

Отчёт по лабораторной работе 1

Соболев Максим Сергеевич

Содержание

1	Цель работы	5
1.1	Установка и конфигурация операционной системы на виртуальную машину	5
1.2	Управление версиями	5
1.3	Markdown	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы (часть 1)	9
4.1	Создание виртуальной машины	9
4.1.1	Настройка виртуального жесткого диска	10
4.1.2	Настройка виртуального процессора	10
4.1.3	Запуск виртуальной машины	11
4.2	Начало установки	12
4.3	Окно с общими сведениями о параметрах будущей установки	12
4.4	Наблюдаем за процессом установки	14
4.5	После окончания процесса установки	14
4.6	Вывод dmesg less	15
4.7	Версия ядра Linux	16
4.8	Сведения о частоте и модели процессора	16
4.9	Сведения о доступной ОЗУ	16
4.10	Сведения об используемом гипервизоре	16
4.11	Сведения о типе файловой системы корневого раздела	17
4.12	Последовательность монтирования файловых систем	17
5	Контрольные вопросы (часть 1)	18
5.1	Какую информацию содержит учётная запись пользователя?	18
5.2	Укажите команды терминала и приведите примеры:	18
5.2.1	Получение справки по команде	18
5.2.2	Перемещение по файловой системе	18
5.2.3	Просмотра содержимого каталога	18
5.2.4	Определение объёма каталога	19
5.2.5	Создание / удаление каталогов / файлов	19
5.2.6	Задание определённых прав на файл / каталог	19
5.2.7	Просмотр истории команд	19

5.3	Что такое файловая система? Приведите примеры с краткой характеристикой.	19
5.4	Как посмотреть, какие файловые системы подмонтированы в ОС?	20
5.5	Как удалить зависший процесс?	20
6	Выполнение лабораторной работы (часть 2)	21
6.1	Настройка GitHub	21
6.2	Установка git-flow	21
6.3	Установка gh	21
6.4	Базовая настройка git	22
6.5	Создание ключей ssh	22
6.5.1	Указываем ключи ssh на GitHub	22
6.6	Создание ключей gpg	22
6.6.1	Указываем ключи gpg на GitHub	23
6.7	Настройка автоматических подписей коммитов git	23
6.8	Настройка gh	23
6.9	Создание репозитория курса на основе шаблона	24
6.10	Настраиваем каталог курса, создаем необходимые каталоги, отправляем результат на сервер	24
7	Контрольные вопросы (часть 2)	25
7.1	Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются?	25
7.2	Объясните следующие понятия VCS и их отношения:	25
7.2.1	Хранилище	25
7.2.2	Commit	26
7.2.3	История	26
7.2.4	Рабочая копия	26
7.3	Что представляют собой и чем отличаются централизованные и децентрализованные VCS	26
7.4	Опишите действия с VCS при единоличной работе с хранилищем.	26
7.5	Опишите порядок работы с общим хранилищем VCS.	27
7.6	Каковы основные задачи, решаемые инструментальным средством git?	27
7.7	Назовите и дайте краткую характеристику командам git.	27
7.8	Приведите примеры использования при работе с локальным и удалённым репозиториями.	28
7.9	Что такое и зачем могут быть нужны ветви (branches)?	28
7.10	Как и зачем можно игнорировать некоторые файлы при commit? .	28
8	Выводы	29
	Список литературы	30

Список иллюстраций

4.1	Создание виртуальной машины	9
4.2	Настройка виртуального ЖД	10
4.3	Настройка виртуального ЦП	11
4.4	Кнопка запуска ВМ	11
4.5	Название рисунка	12
4.6	Сведения об установке	13
4.7	Окно создания пользователя	13
4.8	Синяя полоса прогресса установки	14
4.9	Окончание процесса установки	15
4.10	dmesg less	15
4.11	Версия ядра Linux	16
4.12	Частота процессора	16
4.13	Модель процессора	16
4.14	Доступная ОЗУ	16
4.15	Используемый гипервизор	16
4.16	Файловая система корневого раздела	17
4.17	Последовательность монтирования ФС	17
6.1	Установка gh	21
6.2	Настройка git	22
6.3	Создание ключей ssh	22
6.4	Создание ключей gpg	23
6.5	Настройка автоматических подписей коммитов git	23
6.6	Настройка gh	23
6.7	Создание репозитория	24
6.8	Настройка репозитория курса	24

1 Цель работы

1.1 Установка и конфигурация операционной системы на виртуальную машину

Целью данной работы является приобретение практических навыков установки операционной системы на виртуальную машину, настройки минимально необходимых для дальнейшей работы сервисов.

