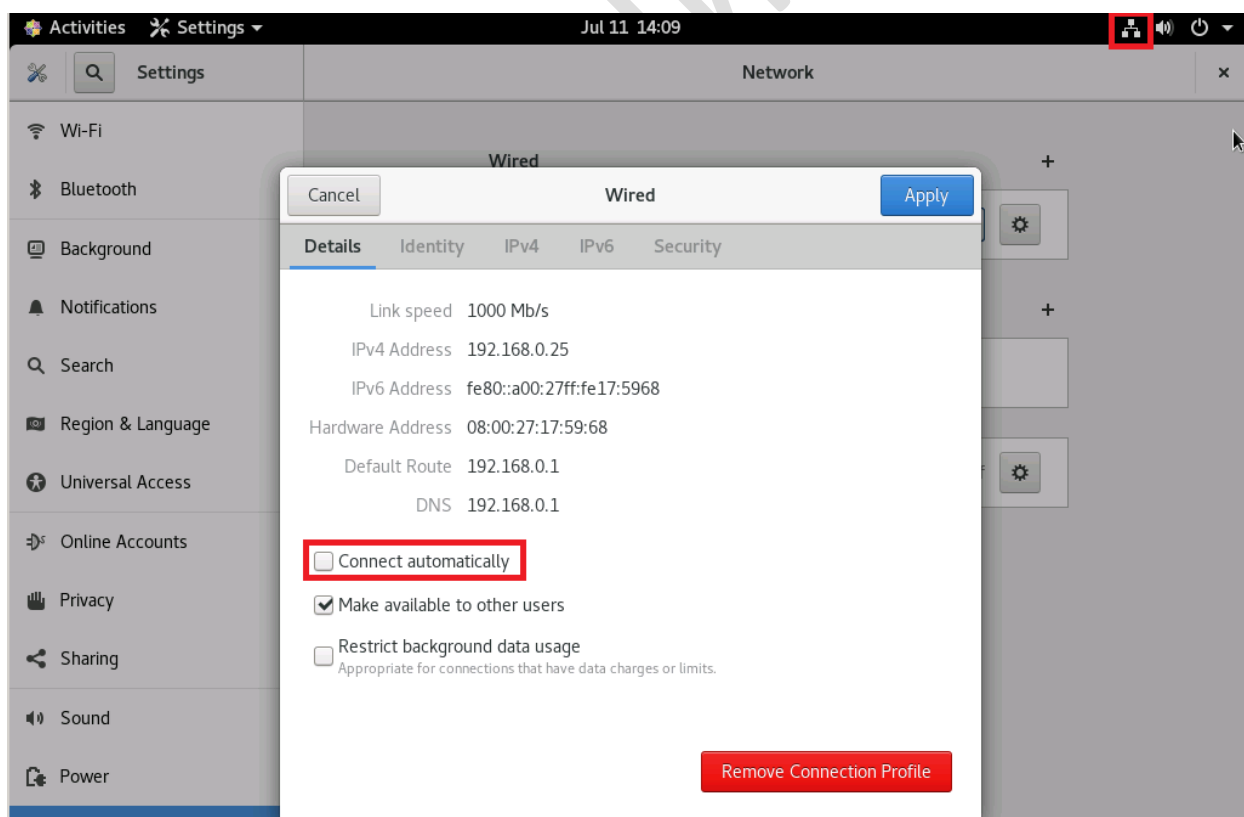
Войдите под user1

Activities – Show Applications - Settings - Network

Включить выбранный интерфейс (если был отключен)



Нажмите на кнопку  в поле Wired. Убедитесь, что интерфейс настроен на подключение при старте системы:



В этом же окне во вкладке IPv4 настройте адрес интерфейса и сопутствующие параметры:



Где X – номер стенда, предоставленный преподавателем.

IP-address: 172.16.1.X

Mask: 255.255.255.0

Default route: 172.16.1.254

DNS: 172.16.1.254

**Прим. Можно задавать настройки, не изменяя конфиг - на лету, но они будут действительны до перезагрузки:**

```
# ifconfig enp0s3 inet 172.16.1.X/24
```

```
# route add default gw 172.16.1.254
```

```
-----
```

```
ip addr add 172.16.1.X/24 dev enp0s3
```

```
ip route add default via 172.16.1.254
```

**Настройка файлов конфигурации.**

```
# cat /etc/hosts
```

```
# sed -i 's/localhost\.localdomain/server\.corpX\.un/' /etc/hosts
```

**Без перезагрузки применить новое имя:**

```
hostnamectl set-hostname server
```

**Опционально локализация и настройка часового пояса:**

**Локализация окружения**

Выполните команды для отображения настроек локализации:

```
# locale
```

```
# localectl status
```

```
# localectl list-locales
```

Для изменения системной локализации используется `localectl set-locale`, например:

```
# localectl set-locale LANG=ru_RU.utf-8
```

Русификация date:

```
dnf install -y glibc-langpack-ru
```

```
localectl set-locale LC_TIME=ru_RU.UTF-8
```

[https://serveradmin.ru/centos-8-locale-for-language-ru\\_ru-is-not-found-on-the-server/](https://serveradmin.ru/centos-8-locale-for-language-ru_ru-is-not-found-on-the-server/)

### **Меняем часовой пояс (сервер времени в след курсе)**

```
ls -l /etc/localtime
```

```
timedatectl set-timezone Europe/Madrid
```

**или**

```
# rm -f /etc/localtime
```

```
# ln -s /usr/share/zoneinfo/Europe/Moscow /etc/localtime
```

**Проверяем:**

```
# date
```

### **Управление сервисом timesyncd (В Centos7 альтернативный NTP клиент и сервер Chrony)**

```
yum install -y chrony
```

```
systemctl enable chronyd.service
```

**Конфиг здесь:**

**/etc/chrony/chrony.conf или /etc/chrony.conf**

**В Ubuntu:**

**Узнать состояние timesyncd позволяет команда timedatectl**

Если сервис timesyncd отключен, введите:

```
sudo timedatectl set-ntp on
```

```
timedatectl
```

В выводе должна быть строка:

Network time on: no

Теперь можно установить ntp:

```
sudo apt-get install ntp
```

**Версия ядра:**

```
uname -r
```

30 мая 2011 [Линус Торвалдс](#) выпустил ядро версии 3.0-rc1. Вместе с ним изменена политика нумерации версий ядра. Отменено использование чётных и нечётных номеров для обозначения стабильности ядра, а третье число означает стабильность ядра. Версия 3.0 практически не несёт никаких изменений, кроме изменения политики нумерации ядра. Таким образом, стабильные версии ядра 3.0 будут именоваться 3.0.X, а следующий после этого релиз будет иметь номер 3.1.

**Версия дистрибутива:**

```
# cat /etc/redhat-release
```

```
# cat /etc/centos-release
```

или

```
# rpm -q centos-release
```

**Модули хранятся в каталоге "/lib/modules/<версия ядра>" в виде файлов с расширением «ko».**

```
# find /lib/modules/`uname -r` -name *.ko
```

детальную информацию о модуле можно получить при помощи команды **modinfo**:

```
# modinfo rt73usb
```

modprobe – используется для добавления или удаления модулей ядра Linux

**Управление модулями ядра осуществляется следующими утилитами:**

- **lsmod** вывод всех загруженных модулей в виде таблицы.
- **modinfo** вывод информации о модуле: файл модуля, краткое описание, авторы, лицензия, параметры.
- **insmod** утилита для загрузки модулей ядра. Повторяет функционал modprobe название\_модуля.

- **rmmod** простая программа для выгрузки модулей. Повторяет функционал `modprobe -r название_модуля`.
- **modprobe** утилита для загрузки и выгрузки модулей.

В большинстве случаев загрузка модулей осуществляется одной из следующих команд:

```
sudo modprobe название_модуля
```

```
sudo insmod название_модуля
```

**Пример:**

```
modprobe -r e1000
```

```
insmod /lib/modules/4.4.0-31-generic/kernel/drivers/net/ethernet/intel/e1000e/e1000e.ko
```

Для того чтобы определенные модули загружались/не загружались во время старта системы их можно прописать в файл `/etc/modules`. Каждый модуль должен быть записан в отдельной строке. Строки начинающиеся со слова `blacklist` запрещают загрузку модуля.

### Управление параметрами ядра «на лету»

Команда `sysctl` используется для изменения параметров ядра во время выполнения.

#### **sysctl и директория /proc**

Доступные параметры перечислены в файле `/proc/sys/`. `Procfs` требуется для поддержки команды `sysctl` в Linux. Вы можете использовать `sysctl` для чтения и записи данных параметра.

Виртуальная директория `/proc/sys` предоставляет интерфейс для параметров `sysctl`, позволяя вам проверять и менять их.

Чтобы начать изучение того, что `sysctl` может изменить, выполните `sysctl -a`, и увидите возможные параметры. Значение `sysctl` загружаются во время загрузки системы из файла `/etc/sysctl.conf`.

Если вы хотите применить его в любой момент времени, вы можете сделать это с помощью команды `sysctl -p`.

