

# **Лабораторная работа №1**

**Дисциплина: Основы информационной безопасности**

Дарижапов Тимур Андреевич

# Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Выводы	23

# List of Figures

3.1	Официальный сайт VirtualBox . . . . .	7
3.2	Создание виртуальной машины . . . . .	8
3.3	Память и процессоры . . . . .	8
3.4	Основная память . . . . .	9
3.5	Выбор языка . . . . .	9
3.6	Отключение KDUMP . . . . .	10
3.7	Настройка интернета . . . . .	10
3.8	Выбор программ . . . . .	11
3.9	Пароль . . . . .	11
3.10	Место установки . . . . .	12
3.11	Перезагрузка . . . . .	12
3.12	Rocky Linux . . . . .	13
3.13	dmesg . . . . .	13
3.14	Версия ядра Linux . . . . .	14
3.15	Частота процессора . . . . .	14
3.16	Модель процессора (CPU0) . . . . .	14
3.17	Объем доступной оперативной памяти . . . . .	14
3.18	Объем доступной оперативной памяти . . . . .	14
3.19	Объем доступной оперативной памяти . . . . .	15
3.20	Знакомство с github . . . . .	16
3.21	Настройка . . . . .	16
3.22	Ключ . . . . .	16
3.23	Вывод ключа в терминале . . . . .	17
3.24	Ключ . . . . .	17
3.25	Рабочее пространство . . . . .	18
3.26	Коммит . . . . .	18
3.27	pandoc . . . . .	22
3.28	Установка pandoc . . . . .	22

## List of Tables

# 1 Цель работы

Часть 1: установка на виртуальную машину системы Rocky Linux и настройка необходимых сервисов. Часть 2: освоение git и применение средств контроля версий. Часть 3: оформление отчётов с помощью языка разметки Markdown.

## 2 Задание

Часть 1 1.Установка и настройка Виртуальной машины VirtualBox (ОС Linux)

Часть 2 1.Создать базовую конфигурацию для работы с git. 2.Создать ключ SSH.  
3.Создать ключ PGP. 4.Настроить подписи git. 5.Зарегистрироваться на GitHub.  
6.Создать локальный каталог для выполнения заданий по предмету.

Часть 3 1.Сделайте отчёт по предыдущей лабораторной работе в формате Markdown. 2.В качестве отчёта предоставить отчёты в 3 форматах: pdf,docx и md

## 3 Выполнение лабораторной работы

Часть 1. Установка и настройка Виртуальной машины VirtualBox

1) Переходим на официальный сайт VirtualBox и скачиваем файл. Устанавливаем VirtualBox на компьютер. Я часто пользуюсь Linux, поэтому VirtualBox уже был установлен на компьютере.



Figure 3.1: Официальный сайт VirtualBox

2) Создаём виртуальную машину. Для этого нажимаем на “Создать”. Имя нашей виртуальной машины - tadarizharov. Выбираем наш образ диска Rocky Linux. Чтобы он распознал его как образ, предварительно скачиваем UltraISO. Автоматически выбирается Linux версии RedHat.

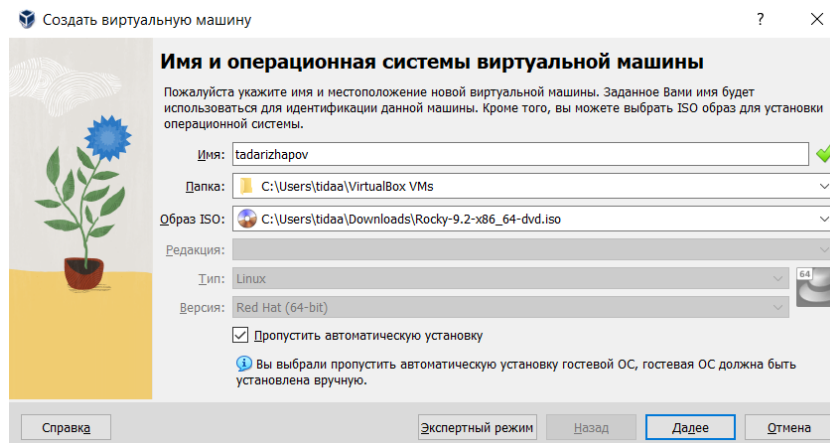


Figure 3.2: Создание виртуальной машины

3) Виртуальной машине требуется ОЗУ. Выделяем 4096 МБ, что является половиной основного ОЗУ. Также выделяем ей 4 ЦП.

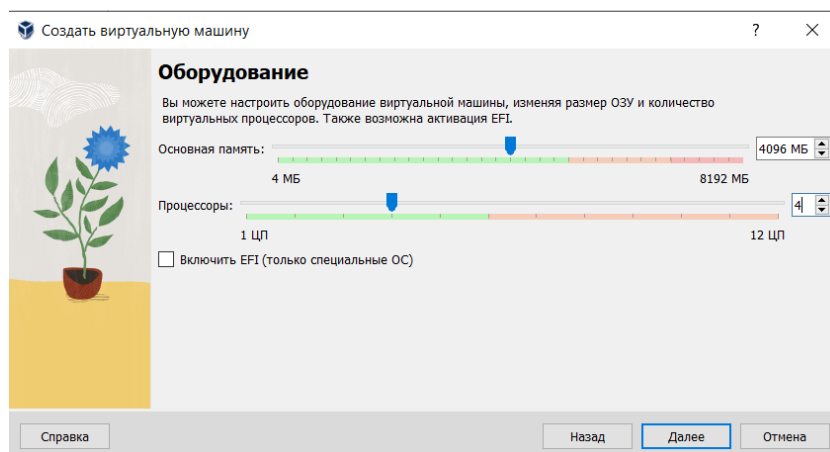


Figure 3.3: Память и процессоры

4) Выделяем 40 ГБ памяти на виртуальную машину, думаю, этого достаточно.