1.2 Управление версиями

Изучить идеологию и применение средств контроля версий.

Освоить умения по работе с git.

1.3 Markdown

Научиться оформлять отчёты с помощью легковесного языка разметки Markdown.

2 Задание

- Установить ОС Linux, с использованием виртуальной машины.
- Освоить умения по работе с git.
- Оформить отчет в Markdown.

3 Теоретическое введение

- Виртуальная машина — программная система, эмулирующая аппаратное обеспечение некоторой платформы и исполняющая программы для guest-платформы на host-платформе или виртуализирующая некоторую платформу и создающая на ней среды, изолирующие друг от друга программы и даже операционные системы;
- Linux в части случаев — семейство Unix-подобных операционных систем на базе ядра Linux, включающих тот или иной набор утилит и программ проекта GNU, и, возможно, другие компоненты. Как и ядро Linux, системы на его основе, как правило, создаются и распространяются в соответствии с моделью разработки свободного и открытого программного обеспечения. Linux-системы распространяются в основном бесплатно в виде различных дистрибутивов — в форме, готовой для установки и удобной для сопровождения и обновлений, — и имеющих свой набор системных и прикладных компонентов, как свободных, так и проприетарных.
- git — распределённая система управления версиями. Проект был создан Линусом Торвальдсом для управления разработкой ядра Linux, первая версия выпущена 7 апреля 2005 года.
- GitHub — крупнейший веб-сервис для хостинга IT-проектов и их совместной разработки с использованием git.
- Markdown — облегчённый язык разметки, созданный с целью обозначения форматирования в простом тексте, с максимальным сохранением его читаемости человеком, и пригодный для машинного преобразования в языки

для продвинутых публикаций (HTML, Rich Text и других).

4 Выполнение лабораторной работы (часть 1)

4.1 Создание виртуальной машины

В этом окне необходимо указать название, директорию установки и тип гостевой ОС (Windows, Linux или другая). Так же предлагается выбрать объем доступной оперативной памяти и способ создания виртуального жесткого диска.

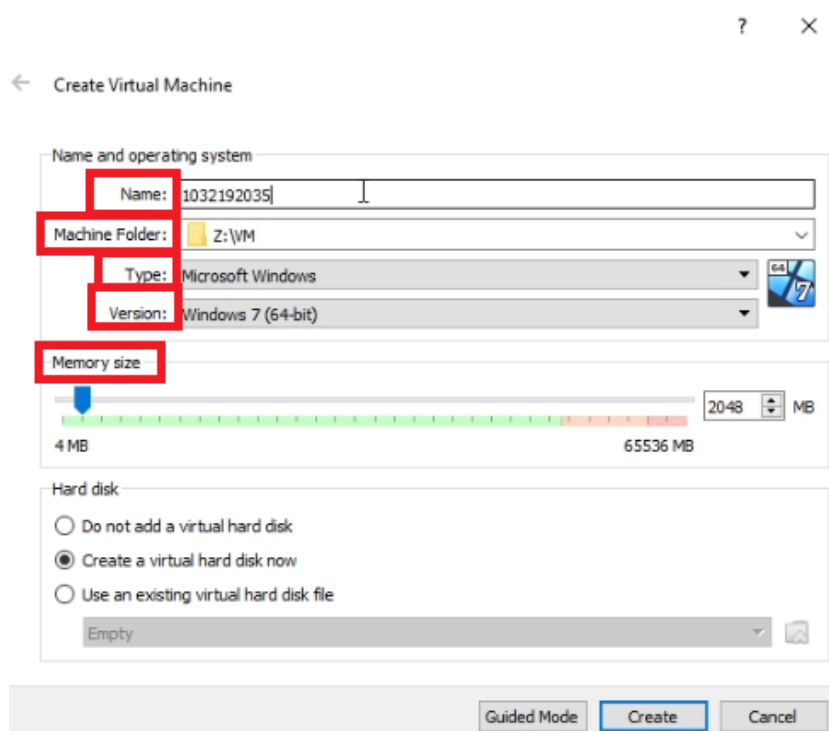


Рис. 4.1: Создание виртуальной машины

4.1.1 Настройка виртуального жесткого диска

Здесь указываем объем виртуального жесткого диска, место его хранения.