Например:

```
# cat >> /etc/sysctl.conf
```

```
net.ipv4.ip_forward=1
```

### ЛАБОРАТОРНАЯ РАБОТА 4.1. НАСТРОЙКА СЕТЕВЫХ ИНТЕРФЕЙСОВ

Цель работы: Научиться настраивать сетевые интерфейсы используя командную строку и конфигурационные файлы.

Откройте для редактирования конфигурационный файл сетевого интерфейса

```
# sudo vi /etc/sysconfig/network-scripts/ifcfg-enp0s3
```

Найдите параметр BOOTPROTO и исправьте его значение на static

```
BOOTPROTO=static
```

Допишите в конец файла строки

```
IPADDR0=172.16.1.X
```

```
PREFIX0=24
```

```
GATEWAY0=172.16.1.254
```

```
DNS1=172.16.1.254
```

Замените X на предоставленное инструктором значение

Для применения изменений без перезагрузки системы выполните:

```
systemctl restart NetworkManager
```

```
nmcli networking off; nmcli networking on
```

Перенастройте адрес подключения в PuTTY на новый

#### **ЛАБОРАТОРНАЯ РАБОТА 4.2. ИЗМЕНЕНИЕ ИМЕНИ СИСТЕМЫ**

Выполните команду для отображения имени системы

```
hostnamectl
```

Смените текущее имя системы на имя server

```
hostnamectl set-hostname server
```

Откройте для редактирования файл /etc/hosts

И допишите в конец каждой строки новое имя системы:

```
127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4 server
```

#### **ЛАБОРАТОРНАЯ РАБОТА 4.3. УПРАВЛЕНИЕ МОДУЛЯМИ ЯДРА**

Выполните команду для отображения всех загруженных модулей ядра

```
lsmod
```

Выполните команду для получения информации о модуле e1000e

Определите, загружен ли модуль e1000e

```
$ lsmod | grep e1000e
```

Загрузите этот модуль

```
sudo modprobe e1000e
```

Проверьте результат:

```
lsmod | grep e1000e
```

Выгрузите этот модуль

```
sudo modprobe -r e1000e
```

**ГЛАВА 5. УПРАВЛЕНИЕ СИСТЕМОЙ ХРАНЕНИЯ ДАННЫХ****Работа с дисками**

Список оборудования (сообщения ядра) выводится:

```
dmesg | less
```

```
# dmesg | grep sd
```

```
# lshw – вывод инфы об оборудовании
```

```
# lshw -C disk – вывод инфы о дисках
```

```
# fdisk -l /dev/sda
```

Подключаем второй диск к контроллеру SATA.

Создать раздел

```
# fdisk /dev/sdb
```

Предупредит, что разделов нет

n - Создать раздел

P – pri partition

1

p – посмотреть разделы

Далее по умолч (Enter)

w – записать изменения

**Создать файловую систему**

```
# mkfs.ext4 /dev/sdb1
```

```
# file -s /dev/sdb1
```

**Смонтировать раздел**

```
# mkdir /disk2
```

```
# mount /dev/sdb1 /disk2
```

```
touch /disk2/file1.txt
```

```
ls -a /disk2
```

```
umount /disk2
```

```
ls -a /disk2
```

```
cat /etc/fstab
```

Чтобы не было путаницы с дисками после переподключения шлейфов, используется UUID

Можно узнать командой

```
# blkid
```

или

```
lsblk
```

нужно прописать монтирование при загрузке в файле /etc/fstab

...

Или UUID=xxxxxx /disk2 ext4 defaults 0 0

Или так /dev/sdb1 /disk2 ext4 defaults 0 0

Если ошибиться в fstab, система не загрузится.

Поэтому перед перезагрузкой проверяем:

```
mount /disk2/
```

```
ls -a /disk2
```

### **Управление Логическими Томами.**

```
# pvcreate /dev/sdb
```

Создаём группу томов vg1:

```
# vgcreate vg1 /dev/sdb
```

командой vgextend добавьте физический диск к существующей группе томов: vgextend vg1 /dev/sdb

```
# создаём раздел с именем «vol1», размером 1Gb
```

```
# lvcreate -n vol1 -L1G vg1
```



посмотреть сколько свободного места в группе томов можно командой:

```
# vgdisplay
```

информацию по созданным логическим томам

```
# lvdisplay
```

информацию по физическим томам

```
# pvdisplay
```

разделы что мы создали появятся в папке `/dev/[имя_vg]/`, точнее там будут ссылки на файлы

```
# ls -l /dev/vg1/vol1
```

форматируем наши разделы в файловые системы:

```
# mkfs ext4 -L volume1 /dev/vg1/vol1
```

создаём папку и подключаем новообразовавшиеся тома:

```
# mkdir /usr/volume1
```

```
# mount /dev/vg1/vol1 /usr/volume1
```

```
# umount /usr/volume1
```

добавляем нужные строки в `/etc/fstab`, например такие:

```
/dev/vg1/vol1 /usr/volume1 ext4 relatime 0 0
```

Проверяем:

```
# mount /usr/volume1
```

```
# cd /usr/volume1
```

```
# cat > file1
```

```
12345
```

```
#cd /
```

```
# umount /usr/volume1
```

```
# ls -a /usr/volume1
```

### **Пример создания зеркального тома (Mirrored volume)**

```
pvcreate /dev/sdc1
```

```
pvcreate /dev/sdc2
```

```
vgcreate vg1 /dev/sdc1 /dev/sdc2
```

```
lvcreate -m1 -L 100M -R 1 -n mirror vg1
```

```
mkfs.ext4 -L mirrored /dev/vg1/mirror
```

```
mkdir /mirror
```

```
mount -L mirrored /mirror
```

```
lvresize -r -L +150M /dev/vg1/mirror
```

### **Пример создания тома raid5**

```
pvcreate /dev/sdd1 /dev/sdd2 /dev/sdd3 /dev/sdd4
```

```
vgcreate vg2 /dev/sdd1 /dev/sdd2 /dev/sdd3 /dev/sdd4
```

```
lvcreate --type raid5 -L 100M -n stripped vg2
```

```
mkfs.ext4 -L stripped /dev/vg2/stripped
```

```
mkdir /stripped
```

```
mount -L stripped /stripped
```