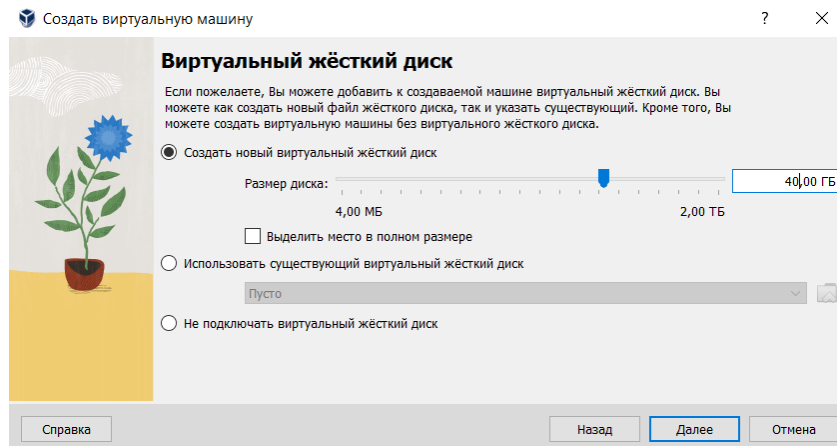


Figure 3.4: Основная память

5) Включаем виртуальную машину, нас встречает выбор языка.

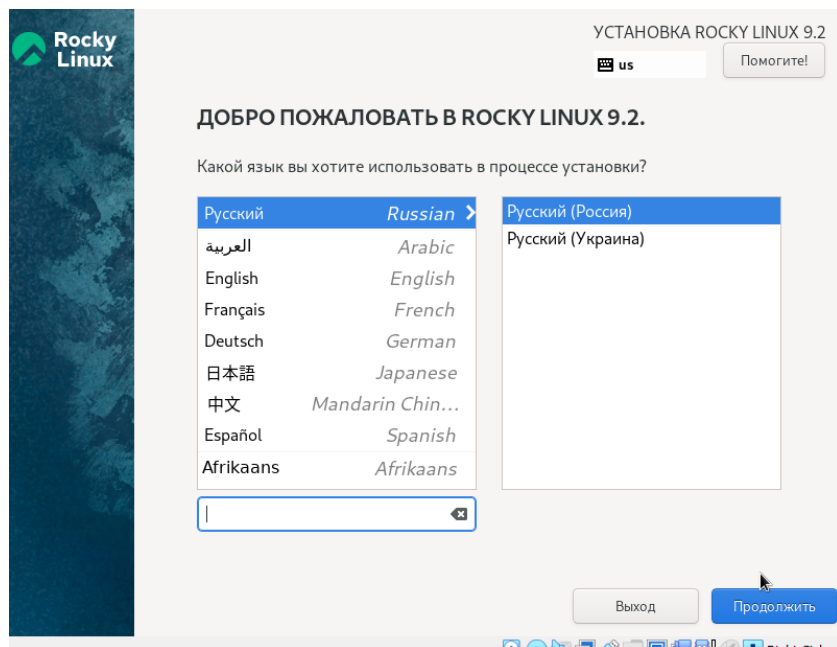


Figure 3.5: Выбор языка

6) Отключаем KDUMP.