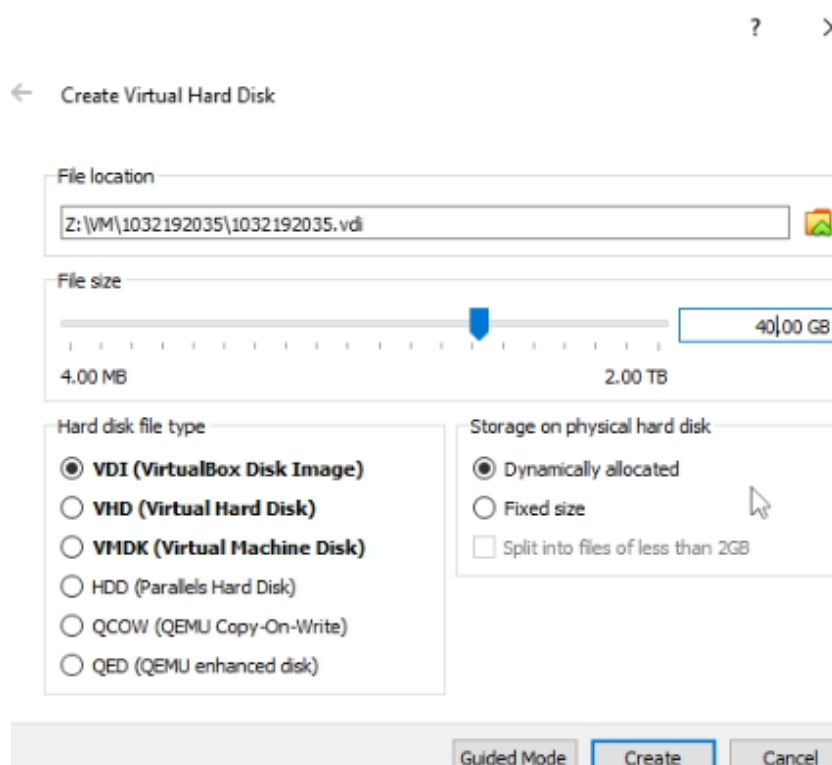


Рис. 4.2: Настройка виртуального ЖД

4.1.2 Настройка виртуального процессора

Добавляем виртуальные ядра, что бы ВМ шевелилась быстрее.

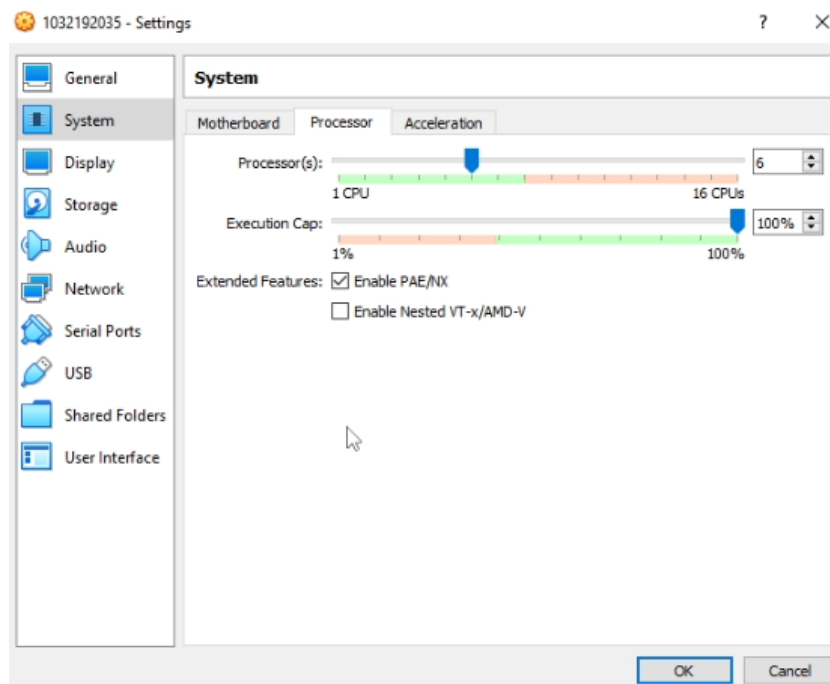


Рис. 4.3: Настройка виртуального ЦП

4.1.3 Запуск виртуальной машины

Нажимаем кнопку “Start”, виртуальная машина запускается.