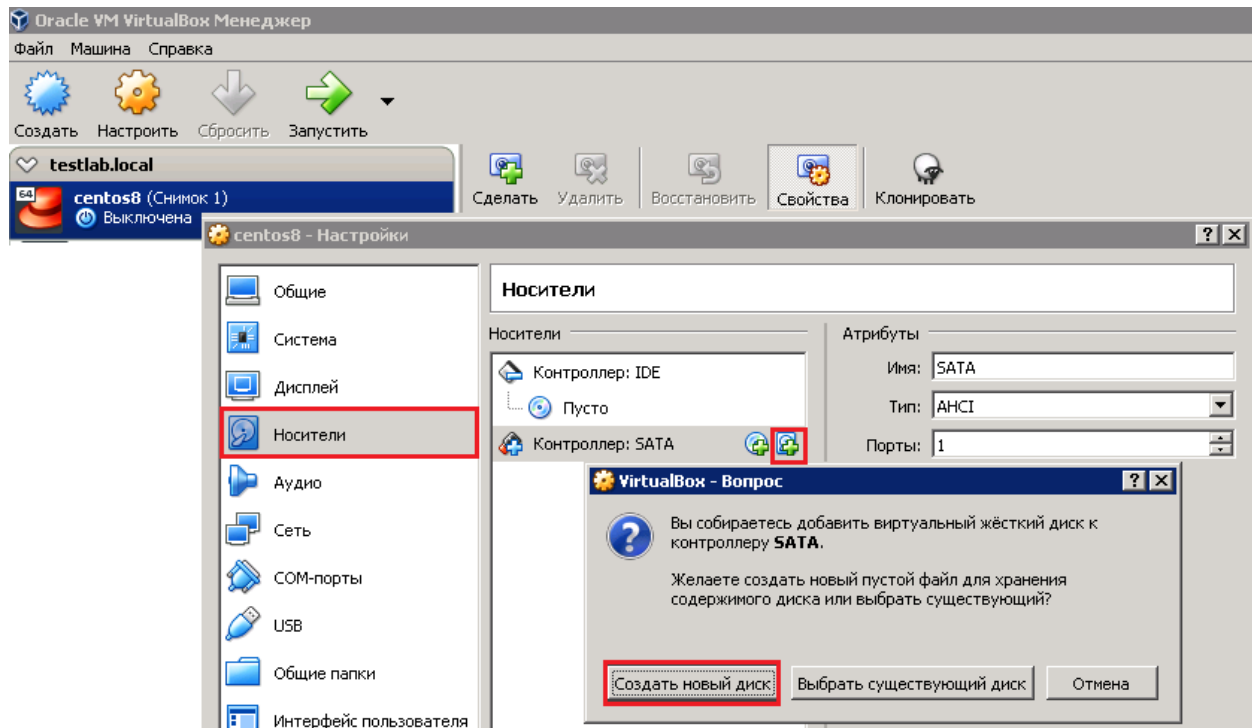
## **ЛАБОРАТОРНАЯ РАБОТА 5.1. ДОБАВЛЕНИЕ ДИСКОВ**

### **Упражнение 1. Добавление дисков**

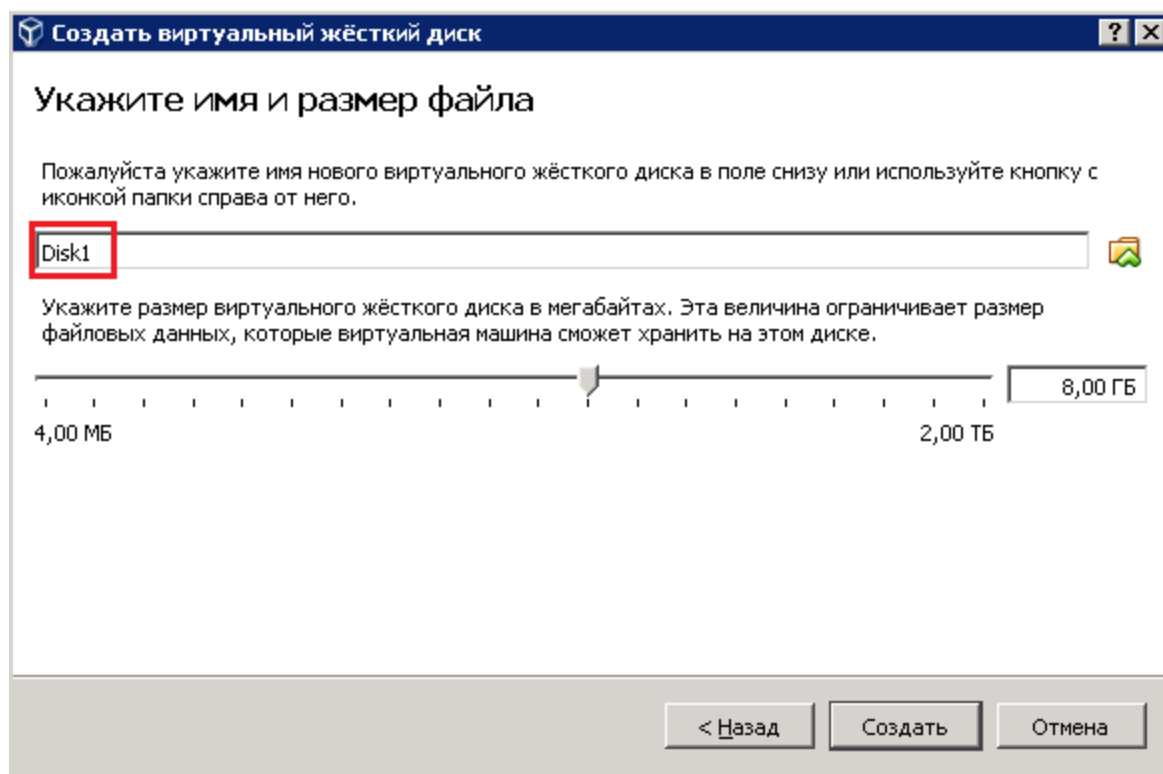
Выключите систему

```
sudo poweroff
```

Войдите в настройки виртуальной машины и в меню Носители добавьте жесткий диск к контроллеру SATA:



Примите все параметры по умолчанию, кроме имени диска, в качестве которого введите Disk1:



Повторите процедуру для подключения еще двух дисков – Disk2 и Disk3, соответственно.

Запустите виртуальную машину и подключитесь к системе по ssh (PuTTY).

Выполните команды для отображения информации о новых дисках:

```
dmesg | grep sd
```

```
sudo lshw -C disk
```

### Упражнение 2. Создание разделов

Создайте на устройстве /dev/sdb раздел размером 100 мегабайт

```
sudo fdisk /dev/sdb
```

Чтобы создать раздел, нажмите n

Оставьте по умолчанию P – primary partition

Оставьте предложенный по умолчанию номер раздела 1

Далее по умолч (Enter)

В финале нажмите w – записать изменения

**Упражнение 3. Создать файловую систему**

Отформатируйте раздел /dev/sdb1 файловой системой ext4

```
sudo mkfs.ext4 /dev/sdb1
```

Посмотрите сведения о файловой системк на разделе /dev/sdb1:

```
sudo file -s /dev/sdb1
```

**Упражнение 4. Смонтировать раздел**

Создайте директорию для монтирования раздела с ФС:

```
mkdir /disk2
```

Смонтируйте раздел ФС

```
sudo mount /dev/sdb1 /disk2
```

Проверьте доступность ФС для записи. Создайте в ней файл:

```
touch /disk2/file1.txt
```

```
ls -a /disk2
```

Размонтируйте файловую систему:

```
umount /disk2
```

```
ls -a /disk2
```

Воспользуйтесь командой

```
sudo blkid
```

Запишите UUID раздела /dev/sdb1

Добавьте запись в конец файла /etc/fstab

```
UUID=<Идентификатор раздела без кавычек> /disk2 ext4 defaults 0 0
```

Закройте файл с сохранением

Проверьте результат

```
sudo mount /disk2
```

**Упражнение 4. Работа с томами LVM**

Создайте физические тома на устройствах /dev/sdc /dev/sdd

```
sudo pvcreate /dev/sdc /dev/sdd
```

Проверьте результат:

```
sudo pvdisplay
```

Создайте группу томов vg1 из ранее созданных физических томов:

```
sudo vgcreate vg1 /dev/sdc /dev/sdd
```

Проверьте результат:

```
sudo vgdisplay
```

Создайте раздел с именем «vol1», размером 1Gb

```
sudo lvcreate -n vol1 -L1G vg1
```

Посмотрите информацию по созданным логическим томам

```
sudo lvdisplay
```

Отформатируйте том файловой системой ext4:

```
sudo mkfs ext4 -L volume1 /dev/vg1/vol1
```

Создайте директорию и смонтируйте том:

```
sudo mkdir /volume1
```

```
sudo mount /dev/vg1/vol1 /volume1
```

```
sudo umount /volume1
```

добавляем строку в /etc/fstab, например:

```
/dev/vg1/vol1 /usr/volume1 ext4 relatime 0 0
```

Проверьте результат:

```
# mount /volume1
```

```
# cd /volume1
```

```
# cat > file1
```

```
12345
```

```
#cd /
```

```
# umount /volume1
```

```
# ls -a /volume1
```

## ГЛАВА 6. УПРАВЛЕНИЕ СЕРВИСАМИ

### Systemd

Systemd – менеджер системы и сервисов в операционной системе Linux. При разработке его стремились спроектировать обратно совместимым со скриптами инициализации SysV init и предоставить полезные функции, такие, как параллельный запуск системных сервисов во время загрузки, активацию демонов по требованию, поддержку снапшотов состояния системы и логику управления сервисами, основанную на зависимостях. В CentOS 7 systemd заменяет Upstart как систему инициализации по умолчанию.

Systemd приносит концепцию юнитов systemd. Юниты представлены конфигурационными файлами, размещенными в

`/usr/lib/systemd/system/` – юниты из установленных пакетов RPM.

`/run/systemd/system/` – юниты, созданные в рантайме. Этот каталог приоритетнее каталога с установленными юнитами из пакетов.

`/etc/systemd/system/` – юниты, созданные и управляемые системным администратором. Этот каталог приоритетнее каталога юнитов, созданных в рантайме.

#### Типы юнитов systemd:

**service** – системный сервис

**target** – группа юнитов systemd

В юните, блок переменных, которые влияют на порядок загрузки сервисов:

Запускать юнит после какого-либо сервиса или группы сервисов (например `network.target`):

`After=network.target`

Для запуска сервиса необходим запущенный сервис `mysql`:

`Requires=mysql.service`

Для запуска сервиса желателен запущенный сервис `redis`:

`Wants=redis.service`

переменная `Wants` носит чисто описательный характер.

Если сервис есть в `Requires`, но нет в `After`, то наш сервис будет запущен параллельно с требуемым сервисом, а не после успешной загрузки требуемого сервиса

#### Управление сервисами

**`systemctl enable name.service`** – активирует сервис (позволяет стартовать во время запуска системы)



**systemctl disable name.service** – деактивирует сервис

Файлы целей systemd .target предназначены для группировки вместе других юнитов systemd через цепочку зависимостей. Например юнит graphical.target, использующийся для старта графической сессии, запускает системные сервисы GNOME Display Manager (gdm.service) и Accounts Service (accounts-daemon.service) и активирует multi-user.target. В свою очередь multi-user.target запускает другие системные сервисы, такие как Network Manager (NetworkManager.service) или D-Bus (dbus.service) и активирует другие целевые юниты basic.target.

Из соображений совместимости имеются алиасы на эти цели, которые напрямую отображаются в уровнях запуска SysV.

Теперь /sbin /init ссылка на /lib/systemd/system

А программы telinit и runlevel – ссылки на интерфейс управления системой инициализации systemd /bin/systemctl

Systemctl –для управления системой инициализации systemd и диспетчеризации служб.