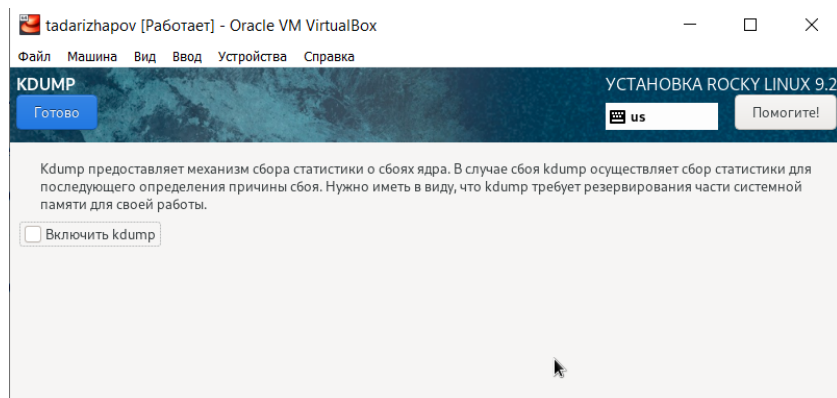


Figure 3.6: Отключение KDUMP

## 7) Настраиваем сеть и имя узла

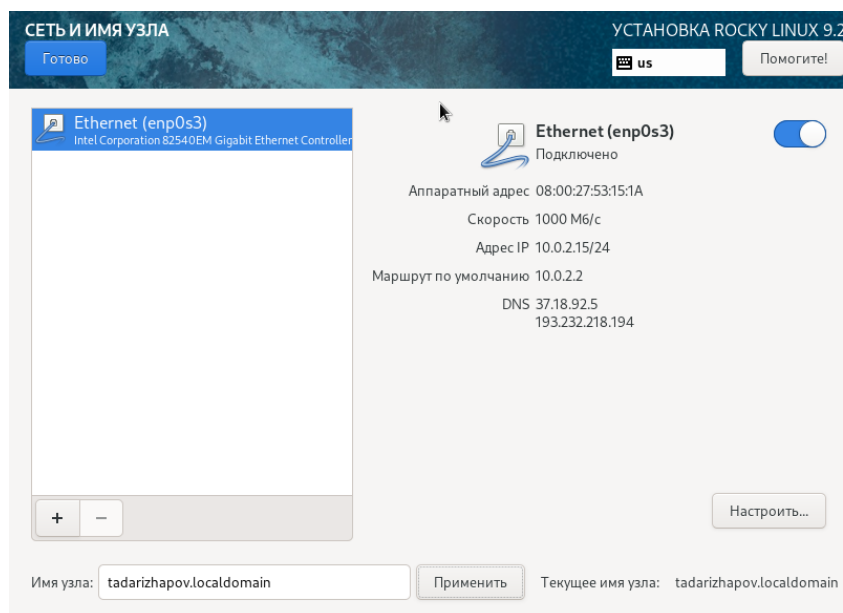


Figure 3.7: Настройка интернета

8) Базовое окружение выбираем Сервер с GUI. В дополнительном окружении выбираем средства разработки.

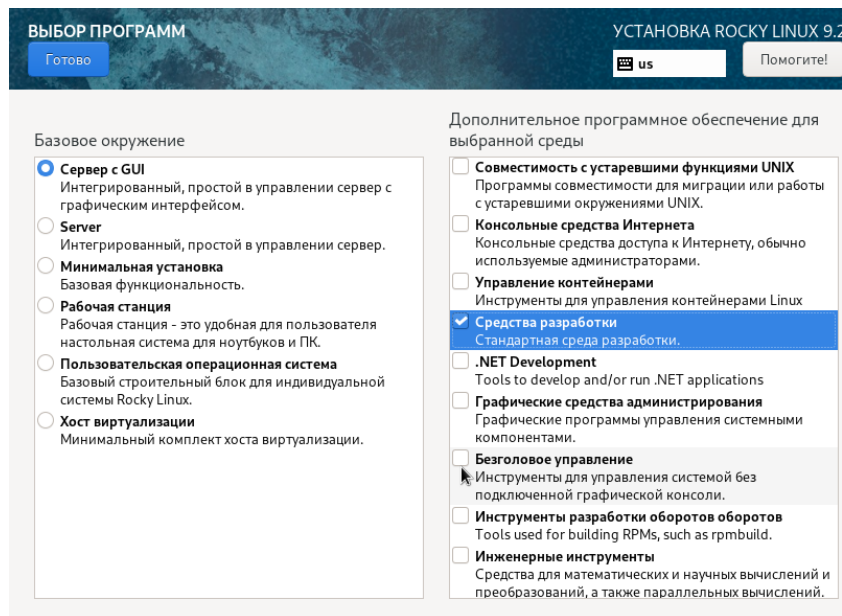


Figure 3.8: Выбор программ

9) Устанавливаем пароль для root пользователя.

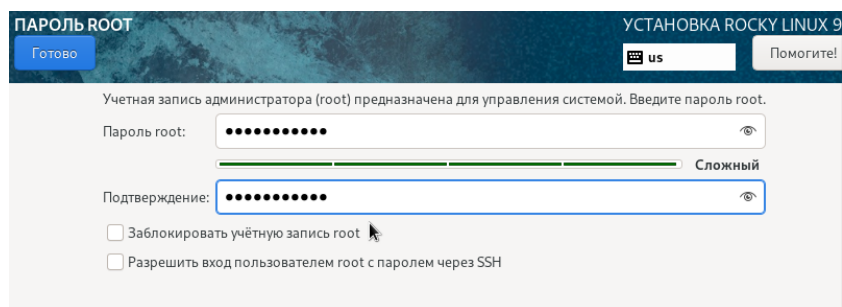


Figure 3.9: Пароль

10) В выборе места установки ставим наш диск.

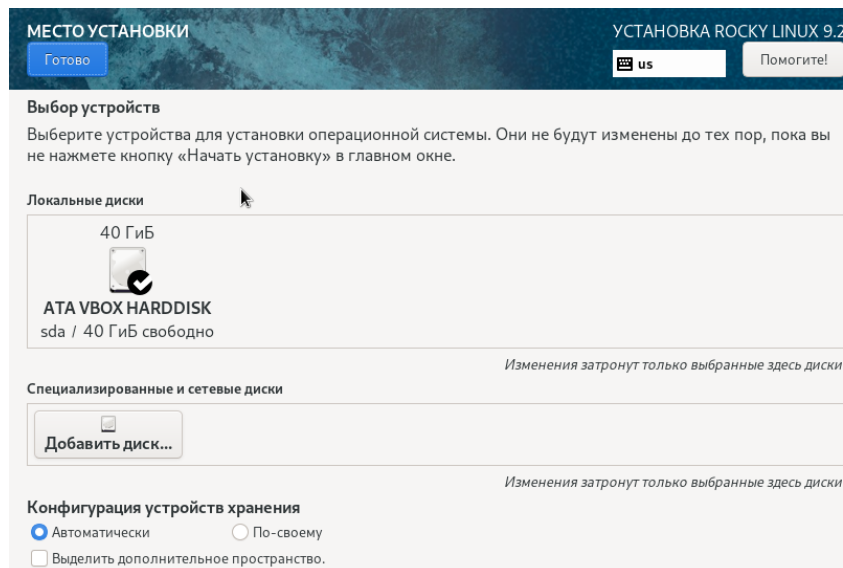


Figure 3.10: Место установки

11) Установка закончена. Перезагружаем.