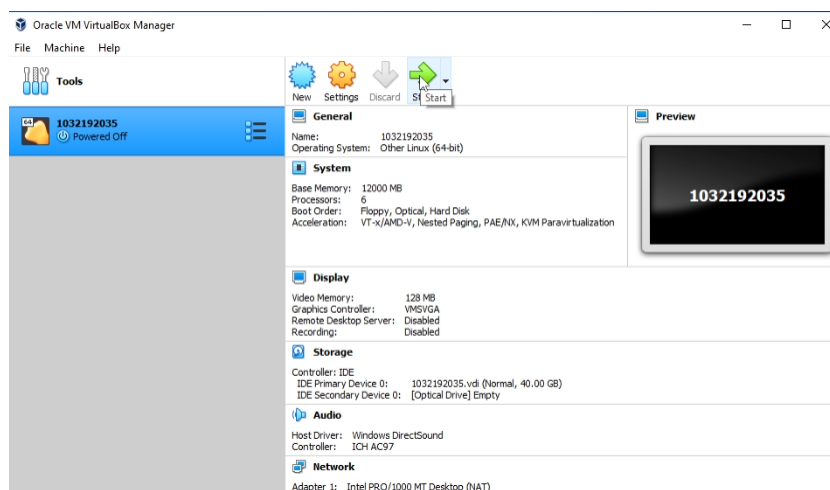


Рис. 4.4: Кнопка запуска VM

4.2 Начало установки

Сразу после загрузки программы установки ОС, нас встречает экран приветствия и предложение выбрать язык интерфейса.

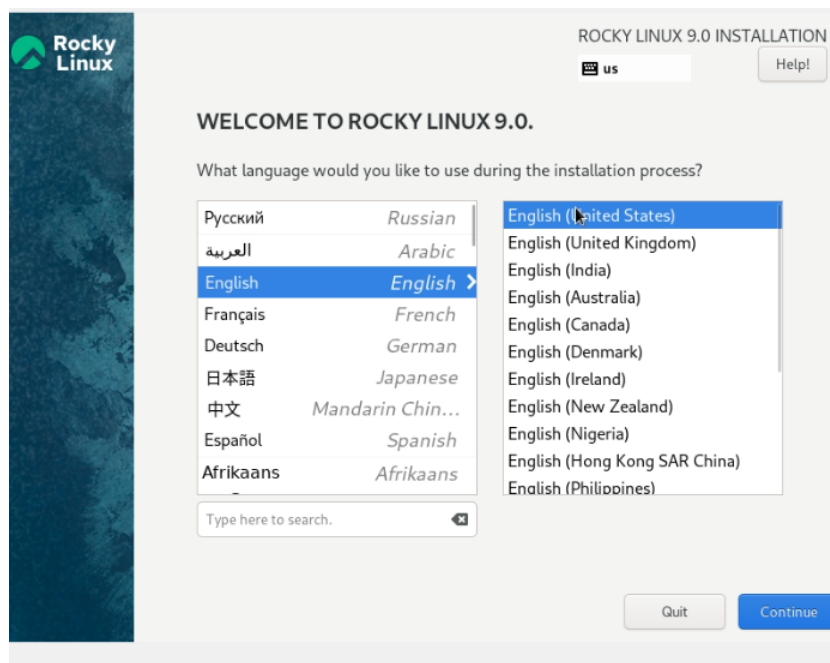


Рис. 4.5: Название рисунка

4.3 Окно с общими сведениями о параметрах будущей установки

Здесь перед установкой мы можем увидеть общие сведения о параметрах будущей установки и настроить их.