**Таргеты можно увидеть здесь:**

```
# ls -l /lib/systemd/system/
```

```
runlevel0.target -> poweroff.target
```

```
runlevel1.target -> rescue.target
```

```
runlevel2.target -> multi-user.target
```

```
runlevel3.target -> multi-user.target
```

```
runlevel4.target -> multi-user.target
```

```
runlevel5.target -> graphical.target
```

```
runlevel6.target -> reboot.target
```

```
systemctl list-unit-files | grep ssh
```

```
systemctl enable ssh.service
```

```
systemctl start ssh.service
```

**Пример:**

```
root@server51:~# systemctl enable ssh.service
```

Synchronizing state of ssh.service with SysV init with /lib/systemd/systemd-sysv-install...

Executing /lib/systemd/systemd-sysv-install enable ssh

Created symlink from /etc/systemd/system/ssh.service to /lib/systemd/system/ssh.service.

## Процессы.

Процесс в Linux создается системным вызовом `fork()`. Процесс, сделавший вызов `fork()` называется родительским, а вновь созданный процесс — дочерним. Новый процесс является точной копией породившего его процесса.

Единственно, чем они различаются — это идентификатором процесса PID. Каждый процесс имеет одного родителя, но может иметь несколько дочерних процессов.

### yum -y install psmisc

# pstree

```
systemd├─agetty
        │
        └─atd
            │
            └─sshd├─sshd──bash
                  │
                  └─sshd──bash──pstree
```

Допустим, пользователь, работая в `bash` запускает команду `pstree`. Текущий процесс (`bash`) делает вызов `fork()`, порождая вторую копию `bash`. В свою очередь, порожденный `bash` вызывает `exec()`, указывая в качестве параметра имя исполняемого файла, образ которого необходимо загрузить в память вместо кода `bash`. Код `pstree` замещает код порожденного `bash`, и утилита `pstree` начинает выполняться.

## Мониторинг процессов

### Утилита top

Здесь показана команда и текущее системное время; общее время работы, количество зарегистрированных в системе пользователей;

load average: показывается средняя нагрузка;

Первая цифра показывает среднюю нагрузку «последней минуты», или «текущую» среднюю нагрузку; вторая цифра показывает «среднюю нагрузку за 5 минут», последняя цифра – «среднюю нагрузку за 15 минут».

Средняя нагрузка – мера среднего числа процессов, ожидающих своей очереди, чтобы совершить какое-либо действие в процессоре

Cpu(s): 0.0%us, 0.0%sy, 0.0%ni, 99.9%id,

### Утилита htop

В правом верхнем углу, **htop** показывает общее количество процессов (как задания [Tasks]) и сколько из них активны.

Возможные значения состояния:

R — [running or runnable] запущенные или находятся в очереди на запуск

- S — [interruptible sleep] прерываемый сон
- D — [uninterruptible sleep] непрерываемый сон (в основном IO)
- Z — [zombie] процесс зомби, прекращенный, но не забранный родителем
- T — Остановленный сигналом управления заданиями
- t — Остановленный отладчиком

### Межпроцессное взаимодействие

Благодаря межпроцессному взаимодействию (*Inter-Process Communication, IPC*) возможно разбить решение сложной задачи на несколько простых операций, каждая из которых доверяется отдельной программе. Последовательная обработка одной задачи несколькими простыми программами очень похожа на конвейерное производство.

Программы, использующие IPC, могут «общаться» друг с другом также, как и с пользователем, в результате чего появляется возможность автоматизировать выполнение сложных задач. Могушество скриптовых языков Unix и Linux во многом основано на возможностях IPC.

### Потоки данных

У каждой программы существует 3 системных потока: stdout, stderr, stdin

Программа, принимает ввод с клавиатуры через стандартный ввод или stdin. В большинстве случаев, это данные, которые поступают с клавиатуры.

Программы текстового режима передают данные для пользователей через стандартный вывод stdout.

Linux предоставляет второй тип потока вывода, известный как стандартная ошибка, или `stderr`.

Этот поток, предназначен для передачи информации, такой как сообщения об ошибках.

### Перенаправление потоков

Linux предоставляет специальные команды для перенаправления каждого потока. Эти команды записывают стандартный вывод в файл. Если вывод перенаправлен в несуществующий файл, команда создаст новый файл с таким именем и сохранит в него перенаправленный вывод.

Команды с одной угловой скобкой переписывают существующий контент целевого файла:

`>` — стандартный вывод

`<` — стандартный ввод

`2>` — стандартная ошибка

Команды с двойными угловыми скобками не переписывают содержимое целевого файла:

`>>` — стандартный вывод

`<<` — стандартный ввод

`2>>` — стандартная ошибка

Рассмотрим перенаправление потоков в файлы, устройства и другие потоки

```
rm example.txt 2> /dev/pts/1
```

Здесь мы использовали номер потока `stderr` (2). По умолчанию оператор `>` перенаправляет поток `stdout`, который имеет номер 1. Чтобы перенаправить другой поток, надо перед оператором `>` поставить его номер.

Можно создать канал между двумя процессами, в который один процесс сможет писать поток байтов, а другой процесс сможет его читать.

При помощи каналов организуются конвейеры оболочки. Когда оболочка видит строку вроде

```
команда1 | команда2
```

Канал использует `stdin`, `stdout` и `stderr`.

Можно создать канал между двумя процессами, в который один процесс сможет писать поток байтов, а другой процесс сможет его читать.

При помощи каналов организуются конвейеры оболочки. Когда оболочка видит строку вроде  
команда1 | команда2

## Сигналы

Сигналы являются способом передачи от одного процесса другому или от ядра операционной системы какому-либо процессу уведомления о возникновении определенного события. Сигналы можно рассматривать как простейшую форму межпроцессного взаимодействия.

Сигнал прерывает нормальный порядок выполнения инструкций в программе и передает управление специальной функции – обработчику сигнала.

**SIGHUP** (номер 1) изначально был предназначен для того, чтобы информировать программу о потере связи с управляющим терминалом (терминалы часто подключались к системе с помощью модемов, так что название сигнала происходит от hung up – повесить трубку). В ответ на получение SIGHUP демон обычно перезапускается (или просто повторно читает файл конфигурации).

Например, поменяли порт в /etc/ssh/sshd\_config- (`sed -i 's/22/222/' /etc/ssh/sshd_config`) сделали

```
ps aux | grep user1
```

```
kill -s HUP <PID>
```

или

```
kill -HUP <PID>
```

оно же

```
kill -1 <PID>
```

и не надо перезагрузки сервиса.

**SIGINT** (номер 2) обычно посылается процессу, если пользователь терминала дал команду прервать процесс (обычно эта команда – сочетание клавиш **Ctrl-C**).

**SIGKILL** (номер 9) завершает работу программы. Программа не может ни обработать, ни игнорировать этот сигнал.

**SIGTERM** (номер 15) вызывает «вежливое» завершение программы. Получив этот сигнал, программа может выполнить необходимые перед завершением операции (например, высвободить занятые ресурсы). Получение SIGTERM свидетельствует не об ошибке в программе, а о желании ОС или пользователя завершить ее.

**SIGCONT** (номер 18) возобновляет выполнение процесса, остановленного сигналом SIGSTOP

**SIGSTOP** (номер 19) приостанавливает выполнение процесса. Как и SIGKILL, этот сигнал не возможно перехватить или игнорировать.

**Пример:** работает юзер через терминал. Смотрим сеансы pts (ps ax | grep userX)

```
[netstat -apnt | grep ssh | grep '*']
```

```
Kill -STOP <PID>          ## остановка процесса
```

```
Kill -CONT <PID>         ## продолжить выполнение остановленного процесса
```

```
Kill -KILL <PID>
```

**SIGTSTP** (номер 20) приостанавливает процесс по команде пользователя (обычно эта команда – сочетание клавиш Ctrl-Z).

Комбинация клавиш	Сигнал
[Ctrl]c	Посылает стгнал SIGINT приоритетному процессу, т.е. программе, выполняющейся в данный момент. Это вызовет завершение программы, если в ней не предусмотрен перехват сигнала
[Ctrl]d	Выход из текущего терминала (обычно работает).
[Ctrl]z	Остановить текущий процесс. SIGTSTP

### Переменные окружения.

переменные окружения служат для организации работы операционной системы и приложений. Но при этом переменные могут использоваться и в скриптах по аналогии с переменными в других языках программирования

Окружение (environment) или среда - это набор пар ПЕРЕМЕННАЯ=ЗНАЧЕНИЕ, доступный каждому пользовательскому процессу.

**/etc/environment** - файл для создания, редактирования и удаления каких-либо переменных окружения на системном уровне.

Посмотреть окружение можно командой **env**

### Переменные оболочки

Кроме переменных окружения, командные оболочки располагают собственным набором пар ПЕРЕМЕННАЯ=ЗНАЧЕНИЕ. Они называются окружением (средой) оболочки

Задать переменную оболочки можно в командной строке, набрав ПЕРЕМЕННАЯ=ЗНАЧЕНИЕ

#### Пример:

```
LOG=/var/log
```

```
echo $LOG
```

Возможно включить локальную переменную оболочки в основное окружение. Для этого используется команда **export**:

#### Пример:

```
export LOG
```

```
env | grep LOG
```

Для просмотра переменных среды, используются команды **env** или **printenv**.