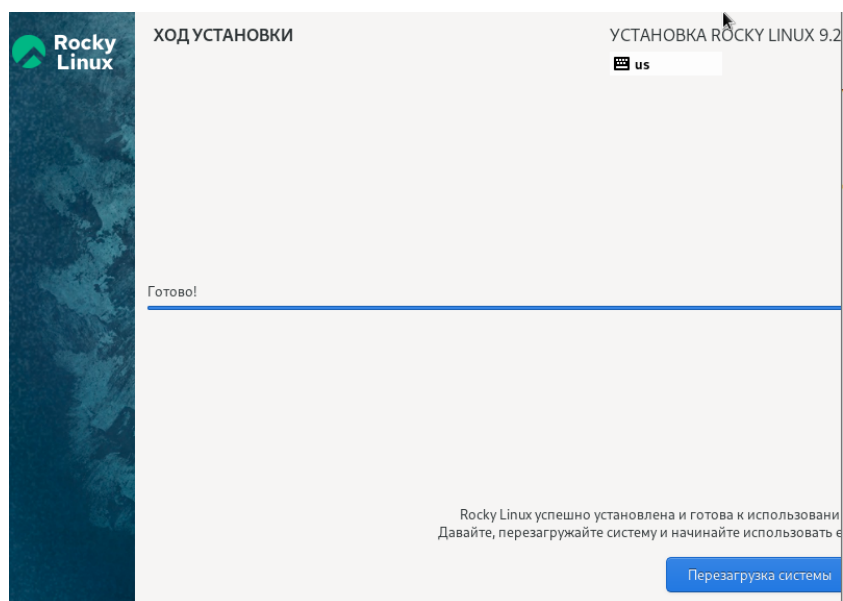
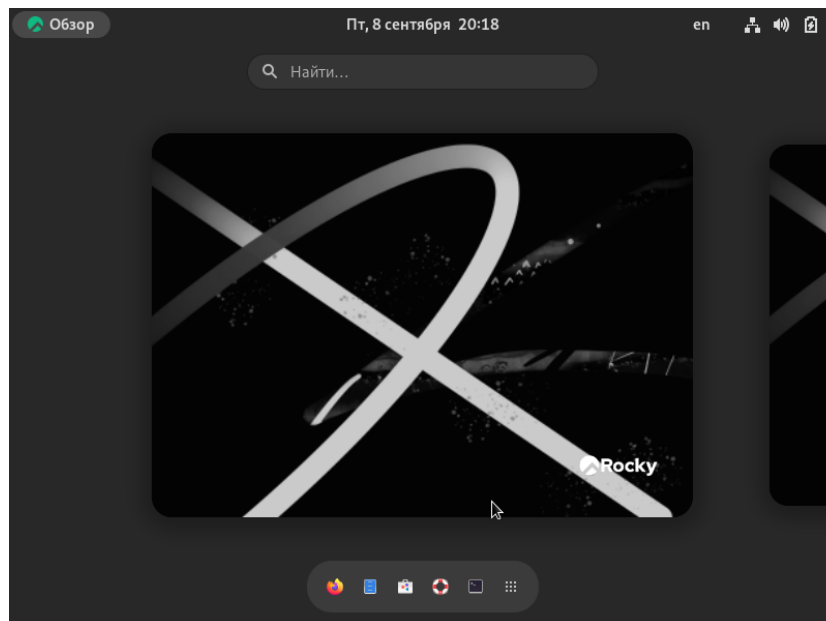


Figure 3.11: Перезагрузка

12) Нас встречает система Rocky Linux. Установка завершена! Осталось подключить образ диска дополнений гостевой ОС.



### ДОМАШНЕЕ ЗАДАНИЕ №1:

1) В окне терминала проанализировал последовательность загрузки системы, выполнив команду `dmesg`. Можно просто просмотреть вывод этой команды:

```
dmesg | less
```