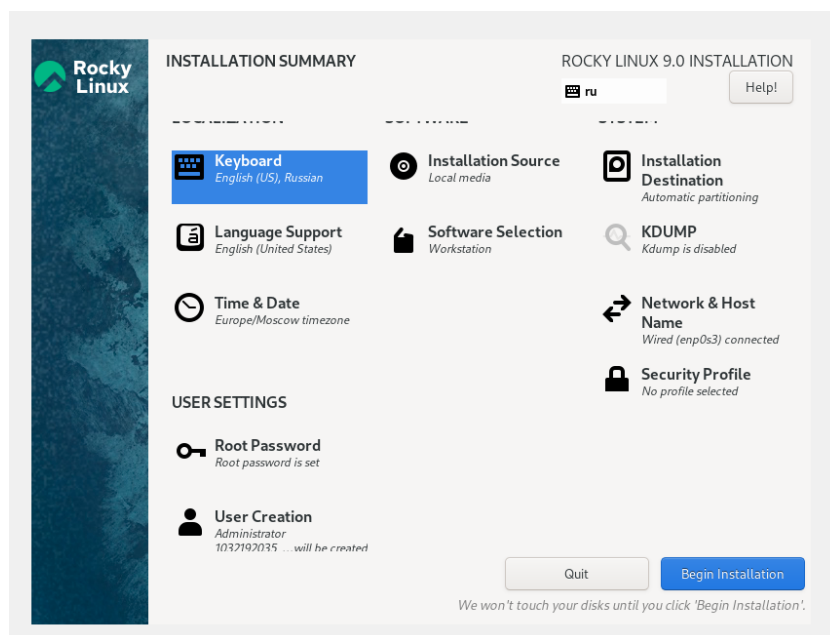


Рис. 4.6: Сведения об установке

Настраиваем место установки, время, параметры раскладки клавиатуры и т.д.
Данные пользователя в соответствии с соглашением:

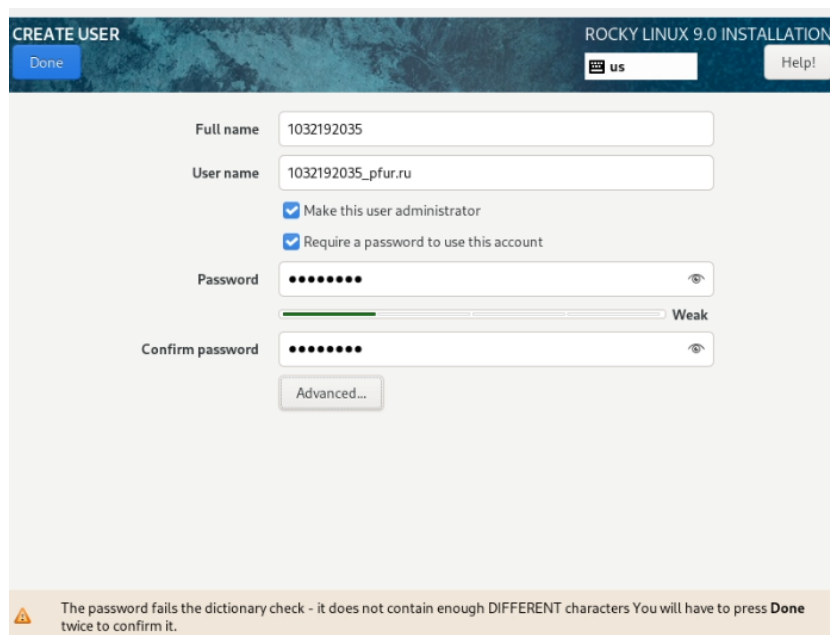


Рис. 4.7: Окно создания пользователя

4.4 Наблюдаем за процессом установки

Откинувшись на спинку кресла внимательно наблюдаем за синей полоской прогресса установки и пьем чай.