Дочерние процессы обычно наследуют переменные среды родительского процесса, что дает возможность менять значения или вносить дополнительные переменные для дочерних процессов.

**ЛАБОРАТОРНАЯ РАБОТА 6. УПРАВЛЕНИЕ СЕРВИСАМИ****Упражнение 1. Использование systemctl**

Получите информацию о текущем target. Для этого выполните команду

```
systemctl get-default
```

Выведите список всех загруженных юнитов

```
systemctl list-units --type target
```

Измените target по умолчанию

```
systemctl set-default multi-user.target
```

Проверьте результат:

```
systemctl get-default
```

Верните первоначальную настройку

```
systemctl set-default graphical.target
```

Переключите систему в multi-user.target

```
systemctl isolate multi-user.target
```

Переключите систему назад в graphical.target

```
systemctl isolate graphical.target
```

**Упражнение 2. Мониторинг процессов**

Отобразите состояние процессов в системе:

```
ps -aux
```

```
ps -auxf
```

Выполните мониторинг текущего состояния процессов:

```
top
```

Выведите информацию, каким процессорным ядром был обработан каждый процесс.

Для этого, не выходя из программы нажмите клавишу F, перейдите в меню на P и нажмите пробел.

Выйдите из меню кнопкой Esc

Выйдите из top – для этого нажмите клавишу q

**Упражнение 3. Использование сигналов**



Выведите список сигналов. Для этого выполните команду `kill -l`

Подключитесь по ssh(PuTTY) под пользователем `user1`

Вернитесь в терминал `root`'а и выполните `ps -aux | grep user1`

Скопируйте `pid`.

Выполните `kill -15 <pid>`

#### Упражнение 4. Определение переменных

Проверьте текущее значение переменной `LANG`

`set | grep LANG`

Отмените значение переменной в текущей сессии

`unset LANG`

`set | grep LANG`

Экспортируйте новое значение переменной в сессию:

`export LANG=ru_RU.UTF-8`

Проверьте результат

`date`

Обратите внимание на язык вывода команды `date`

Создайте переменную окружения `LOG`

`nano /etc/environment`

`LOG="/var/log"`

Переподключите сессию и выполните:

`echo $LOG`

`cd $LOG`

**ГЛАВА 7. ВВЕДЕНИЕ В ИСПОЛЬЗОВАНИЕ СЦЕНАРИЕВ ОБОЛОЧКИ****Сценарии shell**

В shell-скриптах последовательность `#!` должна стоять самой первой и задает интерпретатор (`sh` или `bash`). Интерпретатор, в свою очередь, воспринимает эту строку как комментарий, поскольку она начинается с символа `#`.

```
#!/bin/sh
```

```
#!/bin/bash
```

```
#!/usr/bin/perl
```

```
#!/usr/bin/tcl
```

```
#!/bin/sed -f
```

```
#!/usr/awk -f
```

Например, следующий сценарий `helloworld` просто выполняет команду `echo`.

```
#!/bin/bash
```

```
echo "Hello, world!"
```

При принятии решения о том, как выполнить этот файл, ядро отыщет соответствующий синтаксис. С точки зрения оболочки, "стремящейся" выполнить этот сценарий, первая строка представляет собой просто комментарий.

Для того чтобы подготовить этот файл к выполнению, достаточно установить его бит, "отвечающий" за выполнение.

```
$ chmod +x helloworld
```

```
$ ./helloworld
```

Или:

```
$ bash helloworld
```

```
$ source helloworld
```

**Лабораторная работа 7. Введение в использование сценариев****Упражнение 1. Организация ввода и вывода данных**

Создайте файл сценария

```
touch myscript.sh
```

Сделайте его исполняемым

```
chmod +x myscript.sh
```

чтобы сформировать для пользователя приглашение ввести данные, используйте команду read.

```
nano myscript.sh
```

Введите в файл следующие строки:

```
#!/bin/bash
```

```
echo -n "Enter Name: "
```

```
read p
```

```
echo Hello "$p !!!"
```

Запустите скрипт на исполнение:

```
./myscript.sh
```

## Упражнение 2. Конструкция If-Then-Else:

Перезапишите файл командой:

```
cat > myscript.sh
```

вставив нижеследующее содержимое:

```
#!/bin/bash
```

```
if
```

```
test -d test
```

```
then
```

```
    echo "Directory exists"
```

```
else
```

```
    mkdir test
```

```
fi
```

Запустите скрипт на исполнение:

```
./myscript.sh
```

**Упражнение 3. Конструкция CASE:**

Перезапишите файл командой:

```
cat > myscript.sh
```

вставив нижеследующее содержимое:

```
#!/bin/bash
echo "Please, select editor:"
echo "1 nano"
echo "2 vim"
echo "3 sed"
echo "4 Exit"
read selector
case $selector in
1)
/bin/nano
;;
2)
/usr/bin/vim
;;
3)
/bin/ed
;;
4)
exit 0
;;
*)
```

```
echo "incorrect action"
```

```
esac
```

Запустите скрипт на исполнение:

```
./myscript.sh
```

#### Упражнение 4. Цикл While

Перезапишите файл командой:

```
cat > myscript.sh
```

вставив нижеследующее содержимое:

```
# !/bin/bash
```

```
x=1
```

```
while [ $x -lt 5 ]
```

```
do
```

```
mkdir test$x
```

```
echo "Directory created: test$x"
```

```
x=$(( $x + 1 ))
```

```
done
```

Запустите скрипт на исполнение:

```
./myscript.sh
```

**ГЛАВА 8. СИСТЕМА БЕЗОПАСНОСТИ LINUX****Управление пользователями**

Локальные учетные записи пользователей хранятся в

```
cat /etc/passwd
```

Пароли:

```
cat /etc/shadow
```

пароль (хэш-пароля) и параметры (устаревание смена блокировка)

Чтобы увидеть к каким группам вы принадлежите используйте команду *groups*:

```
$ groups
```

Смена первичной группы пользователя

*-g* — меняет первичную группу указанного пользователя

*newgrp группа* — временно меняет первичную группу пользователя в пределах нового открываемого этой командой сеанса.

Добавить в группу

```
usermod -G группа пользователь
```

Добавить в группу множество пользователей

```
root@server:~# gpasswd -M user1,user2 test
```

Удалить из группы

```
usermod -G '' userX
```

или

Удалить пользователя из указанной группы

```
gpasswd -d, --delete пользователь группа
```

**Сервисные учетные записи (не для интерактивного входа)**

Использование *nologin* вместо *shell* заменой шелла на */sbin/nologin*

**Управление учетными записями**

Для временных пользователей можно использовать устаревание учетных записей:

```
$ sudo chage -E 2014-09-11 user2      ##(-1— unlock)
```

Для принуждение к смене пароля при следующем входе:

```
$ sudo chage -d 0 user2
```

### **PAM (Pluggable Authentication Modules)**

PAM выполнен в виде разделяемых библиотек-модулей комфортно расположившихся в каталоге `/lib/x86_64-linux-gnu/security`. В CentOS7ls `/usr/lib64/security/`

Одной из наиболее распространенных конфигураций PAM является отказ в доступе для входа в систему после определенного количества неудачных попыток. Это делается с помощью модуля `pam_tally2`. Например, чтобы запретить вход через `ssh` после трех неудачных попыток входа, отредактировав файл `/etc/pam.d/sshd` так, чтобы запретить вход в систему после трех неудачных попыток:

```
auth required pam_tally2.so deny=3 onerr=fail
```

```
account required pam_tally2.so
```

### **Делегирование административных полномочий.**

Файл конфигурации `sudo`

```
sudo less /etc/sudoers
```

```
sudo visudo
```

Редактирование `sudoers` (`visudo`)

`visudo` позволяет использовать любой редактор. Например, `vim`:

```
EDITOR=vim visudo
```

Или

```
# ls -l /etc/alternatives/editor
```

После файла `/etc/sudoers` анализируются файлы в директории `/etc/sudoers.d`

Для делегирования административных полномочий рекомендуется создавать файл в этой директории.

### **Режимы доступа в Linux и модель владения (ownership).**

Каждый файл принадлежит одному пользователю и одной группе. Это модель режима доступа в Linux.

Определить, какому пользователю и группе принадлежит файл можно с помощью команды `ls -l`

Первое поле `-rwxr-xr-x` содержит символическое представление прав на данный файл. Первый знак (-) в этом поле определяет тип файла

Остальная часть поля состоит из трех троек символов. Первая тройка представляет права владельца файла, вторая представляет права группы файла и третья права всех остальных пользователей.

### Изменение пользователя и группы владельца

Чтобы изменить владельца или группу файла (или другого объекта) используется команды *chown* или *chgrp* соответственно. Сначала нужно передать имя группы или владельца, а потом список файлов.

Чтобы использовать команду *chown* нужны права суперпользователя.