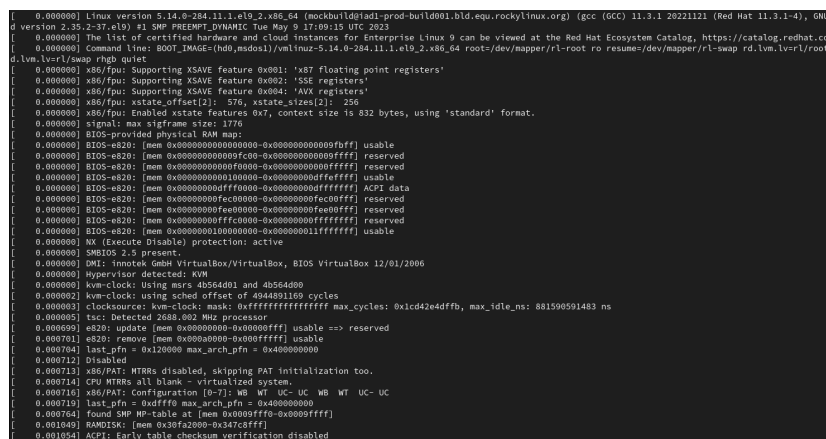


Figure 3.13: dmesg

Можно использовать поиск с помощью `grep:dmesg | grep -i"то, что ищем"` Команда `grep` используется для поиска данных. а. Версия ядра Linux (Linux version).

```
[tadarizhapov@tadarizhapov ~]$ dmesg | grep -i "Linux version"
[ 0.000000] Linux version 5.14.0-284.11.1.el9_2.x86_64 (mockbuild@iad1-prod-build001.bld.equ.rockylinux.org) (gcc (GCC) 11.3.1 20221121 (Red Hat 11.3.1-4), GNU ld version 2.35.2-37.el9) #1 SMP PREEMPT_DYNAMIC Tue May 9 17:09:15 UTC 2023
[tadarizhapov@tadarizhapov ~]$
```

Figure 3.14: Версия ядра Linux

b. Частота процессора (Detected Mhz processor).

```
[tadarizhapov@tadarizhapov ~]$ dmesg | grep -i "MHz"
[ 0.000005] tsc: Detected 2688.002 MHz processor
[ 1.483689] e1000 0000:00:03:0 eth0: (PCI:33MHz:32-bit) 08:00:27:9a:e0:28
[tadarizhapov@tadarizhapov ~]$
```

Figure 3.15: Частота процессора

c. Модель процессора (CPU0)

```
[tadarizhapov@tadarizhapov ~]$ dmesg | grep -i "CPU0"
[ 0.179062] smpboot: CPU0: 11th Gen Intel(R) Core(TM) i5-11400H @ 2.70GHz (family: 0x6, model: 0x8d, stepping: 0x1)
[tadarizhapov@tadarizhapov ~]$
```

Figure 3.16: Модель процессора (CPU0)

d. Объем доступной оперативной памяти (Memory available)

```
[tadarizhapov@tadarizhapov ~]$ dmesg | grep -i "Memory"
[ 0.001079] ACPI: Reserving FACP table memory at [mem 0xdfff00f0-0xdfff01e3]
[ 0.001080] ACPI: Reserving DSDT table memory at [mem 0xdfff0620-0xdfff2972]
[ 0.001080] ACPI: Reserving FACS table memory at [mem 0xdfff0200-0xdfff023f]
[ 0.001081] ACPI: Reserving FACS table memory at [mem 0xdfff0200-0xdfff023f]
[ 0.001081] ACPI: Reserving APIC table memory at [mem 0xdfff0240-0xdfff02ab]
[ 0.001081] ACPI: Reserving SSDT table memory at [mem 0xdfff02b0-0xdfff061b]
[ 0.001388] Early memory node ranges
[ 0.011544] PM: hibernation: Registered nosave memory: [mem 0x00000000-0x00000fff]
```

Figure 3.17: Объем доступной оперативной памяти

e. Тип обнаруженного гипервизора (Hypervisor detected).

```
[tadarizhapov@tadarizhapov ~]$ dmesg | grep -i "Hypervisor detected"
[ 0.000000] Hypervisor detected: KVM
[tadarizhapov@tadarizhapov ~]$
```

Figure 3.18: Объем доступной оперативной памяти

- f. Тип файловой системы корневого раздела. Последовательность монтирования файловых систем

```
[tadarizhapov@tadarizhapov ~]$ dmesg | grep -i "Mount"
[ 0.067079] Mount-cache hash table entries: 8192 (order: 4, 65536 bytes, linear)
[ 0.067084] Mountpoint-cache hash table entries: 8192 (order: 4, 65536 bytes, linear)
[ 3.452290] XFS (dm-0): Mounting V5 Filesystem
[ 3.461344] XFS (dm-0): Ending clean mount
[ 3.926515] systemd[1]: Set up automount Arbitrary Executable File Formats File System Automount Point.
[ 3.931460] systemd[1]: Mounting Huge Pages File System...
[ 3.932842] systemd[1]: Mounting POSIX Message Queue File System...
[ 3.933623] systemd[1]: Mounting Kernel Debug File System...
[ 3.934343] systemd[1]: Mounting Kernel Trace File System...
[ 3.945566] systemd[1]: Starting Remount Root and Kernel File Systems...
[ 3.950705] systemd[1]: Mounted Huge Pages File System.
[ 3.950876] systemd[1]: Mounted POSIX Message Queue File System.
[ 3.951010] systemd[1]: Mounted Kernel Debug File System.
[ 3.951208] systemd[1]: Mounted Kernel Trace File System.
[ 3.954525] systemd[1]: Mounting FUSE Control File System...
[ 3.956333] systemd[1]: Mounting Kernel Configuration File System...
[ 3.957405] systemd[1]: Mounted FUSE Control File System.
[ 4.973558] XFS (sda1): Mounting V5 Filesystem
[ 5.208748] XFS (sda1): Ending clean mount
```