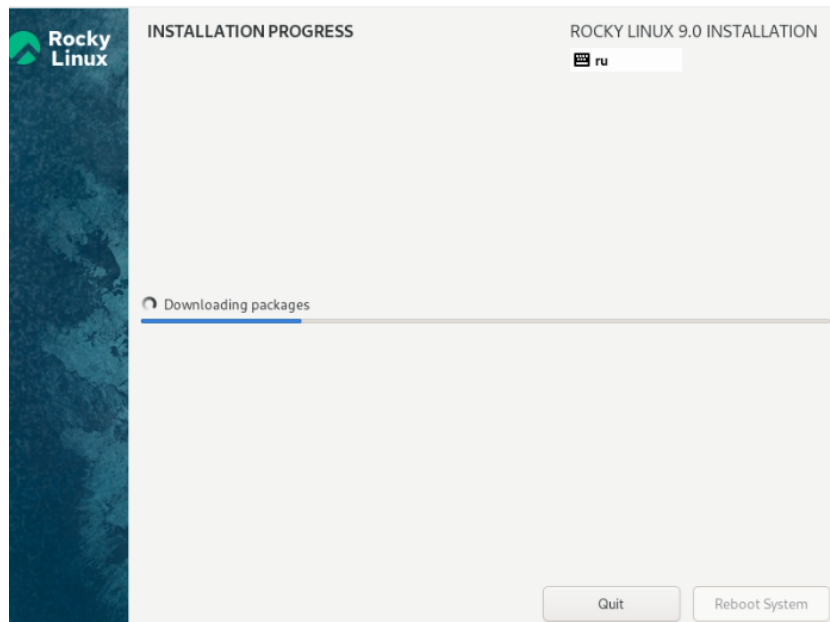


Рис. 4.8: Синяя полоса прогресса установки

4.5 После окончания процесса установки

После окончания процесса установки виртуальная машина автоматически перезагрузится. Нас встречает окно выбора пользователя.

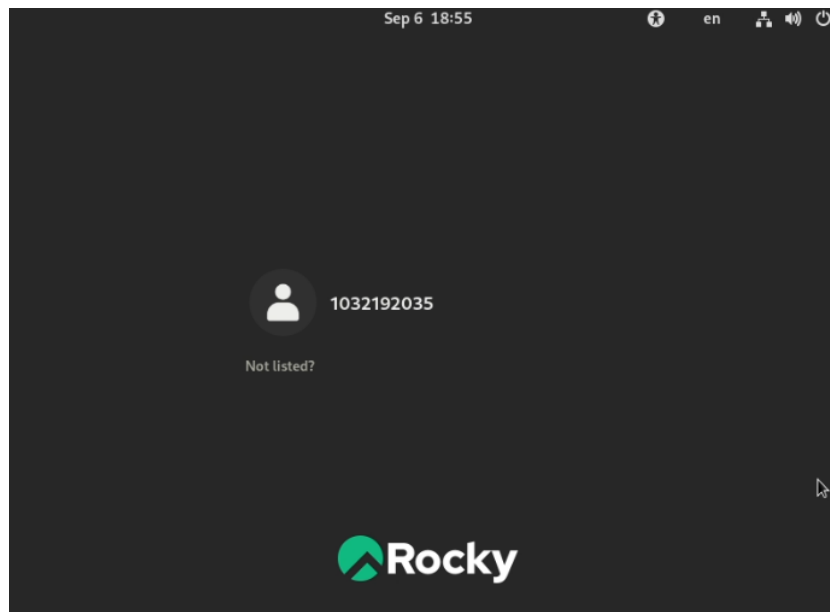


Рис. 4.9: Окончание процесса установки

4.6 Вывод dmesg | less

Начинаем выполнять задания.

```

1032192035_pfur.ru@1032192035:~ — less
GNU ld version 2.35.2-17.el9) #1 SMP PREEMPT Wed May 25 21:01:57 UTC 2022
[ 0.000000] The list of certified hardware and cloud instances for Red Hat Enterprise Linux 9 can be viewed at the Red Hat Ecosystem Catalog, https://catalog.redhat.com.
[ 0.000000] Command line: BOOT_IMAGE=(hd0,msdos1)/vmlinuz-5.14.0-70.13.1.el9_0.x86_64 root=/dev/mapper/rl-root ro resume=/dev/mapper/rl-swap rd.lvm.lv=rl/root rd.lvm.lv=rl/swap rhgb quiet
[ 0.000000] x86/fpu: Supporting XSAVE feature 0x001: 'x87 floating point registers'
[ 0.000000] x86/fpu: Supporting XSAVE feature 0x002: 'SSE registers'
[ 0.000000] x86/fpu: Supporting XSAVE feature 0x004: 'AVX registers'
[ 0.000000] x86/fpu: xstate_offset[2]: 576, xstate_sizes[2]: 256
[ 0.000000] x86/fpu: Enabled xstate features 0x7, context size is 832 bytes, using 'standard' format.
[ 0.000000] signal: max sigframe size: 1776
[ 0.000000] BIOS-provided physical RAM map:
[ 0.000000] BIOS-e820: [mem 0x0000000000000000-0x000000000009fbff] usable
[ 0.000000] BIOS-e820: [mem 0x000000000009fc00-0x000000000009ffff] reserved
[ 0.000000] BIOS-e820: [mem 0x00000000000f0000-0x00000000000fffff] reserved
[ 0.000000] BIOS-e820: [mem 0x0000000000100000-0x0000000000dfffff] usable
[ 0.000000] BIOS-e820: [mem 0x000000000ffff0000-0x000000000ffffffffff] ACPI data
[ 0.000000] BIOS-e820: [mem 0x00000000fec00000-0x00000000fec00fff] reserved
[ 0.000000] BIOS-e820: [mem 0x00000000fee00000-0x00000000fee00fff] reserved
:

```

Рис. 4.10: dmesg | less

4.7 Версия ядра Linux

```
[1032192035_pfur.ru@1032192035 ~]$ dmesg | grep "Linux version"
[ 0.000000] Linux version 5.14.0-70.13.1.el9_0.x86_64 (mockbuild@dal1-prod-builder001.bld.equ.rockylinux.org) (gcc (GCC) 11.2.1 20220127 (Red Hat 11.2.1-9), GNU ld version 2.35.2-17.el9) #1 SMP PREEMPT Wed May 25 21:01:57 UTC 2022
```

Рис. 4.11: Версия ядра Linux

4.8 Сведения о частоте и модели процессора

```
[1032192035_pfur.ru@1032192035 ~]$ dmesg | grep "Detected"
[ 0.000005] tsc: Detected 3599.998 MHz processor
[ 0.923341] systemd[1]: Detected virtualization oracle.
[ 0.923342] systemd[1]: Detected architecture x86-64.
[ 3.794560] systemd[1]: Detected virtualization oracle.
[ 3.794562] systemd[1]: Detected architecture x86-64.
```

Рис. 4.12: Частота процессора

```
[ 3.794562] systemd[1]: Detected architecture x86-64.
[1032192035_pfur.ru@1032192035 ~]$ dmesg | grep "CPU0"
[ 0.152265] smpboot: CPU0: Intel(R) Core(TM) i9-9900KF CPU @ 3.60GHz (family: 0x6, model: 0x9e, stepping: 0xd)
```

Рис. 4.13: Модель процессора

4.9 Сведения о доступной ОЗУ

```
[1032192035_pfur.ru@1032192035 ~]$ dmesg | grep "Memory: "
[ 0.026659] Memory: 3680172K/12287544K available (14345K kernel code, 5945K r
wdata, 9052K rodata, 2548K init, 5460K bss, 392920K reserved, 0K cma-reserved)
```

Рис. 4.14: Доступная ОЗУ

4.10 Сведения об используемом гипервизоре

```
[1032192035_pfur.ru@1032192035 ~]$ dmesg | grep "Hypervisor"
[ 0.000000] Hypervisor detected: KVM
```

Рис. 4.15: Используемый гипервизор

4.11 Сведения о типе файловой системы корневого раздела

Раздел / форматирован как XFS. XFS — высокопроизводительная журналируемая ФС, используемая в Rocky Linux по-умолчанию

```
[1032192035_pfur.ru@1032192035 ~]$ df -Th | grep "root"
/dev/mapper/rl-root xfs      35G  5.4G  30G  16% /
[1032192035_pfur.ru@1032192035 ~]$
```

Рис. 4.16: Файловая система корневого раздела

4.12 Последовательность монтирования файловых систем

```
[1032192035_pfur.ru@1032192035 ~]$ cat /etc/fstab
#
# /etc/fstab
# Created by anaconda on Tue Sep  6 15:50:03 2022
#
# Accessible filesystems, by reference, are maintained under '/dev/disk/'.
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info.
#
# After editing this file, run 'systemctl daemon-reload' to update systemd
# units generated from this file.
#
/dev/mapper/rl-root      /                    xfs     defaults        0 0
UUID=22661f04-8c06-48df-8a26-3e7a8de67657 /boot                xfs     defaults        0 0
/dev/mapper/rl-swap      none                 swap    defaults        0 0
```

Рис. 4.17: Последовательность монтирования ФС

5 Контрольные вопросы (часть 1)

5.1 Какую информацию содержит учётная запись пользователя?