*chgrp* может быть использована всеми, чтобы изменить группу-владельца файла на группу, к которой они принадлежат.

```
# chown root /etc/passwd
# chgrp wheel /etc/passwd
```

### Права доступа к папке

Для директорий право на исполнение позволяет сделать данную директорию текущей, например, перейти в неё командой "cd".

Также для получения подробной информации о файлах, находящихся в каталоге нужно иметь доступ на исполнение папки

*chmod* - программа для изменения прав доступа к файлам и директориям. Меняет маску доступа к файлу.

```
chmod 666 file1
```

Другой способ назначения прав - это символическое представление.

u - владельцу объекта;

g - группе объекта;

o - пользователю «все остальные»;

a - все вышеперечисленное.

### Идентификатор пользователя (UID) и текущий идентификатор пользователя (EUID)

UID (User ID) — это идентификатор пользователя, создавшего данный процесс.

EUID (Effective User ID) — это текущий пользовательский идентификатор процесса.

SUID-программы всегда выполняются с правами владельца программы.

**setuid** и **setgid** являются флагами прав доступа, которые разрешают пользователям запускать исполняемые файлы с правами владельца или группы владельца исполняемого файла

Добавить *setuid*: ***chmod u+s file1***



- восьмеричные значения для SUID и SGID - 4000 и 2000.
- Символьные: u+s и g+s.

### POSIX ACL

```
yum -y install acl
```

```
setfacl -m u:user1:rw file.txt
```

- назначает пользователю user1 права на чтение и запись.

```
setfacl -m g:group1:r file.txt
```

- назначает группе group1 права на чтение

Просмотр ACL:

```
getfacl file.txt
```

## **ЛАБОРАТОРНАЯ РАБОТА 8. СИСТЕМА БЕЗОПАСНОСТИ LINUX**

### **Упражнение 1. Работа с пользовательскими учетными записями**

#### **Добавление учетной записи пользователя**

`useradd user2`

Удалите пользователя с каталогом и открытым сеансом:

`userdel user2 -rf`

Добавьте пользователей:

`useradd user2`

`passwd user2`

и

`useradd backupuser`

`passwd backupuser`

111

#### **Изменить параметры пользователя**

`usermod user2 -s 'Ivanov Ivan Ivanovich,239,45-67,499-239-45-33'`

Задайте пароль:

`passwd user2`

Посмотрите информацию о пользователе:

`id user2`

или

`cat /etc/passwd | grep user`

Создайте группы

`groupadd -g 1500 mygroup1`

`groupadd -g 1501 mygroup2`

Проверьте:

`cat /etc/group | grep mygroup`

Добавьте user2 в группы

```
usermod -aG mygroup1,mygroup2 user2
```

Удалите его из групп

```
usermod -G '' userX
```

Чтобы увидеть к каким группам вы принадлежите используйте команду *groups*:

```
$ groups
```

Добавьте в группу mygroup1 множество пользователей

```
root@server:~# gpasswd -M user1,user2 test
```

Удалите user1 из этой группы

```
gpasswd -d user1 mygroup1
```

## Упражнение 2. Блокировка учетных записей:

### 1. nologin shell

Выполните замену шелла на /sbin/nologin:

```
# vim /etc/passwd
```

```
user2:x:1002:1002:Petrov Petr:/home/petrov:/sbin/nologin
```

Попробуйте подключиться по ssh как user2

Верните шелл /bin/bash

Попробуйте подключиться по ssh снова

Отключите учетную запись:

```
$ sudo usermod -L user2      ##(-U – unlock)
```

Заставьте пользователя user2 сменить пароль при следующем входе:

```
$ sudo chage -d 0 user2
```

## Упражнение 3. Настройте использование ключей вместо паролей для аутентификации ssh подключения

Зайдите как backupuser

Генерируем ключевые пары

```
backupuser@server:~$ ssh-keygen
```

Копируем ключ на сервер:

```
$ ssh-copy-id server
```

\$ ssh server – подключение без запроса ввода пароля.

### Упражнение 3. PAM (Pluggable Authentication Modules)

Настройте запретит входа через ssh после трех неудачных попыток.

Для этого отредактируйте файл /etc/pam.d/sshd добавив в его конец строки:

```
auth required pam_tally2.so deny=3 onerr=fail
```

```
account required pam_tally2.so
```

Проверьте результат – попытайтесь войти с неверным паролем более 3 раз.

### Упражнение 3. Настройка sudo

Настройте разрешение использовать sudo пользователю **backupuser** для выполнения резервного копирования утилитой tar:

В директории /etc/sudoers.d/ создайте файл с именем пользователя

Прим. Файлы в /etc/sudoers.d / должны иметь доступ 0440 и принадлежать root:root.

Выполните:

```
cat > /etc/sudoers.d/backupuser
```

```
backupuser    ALL= NOPASSWD: /usr/bin/tar
```

Проверьте под backupuser:

```
sudo tar -C / -czf etc`date '+%Y%m%d'`.tar.gz etc/
```

```
ls -l ~
```

### Упражнение 4. Разрешения в Linux/UNIX

```
touch myfile1
```

**ls -l**

**chmod 666 myfile1**

**ls -l**

### **POSIX ACL**

Назначьте пользователю user1 и mygroup1 разрешения чтения и записи на myfile1.

**setfacl -m u:user1:rw file.txt**

Назначьте группе mygroup1 права на чтение

**setfacl -m g:mygroup1:r file.txt**

Просмотр ACL:

**getfacl file.txt**

**ГЛАВА 9. УПРАВЛЕНИЕ ПО****Управление ПО.*****Утилита ldd - print shared library dependencies******# ldd /bin/bash***

Зависимости:

**RPM****В какой пакет входит файл**

# rpm -q bash – узнать версию пакета

rpm -qi bash – информация о пакете

rpm -qf /bin/bash – узнать, из какого пакета был взят этот файл

rpm -ql bash - Какие файлы были установлены этим пакетом

rpm -qa - Список всех установленных пакетов в системе.

Опции:

-i – установка пакета

-U – обновить пакет

-e удалить пакет

-e --test для проверки возможности безопасного удаления пакета

Для анализа и установки/удаления пакета и зависимостей используется Yum (Advanced Packaging Tool).

**Yellowdog Updater, Modified (YUM) —менеджер пакетов для дистрибутивов, использующих пакеты формата RPM.**

Конфиг файлы в /etc/yum.repos.d имеют расширение .repo

EPEL или Extra Packages for Enterprise Linux - это репозиторий дополнительных пакетов для Red Hat, поддерживаемый командой проекта Fedora. Здесь есть новые версии программ, а также некоторые другие программы, которых нет в официальных репозиториях. Чтобы добавить репозиторий CentOS 7 выполните:

yum -y install epel-release

yum repolist

yum кэширует информацию и базы данных для ускорения производительности. Чтобы удалить часть или всю кэшированную информацию, можно выполнить команду:

`yum clean [ packages | metadata | expire-cache | rpmdb | plugins | all ]`

### **yum install bash-completion**

Основные команды yum:

- `yum list [installed | updates | available]` - список установленных | доступных пакетов
- `yum search` - поиск пакетов по имени
- `yum info` - показать подробную информацию о пакете

`yum verify [package]` - проверка пакета, также: `yum verify-rpm [package]` эквивалентно `rpm -V`

- `update` - обновить списки доступных пакетов
- `install` - установить пакет

`yum localinstall package-file` установить пакет из локального rpm-файла

- `remove` - удалить пакет
- `update` - установить доступные новые версии пакетов
- `yum repolist` – список репозиторий

`yum history` – история команд yum

Чтобы узнать в какой пакет входит требуемая утилита, например:

```
[root@server ~]# semanage
```

```
-bash: semanage: command not found
```

Можно воспользоваться

```
# yum whatprovides /usr/sbin/semanage
```

Или

```
# yum provides /usr/sbin/semanage
```

```
yum -y install htop
```

### **Установка группы пакетов на примере настройки GUI**

Установка X сервера, менеджера дисплеев, оконного менеджера и клиентских программ

```
# yum groupinstall "X Window System" "Fonts" -y
```

```
# yum grouplist | grep KDE
```

```
# yum groupinstall "KDE Plasma Workspaces"
```

**ЛАБОРАТОРНАЯ РАБОТА 9. УПРАВЛЕНИЕ ПО****Упражнение 1. Сборка и установка пакетов из исходников**

Произведите установку всех необходимых пакетов:

```
sudo yum install gcc ncurses-devel make rpm-build rpmdevtools wget -y
```

Утилита checkinstall отслеживает действия установочного скрипта и из них создает установочные инструкции для сборки пакета.