Figure 3.19: Объем доступной оперативной памяти

## КОНТРОЛЬНЫЕ ВОПРОСЫ №1

1) Учётная запись пользователя содержит сведения, необходимые для идентификации пользователя при подключении к системе, такие как имя пользователя, имя хоста и пароль. 2) Команды терминала: а. Для получения справки используется ключ `-help` или команда `man`. Например, `ls -help` или `man ls`. б. Для перемещения по файловой системе используется команда `cd`. Например `cd ~`. в. Для просмотра содержимого каталога используется команда `ls`. Например `ls ~/work`. г. Для определения объёма каталога используется команда `du`. д. Для создания каталогов используется `mkdir`, для удаления пустых каталогов используется `rmdir`. Для создания файлов используется `touch`, для удаления файлов и каталогов используется `rm`. е. Для задания прав используется команда `chmod`. Например, `chmod u-w test.txt`. ж. Для просмотра истории команд используется команда `history`. 3) Файловая система — часть ОС, которая обеспечивает чтение и запись файлов на дисковых носителях информации. а. Ext2 — расширенная файловая система. Данные сначала кэшируются и только потом записываются на диск. б. Ext3 и Ext4 — журналируемые файловые системы. Осуществляется хранение в виде журнала

со списком изменений, что помогает сохранить целостность при сбоях. c.XFS — высокопроизводительная журналируемая файловая система, рассчитанная для работы на дисках большого объёма. 4)Для просмотра подмонтированных в ОС файловых систем необходимо использовать команду `findmnt`. 5)Для удаления зависшего процесса используется команда `kill PID` или `killall название`.

## Часть 2. Работа с GitHub

1)Первым делом, мы регистрируемся на сайте `github.com`. Так как мы не в первый раз имеем дело с `github`, то аккаунт уже готов.

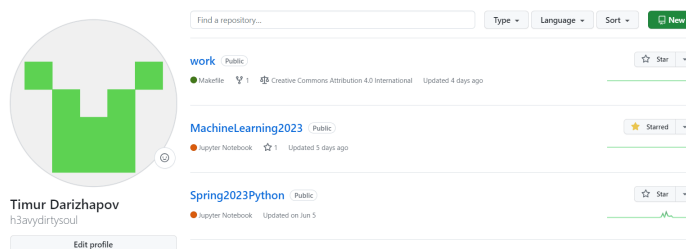


Figure 3.20: Знакомство с github

2)Первоначальная настройка для работы с `github`.

```
[tadarizhapov@tadarizhapov ~]$ git config --global user.name "Timur Darizhapov"
[tadarizhapov@tadarizhapov ~]$ git config --global user.email "timaanddar@mail.ru"
```

Figure 3.21: Настройка

3)Генерация SSH ключа.

```
[tadarizhapov@tadarizhapov ~]$ ssh-keygen -C "Timur Darizhapov <timaanddar@mail.ru>"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/tadarizhapov/.ssh/id_rsa):
Created directory '/home/tadarizhapov/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/tadarizhapov/.ssh/id_rsa
Your public key has been saved in /home/tadarizhapov/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:E9vEvNCK9vVzQ4K/tjqzsjh8nBcQCSGA4CRk6S/dKs Timur Darizhapov <timaanddar@mail.ru>
The key's randomart image is:
+---[RSA 3072]-----+
|+*... .00+|
|*  o. . . |
|..  o =   |
|.  o & .  |
|o . S =...|
|. o . B o..|
|..  + .o..|
|.  + ..o. |
|E    .+0o. |
+---[SHA256]-----+
```

Figure 3.22: Ключ



4) Переходим на сайте в настройки и в SSH Keys. Копируем наш SSH ключ, предварительно выведя его в консоль с помощью `cat ~/.ssh/id_rsa.pub`. Также нужно было написать `git init` в терминале для создания основного дерева репозитория.

```
[tadarizhapov@tadarizhapov tadarizhapov]$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQGC/PfawKPwncdFLpQ62t8xdJ0Yr9yU7N6vu1fAXP8GuyideNG0j/76FVfjCJHncos
gjlaFZYxjmwHm1iAUH5hsA2KIW62ois6IOU0/Xact0gZU4RoxGGAvSVU8o5DF0A2pr+UAnerGCU8+6n0nyLZeN27yju007SRtZCLM
vHwM9kGYzlvjWok8qxZZ+bcL+LJMeQ5U10jrCskQoy0UqWV26f5H8CqSiLkp109XWlkoc6tV2S0hwsQ0B7E3GwL+i1YxgTtPp6pUg
IH3eDLE75oF6Ci8lyftUdq0QA/nEx971v8Nq0k20mMbeIehzc2ccQLDB+TKjQ0fAbwRtgqjSQE0iNVc1fyAitny3u+1dbw+S0pyQnR
bVZJWfpC0UokCwNhmBxJBSJ00D3mXNyOcWUaD82UKMKNSDL74b9wkEOoAJPG2wRG+3wn6FVNm0jckHt/tB9EsMwAZjbZGADiNpwVc5
M+0nDc3XokdtnHsQPnnecGd9UerhdbeLV/Ft57JN8= Timur Darizhapov <timaanddar@mail.ru>
```

Figure 3.23: Вывод ключа в терминале

5) Ключ отобразился на сайте.