Загрузите и установите пакет checkinstall:

```
wget --no-check-certificate https://filebox.ece.vt.edu/~mclint/puppet/files/checkinstall-1.6.2-3.el6.1.x86\_64.rpm
```

```
sudo rpm -i checkinstall-1.6.2-3.el6.1.x86_64.rpm
```

Создайте директорию rpmbuild/SOURCES в домашнем каталоге

```
cd
```

Для примера рассмотрим сборку и установку консольного браузера lynx. Исходники можно найти на сайте проекта:

<http://lynx.invisible-island.net/lynx.html>

Для сборки из исходников перейдите в директорию /usr/src/:

```
cd /usr/src/
```

если это не первая сборка, во избежании артефактов выполните очистку каталога от всех файлов полученных в результате предыдущей компиляции.

```
make clean
```

Загрузите исходники с сайта проекта:

```
wget https://invisible-mirror.net/archives/lynx/tarballs/lynx2.8.9rel.1.tar.gz
```

Распакуйте архив

```
tar -xvf lynx2.8.9rel.1.tar.gz
```

Перейдите в директорию с распакованными файлами:

```
cd lynx2.8.9rel.1/
```

запустите скрипт ./configure который создаст Makefile:



./configure

Запустите процесс сборки:

make

Теперь соберите RPM пакет без его инсталляции:

checkinstall --install=no

Проверьте появление пакета в указанной директории:

ls rpmbuild/RPMS/x86\_64/

Установите собранный пакет:

sudo rpm -i ~/rpmbuild/RPMS/x86\_64/lynx2-8-8-1.x86\_64.rpm

Проверьте результат:

lynx <http://www.google.com>

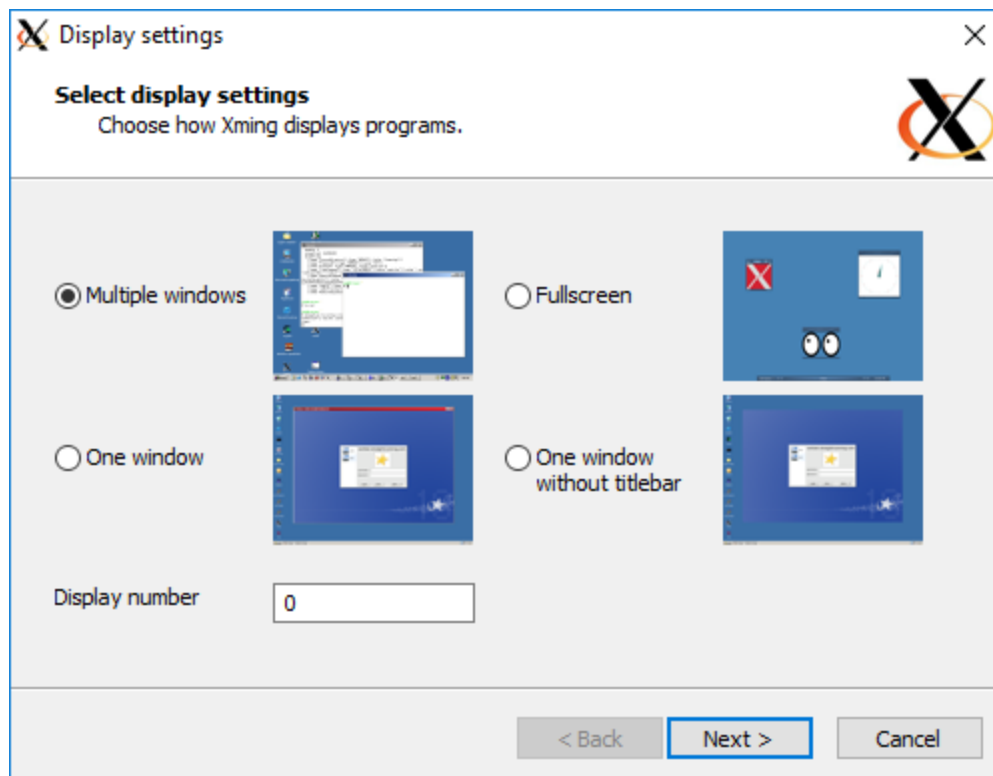
Удалите инсталляцию:

sudo rpm -e lynx2-8-8-1

## Упражнение 2. Удаленное подключения к графической подсистеме Linux

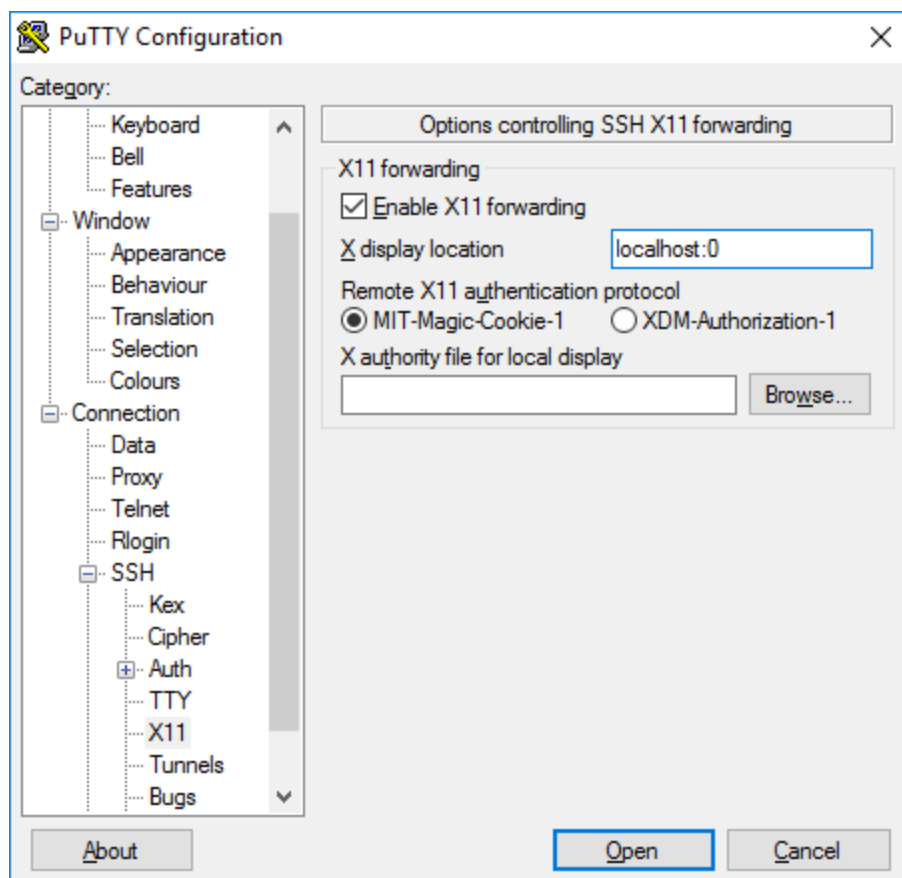
На хост Windows ставим Xming <http://www.straightrunning.com/XmingNotes/changes.php>

<https://sourceforge.net/projects/xming/>



И далее по умолчанию

Настраиваем putty:



Проверяем наличие указанных параметров настройки в файле:

```
sudo nano /etc/ssh/sshd_config
```

```
X11Forwarding yes
```

```
X11DisplayOffset 10
```

```
X11UseLocalhost yes
```

Применяем настройки:

```
systemctl restart sshd.service
```

Запускаем графическое приложение firefox для перенаправления:

```
firefox &
```

**ГЛАВА 10. РЕЗЕРВНОЕ КОПИРОВАНИЕ И ВОССТАНОВЛЕНИЕ****Резервное копирование и восстановление****Утилиты для резервного копирования**

Для выполнения этих задач использовать средства самих приложений, если предусмотрены. Особенно для СУБД (mysqldump), служб каталогов и т.д.

**Tape archiver – tar. Архивирует каталоги.**

#cd / - переходим в родительский относительно архивируемого каталог (чтоб при восстановлении не перезаписать по абс пути данные)

-c #create (создать)

-t #type (просмотреть)

-x #extract (распаковать)

-j #сжимать

-f (файл архива)

Далее указываются каталоги для архивирования.

-v #(verbose)

# sudo tar-cvf etc.tar etc/

# sudo tar -xf /etc.tar -в текущий каталог

**Планирование заданий**

**Для задач по расписанию в linux для каждого пользователя доступен crontab**

crontab -e

отрывается редактор:

0 \*/1 \* \* \* <задача для выполнения раз в час>

раз в минуту, тогда запись будет выглядеть так:

\*/1 \* \* \* \* xxxxx

Для автоматизации административных задач используется системный crontab. Его конфигурационный файл /etc/crontab

Также возможно использовать для целей резервного копирования **rsync**

```
rsync -avbz test /var/backups/test1
```

```
rsync -avzb test root@server:/var/backups/test2
```

В частности по расписанию:

**Пример:**

```
crontab -e
```

```
*/1 * * * * rsync -avzhe ssh test root@192.168.51.1:/var/backups/
```

-a — Режим архивирования

-v — Выводить подробную информацию о процессе копирования

-z — сжимать файлы перед передачей

-e — использовать другой транспорт

-b — создание резервной копии:

## Лабораторная работа 10. Создание автоматизированной задачи резервного копирования

### Упражнение 1. Создание сценария задачи резервного копирования для регулярного выполнения

Создайте файл сценария резервного копирования каталога `/etc/` в указанной директории:

```
cat > /etc/cron.hourly/backup_conf.sh
```

Введите следующие строки:

```
#!/bin/sh
```

```
echo Backup conf
```

```
CMD="sudo /usr/bin/tar"
```

```
RCMD="ssh backupuser@server"
```

```
DIRS="/etc/"
```

```
cd /; $CMD -czf - $DIRS | $RCMD "cat > `hostname`.backup_conf.`date '+%Y%m%d'`.tbz"
```

Нажмите Ctrl + d

Или

```
#!/bin/bash
```

```
tar -C / -czf - etc | ssh root@localhost cat > `hostname`.tbz
```

Или оно же:

```
tar -C / -czf --login etc | ssh root@localhost cat > `hostname`.conf.`date '+%Y%m%d'`.tbz
```

Рекомендуется всегда использовать опцию --login (вместо ее ярлыка '--' чтобы избежать побочных эффектов, вызванных смешиванием сред

Сделайте сценарий исполняемым:

```
chmod +x /etc/cron.hourly/backup_conf.sh
```

Настройте расписание в системном crontab (для демо примера на ежеминутное выполнение):

```
# cat >> /etc/crontab
```

```
*/1 * * * * backupuser run-parts /etc/cron.hourly
```

Проверьте через минуту наличие файла бэкапа в домашнем каталоге пользователя backupuser:

```
ls -l ~backupuser
```

Упражнение 2. Использование Rsync для резервного копирования:

Создайте каталоги – исходный и для резервной копии:

```
sudo -i
```

```
mkdir test
```

```
mkdir /var/backups/
```

и файл в исходном каталоге:

```
cat > test/file1
```

```
11111
```

Нажмите Ctrl + d

Выполните синхронизацию:

```
rsync -avz test /var/backups/
```

Проверьте наличие резервной копии:

```
ls /var/backups/
```

```
ls /var/backups/test/
```

```
less /var/backups/test/file1
```

```
cat >> test/file1
```

```
22222
```

```
less /var/backups/test/file1
```

Также возможно выполнять процедуру по сети:

```
rsync -avze ssh test root@server:/var/backups/
```

```
less /var/backups/test/file1
```

## ГЛАВА 11. АНАЛИЗ ПРОИЗВОДИТЕЛЬНОСТИ И ОПТИМИЗАЦИЯ СИСТЕМЫ

Одним из основных средств мониторинга производительности является утилита `vmstat`.

`vmstat` предоставляет информацию о различных ресурсах системы и связанных с ними неполадках, приводящих к снижению производительности.

Вывод команды **`vmstat`** содержит информацию о нитях ядра, находящихся в очереди выполнения и ожидающих запуска, а также об оперативной памяти, подкачке, дисках, прерываниях, системных вызовах, переключении контекста и работе процессоров

### CPU

Когда в системе наблюдается большое количество процессов в состоянии исполнения (`r`), это может означать, что необходимо или увеличить количество процессоров или заменить процессор на более производительный, так как процессы постоянно ждут, пока освободится процессор. Допустимая длина очереди ожидающих процессов пропорциональна количеству процессоров в системе

### RAM

Если в системе интенсивно используется области подкачки в большом объеме, при этом мало свободной памяти, а также имеется постоянная активность по перемещению страниц в область подкачки. Эти обстоятельства говорят о недостатке памяти.

В случаях, когда исчерпывается вся виртуальная память, запускается особый механизм (the Out of Memory Killer - OOM Killer). Он выбирает процесс, который будет уничтожен для того, чтобы освободить для системы часть физической памяти.

### Настройка системы ввода-вывода

Наиболее часто встречающаяся проблема – появление узких мест в подсистеме ввода-вывода

Ввод/вывод может быть разбит на две больших категории: дисковый I/O и сетевой I/O.

Обычно признаком наличия проблем с вводом-выводом является большое количество процессов в состоянии D (uninterruptible sleep), в которое процесс переходит при ожидании ввода-вывода.

За распределение дисковых операций по процессам отвечает планировщик подсистемы ввода и вывода. Определить, какой планировщик задействован, можно посмотрев файл `/sys/block/sda/queue/scheduler`:

```
cat /sys/block/sda/queue/scheduler
```

Команда **iostat** позволяет быстро обнаружить неполадки дискового ввода-вывода, приводящие к снижению производительности

Пакет **ktune** для оптимизации производительности высоконагруженного сервера доступен в репозитории CentOS.

Пакет предлагает набор настроек для оптимизации работы ядра на серверах с большим размером ОЗУ работающих в условиях повышенной сетевой и дисковой нагрузки.

### ЛАБОРАТОРНАЯ РАБОТА 11. АНАЛИЗ ПРОИЗВОДИТЕЛЬНОСТИ

#### Упражнение 1. Использование утилит мониторинга

Запустите утилиту **vmstat** для проведения 5 замеров с интервалом 5 секунд:

```
# vmstat 5 5
```

Запустите утилиту **iostat**

```
# iostat -t 2 6
```

Ознакомьтесь с параметрами вывода команд

Колонка	Описание



tps	transfers per second — активность I/O операций в секунду, несколько логических запросов могут быть объединены в один;
Blk_read/s	количество запросов на чтение в секунду, выраженное в блоках (512 bytes);
Blk_wrtn/s	количество запросов на запись в секунду, выраженное в блоках (512 bytes);
Blk_read	общее количество прочитанных блоков;
Blk_wrtn	общее количество записанных блоков;

`iostat -xk -t 10`

параметр -t 10. Это интервал, за который усредняются значения и вычисляются «средние» в секундах.

<https://habr.com/ru/post/165855/>

## Упражнение 2. Определение планировщика подсистемы ввода-вывода

Посмотрите содержимое файла `/sys/block/sda/queue/scheduler`

```
cat /sys/block/sda/queue/scheduler
```

Какой планировщик используется в Вашей системе?

## Упражнение 3. Вызов OOM Killer

Отключите подкачку:

```
$ sudo /sbin/swapoff -a
```

Установите и запустите утилиту stress которая вызовет рост потребления памяти:

```
$ sudo yum -y install stress
```

```
$ stress -m 80 -t 10s
```

Вы можете увидеть результат работы OOM Killer, запустив `dmesg` или посмотреть журналы `/var/log/messages` или `/var/log/syslog`.

УД "Специалист"

## **ГЛАВА 12. ИСПОЛЬЗОВАНИЕ ВЕБ КОНСОЛИ COCKPIT ДЛЯ УДАЛЕННОГО АДМИНИСТРИРОВАНИЯ**

Cockpit-это удобный веб-интерфейс для администрирования серверов. Позволяет легко контролировать системные ресурсы и управлять конфигурацией.

Кроме Centos 8 может быть установлен и на другие дистрибутивы: <https://computingforgeeks.com/how-to-install-cockpit-on-ubuntu-18-04-debian-9/>

Cockpit является модульным и может быть расширен за счет установки дополнительных модулей. Есть возможность разрабатывать собственные модули.

В Centos 8 Cockpit встроен в систему. Для использования его нужно активировать посредством `systemctl enable --now cockpit.socket`

Для доступа к администрированию системы по сети в веб-браузере введите url или ip адрес сервера и укажите порт 9090.

Например, 172.16.1.10:9090

После того, как вы вошли в систему, вы увидите основной интерфейс Cockpit. Он имеет вкладку приборной панели в верхней части и боковое меню с подробной информацией для выбранной системы слева. На панели мониторинга отображается список всех систем, добавленных на сервер Cockpit, с графиками использования процессора, памяти, дискового ввода-вывода и сетевого трафика.

Из панели мониторинга вы можете выбрать имя системы и посмотреть на боковое меню:

System: показывает информацию о системе, на которой работает Cockpit, такую как использование процессора, памяти, дискового ввода-вывода и сетевого трафика, а также сведения об оборудовании и операционной системе.

Logs: Просмотр сообщений, созданных журналом systemd, включая ошибки, предупреждения и уведомления. Журнал похож на выходные данные команды journalctl. В журнале сначала отображаются самые новые записи с возможностью фильтрации по типу.

Networking: Позволяет увидеть сетевые интерфейсы (например, eth0) и активные графики отправленных и полученных данных.

Учетные записи: показывает, какие административные (root) и другие пользователи имеют учетные записи в системе.

Службы: показывает службы systemd, запущенные на сервере Cockpit. Вы можете видеть, какие из них активны/включены или неактивны. Вы также можете увидеть другие функции systemd: цели, сокеты, таймеры и пути.

Диагностические отчеты: Cockpit собирает информацию о конфигурации системы и диагностике и готовит отчет в сжатом формате xz.

Конфигурация дампа ядра: показывает состояние и конфигурацию kdump и позволяет аварийно завершить работу ядра для тестирования kdump.

SELinux: показывает, включен ли SELinux, и показывает ошибки управления доступом.

УЦ "Специалист"

**ЛАБОРАТОРНАЯ РАБОТА 12. АКТИВАЦИЯ СОСКРИТ И ПОДКЛЮЧЕНИЕ.**

Активируйте Cockpit:

```
sudo systemctl enable --now cockpit.socket
```

Разрешите подключения к порту 9090 через брандмауэр:

```
sudo firewall-cmd --permanent --add-port=9090/tcp
```

запустите сервис:

```
sudo systemctl start cockpit.socket
```

Подключитесь со своей хостовой системы:

<http://172.16.1.X:9090>

Войдите в Cockpit.