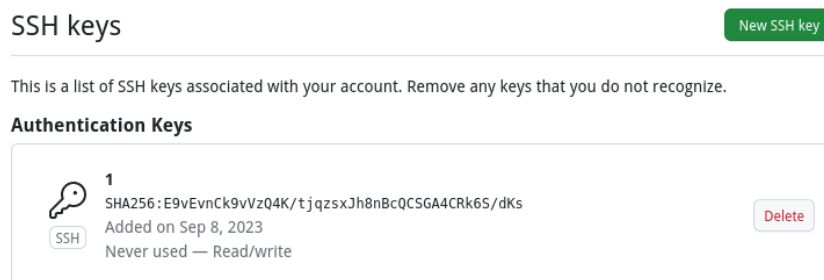


Figure 3.24: Ключ

6) Клонировать наш репозиторий `work` из 2021, когда мы работали с ОС. Создаём рабочее пространство для наших лабораторных работ, как сказано в документации.

```
[tadarizhapov@tadarizhapov tadarizhapov]$ git clone git@github.com:h3avydirtysoul/work.git
Клонирование в «work...
The authenticity of host 'github.com [140.82.121.3]' can't be established.
ED25519 key fingerprint is SHA256:D1Y3wrvV6TUJJhbpZ1sF/zLDA0zPMSvHdkr4UvC0Qu.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.
remote: Enumerating objects: 684, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 684 (delta 0), reused 0 (delta 0), pack-reused 677
Получение объектов: 100% (684/684), 32.46 МБ | 1.42 МБ/с, готово.
Определение изменений: 100% (128/128), готово.
[tadarizhapov@tadarizhapov tadarizhapov]$ ls
work
[tadarizhapov@tadarizhapov tadarizhapov]$ cd work/
[tadarizhapov@tadarizhapov work]$ ls
2020-2021 legalcode.txt LICENSE README.md
[tadarizhapov@tadarizhapov work]$ mkdir 2023-2024
[tadarizhapov@tadarizhapov work]$ cd 2023-2024/
2023-2024]$ mkdir Информационная безопасность
[tadarizhapov@tadarizhapov 2023-2024]$ cd Информационная/
Информационная]$ cd ..
[tadarizhapov@tadarizhapov 2023-2024]$ mkdir 'Информационная безопасность'
[tadarizhapov@tadarizhapov 2023-2024]$ ls
безопасность Информационная 'Информационная безопасность'
[tadarizhapov@tadarizhapov 2023-2024]$ rm -R безопасность Информационная
[tadarizhapov@tadarizhapov 2023-2024]$ ls
Информационная безопасность
[tadarizhapov@tadarizhapov 2023-2024]$ cd Информационная/ безопасность/
Информационная безопасность]$ ls
Информационная безопасность]$ mkdir infosec
Информационная безопасность]$ cd infosec/
infosec]$ mkdir labs
infosec]$ cd labs/
labs]$ mkdir lab01
labs]$ cd lab01/
lab01]$ touch report.md
lab01]$ ls
report.md
```

Figure 3.25: Рабочее пространство

7)Пишем `git add` . После этого делаем первый коммит за 2 года. Отправляем изменения на `github`.

[illegible]

Figure 3.26: Коммит

Вторая часть закончена. Мы связали наш репозиторий с репозиторием на [github.com](https://github.com).

## КОНТРОЛЬНЫЕ ВОПРОСЫ №2:

1) Система контроля версий Git представляет собой набор программ командной строки. Доступ к ним можно получить из терминала посредством ввода команды `git` различными опциями. Системы контроля версий (Version Control

System,VCS)применяются при работе нескольких человек над одним проектом.

2)В классических системах контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов. Выполнение большинства функций по управлению версиями осуществляется специальным сервером.Участник проекта (пользователь) перед началом работы посредством определённыхкоманд получает нужную ему версию файлов. После внесения изменений, пользователь размещает новую версию в хранилище. При этом предыдущие версии не удаляютсяиз центрального хранилища и к ним можно вернуться в любой момент. Сервер может сохранять неполную версию изменённых файлов, а производить так называемую дельта-компрессию—сохранять только изменения между последовательными версиями,чтопозволяет уменьшить объём хранимых данных. Системы контроля версий также могут обеспечивать дополнительные, более гибкие функциональные возможности. Например,они могут поддерживать работу с несколькими версиями одного файла,сохраняя общую историю изменений до точки ветвления версий и собственные истории изменений каждой ветви. Крометого, обычно доступна информация о том, кто из участников, когда и какие изменения вносил. Обычно такого рода информация хранится в журнале изменений, доступ к которому можно ограничить.

3)Централизованные системы — это системы, которые используют архитектуру клиент / сервер, где один или несколько клиентских узлов напрямую подключены к центральному серверу. Пример - Wikipedia. В децентрализованных системах каждый узел принимает свое собственное решение. Конечное поведение системы является совокупностью решений отдельных узлов. Пример — Bitcoin. В классических системах контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов.Выполнение большинства функций по управлению версиями осуществляется специальным сервером.

4)Создадим локальный репозиторий. Сначала сделаем предварительную

конфигурацию, указав имя и email владельца репозитория: `git config --global user.name "Имя Фамилия"` `git config --global user.email "work@mail"` и настроив utf-8 в выводе сообщений: `git config --global quotePath false` Для инициализации локального репозитория, расположенного, например, в каталоге `~/tutorial`, необходимо ввести в командной строке: `cd mkdir tutorial cd tutorial git init`

5) Для последующей идентификации пользователя на сервере репозитория необходимо сгенерировать пару ключей (приватный и открытый): `ssh-keygen -C "Имя Фамилия work@mail"` Ключи сохраняются в каталоге `~/.ssh/`. Скопировав из локальной консоли ключ в буфер обмена `cat ~/.ssh/id_rsa.pub | xclip -sel clip` вставляем ключ в появившееся на сайте поле.

6) У Git две основных задачи: первая — хранить информацию о всех изменениях в вашем коде, начиная с самой первой строчки, а вторая — обеспечение удобства командной работы над кодом.

7) Основные команды git: Наиболее часто используемые команды git: – создание основного дерева репозитория: `git init` – получение обновлений (изменений) текущего дерева из центрального репозитория: `git pull` – отправка всех произведённых изменений локального дерева в центральный репозиторий: `git push` – просмотр списка изменённых файлов в текущей директории: `git status` – просмотр текущих изменений: `git diff` – сохранение текущих изменений: – добавить все изменённые и/или созданные файлы и/или каталоги: `git add .` – добавить конкретные изменённые и/или созданные файлы и/или каталоги: `git add имена_файлов` – удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории): `git rm имена_файлов` – сохранение добавленных изменений: – сохранить все добавленные изменения и все изменённые файлы: `git commit -am "Описание коммита"` – сохранить добавленные изменения с внесением комментария через встроенный редактор: `git commit` – создание новой ветки, базирующейся на текущей: `git checkout -b имя_ветки` – переключение на некоторую ветку: `git checkout имя_ветки` (при переключении на ветку, которой ещё нет в локальном репозитории, она будет создана и связана с удалённой) – от-

правка изменений конкретной ветки в центральный репозиторий: `git push origin имя_ветки`—слияние ветки стекущим деревом: `git merge --no-ff имя_ветки`—удаление ветки: – удаление локальной уже слитой с основным деревом ветки: `git branch -D имя_ветки`—принудительное удаление локальной ветки: `git branch -D имя_ветки`—удаление ветки с центрального репозитория: `git push origin :имя_ветки`

8)Использования `git` при работе с локальными репозиториями (добавления текстового документа в локальный репозиторий): `git add hello.txt git commit -am'Новый файл`

9)Проблемы, которые решают ветки `git`: • нужно постоянно создавать архивы с рабочим кодом • сложно “переключаться” между архивами • сложно перетаскивать изменения между архивами • легко что-то напутать или потерять

10)Во время работы над проектом так или иначе могут создаваться файлы, которые не требуется добавлять в последствии в репозиторий. Например, временные файлы, создаваемые редакторами, или объектные файлы, создаваемые компиляторами. Можно прописать шаблоны игнорируемых при добавлении в репозиторий типов файлов в файл `.gitignore` с помощью `ервисов`. Для этого сначала нужно получить списки имеющихся шаблонов: `curl -L -s https://www.gitignore.io/api/list` Затем скачать шаблон, например, для C и C++ `curl -L -s https://www.gitignore.io/api/c » .gitignore` `curl -L -s https://www.gitignore.io/api/c++ » .gitignore`

### Часть 3. Создание Markdown файла

1)Для того, чтобы из Markdown получить PDF и DOCX файлы, требуется скачать `pandoc` и `pandoc-crossref`. Это непростая задача, пришлось скачивать их с `github.com` старой версии 2.14. Переместить исполняемые файлы в папку `~/bin`. Сперва скачаем zip-файлы из гитхабов `jgm` и `lierdakil`. Смотрим ветки и выбираем всё для `pandoc 2.14`.



Figure 3.27: pandoc

2) Таким образом, как на скриншоте, мы перемещаем в корневую папку pandoc. Аналогично с pandoc-crossref.

```
[tadarizhapov@tadarizhapov ~]$ cd pandoc-2.14-linux-amd64/
[tadarizhapov@tadarizhapov pandoc-2.14-linux-amd64]$ ls
pandoc-2.14
[tadarizhapov@tadarizhapov pandoc-2.14-linux-amd64]$ cd pandoc-2.14/
[tadarizhapov@tadarizhapov pandoc-2.14]$ cd bin
[tadarizhapov@tadarizhapov bin]$ ls
pandoc
[tadarizhapov@tadarizhapov bin]$ sudo cp -R pandoc /bin

Мы полагаем, что ваш системный администратор изложил вам основы
безопасности. Как правило, всё сводится к трём следующим правилам:

№1) Уважайте частную жизнь других.
№2) Думайте, прежде что-то вводить.
№3) С большой властью приходит большая ответственность.

[sudo] пароль для tadarizhapov:
[tadarizhapov@tadarizhapov bin]$
```

Figure 3.28: Установка pandoc

3) В консольной строке пришем make и получаем наши файлы. Примечание: компиляцию файлов я делал в Ubuntu, так как в Rocky это делается немножко криво. Нужно не забыть написать команду `sudo apt-get install texlive-full`, чтобы все нужные шрифты были готовы.

## 4 Выводы

Я установил VirtualBox, поставил на виртуальную машину Rocky Linux. Разобрался с системой контроля версий и github. Скомпилировал файлы из формата md в pdf и docx.