Учётная запись содержит имя учётной записи и список групп, в которые она включена;

“Учётки” пользователей могут иметь путь к shell, hash пароля и путь к домашней папке.

5.2 Укажите команды терминала и приведите примеры:

5.2.1 Получение справки по команде

```
man <команда>
```

5.2.2 Перемещение по файловой системе

```
cd <относительный или абсолютный путь>
```

5.2.3 Просмотра содержимого каталога

```
ls
```

5.2.4 Определение объёма каталога

```
du -sh
```

5.2.5 Создание / удаление каталогов / файлов

`rm` <название файла> (с различными флагами) для удаления файлов/каталогов;

`touch` <название файла> для создания файла;

`mkdir` <название каталога> для создания каталога.

5.2.6 Задание определённых прав на файл / каталог

```
chmod XXX
```

5.2.7 Просмотр истории команд

```
history
```

5.3 Что такое файловая система? Приведите примеры с краткой характеристикой.

Файловая система – это инструмент, позволяющий операционной системе и программам обращаться к нужным файлам и работать с ними. При этом программы оперируют только названием файла, его размером и датой создания. Все остальные функции по поиску необходимого файла в хранилище и работе с ним берет на себя файловая система накопителя.

Пример: FAT32

Современная версия FAT32 вышла в 1995 году. Она может работать с томами размером до 32 Гб и файлами размером до 4 Гб. При этом система не работает с накопителями объемом более 8 Тб. Поэтому сегодня FAT32 используется в

основном только на флешках, картах памяти фотоаппаратов и музыкальных плееров.

5.4 Как посмотреть, какие файловые системы подмонтированы в ОС?

```
df
```

5.5 Как удалить зависший процесс?

```
kill <PID процесса>
```

6 Выполнение лабораторной работы (часть 2)

6.1 Настройка GitHub

Аккаунт GitHub был создан ранее. В данной работе регистрация не рассматривается.

6.2 Установка git-flow

git-flow был установлен из пакетов, скачанных в интернете, поскольку при попытке сборки из исходников возникли проблемы, связанные с доступностью одного из сабрепозиториев.

6.3 Установка gh

gh – это утилита командной строки для управления репозиториями на GitHub.

```
[1032192035 pfur.ru@1032192035 ~]$ dnf install 'dnf-command(config-manager)'
Error: This command has to be run with superuser privileges (under the root user on most systems).
[1032192035 pfur.ru@1032192035 ~]$ sudo !!
sudo dnf install 'dnf-command(config-manager)'
[sudo] password for 1032192035 pfur.ru:
Last metadata expiration check: 0:02:42 ago on Tue 06 Sep 2022 07:07:33 PM MSK.
Package dnf-plugins-core-4.0.24-4.el9_0.noarch is already installed.
Dependencies resolved.
Nothing to do.
Complete!
[1032192035 pfur.ru@1032192035 ~]$ sudo su
[root@1032192035 1032192035 pfur.ru]# dnf config-manager --add-repo https://cli.github.com/packages/rpm/gh-cli.repo
Adding repo from: https://cli.github.com/packages/rpm/gh-cli.repo
[root@1032192035 1032192035 pfur.ru]# dnf install gh
```

Рис. 6.1: Установка gh

6.4 Базовая настройка git

Вводим свои данные. Настраиваем в соответствии с заданием.

```
[root@1032192035 gitflow]# git config --global user.name "Maksim Sobolev"
[root@1032192035 gitflow]# git config --global user.email "sobolek322lorek@gmail.com"
[root@1032192035 gitflow]# git config --global core.quotepath false
[root@1032192035 gitflow]# git config --global init.defaultBranch master
[root@1032192035 gitflow]# git config --global core.autocrlf input
[root@1032192035 gitflow]# git config --global core.safecrlf warn
```

Рис. 6.2: Настройка git

6.5 Создание ключей ssh

```
[root@1032192035 gitflow]# ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/root/.ssh/id_ed25519):
Created directory '/root/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_ed25519
Your public key has been saved in /root/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:WuVEsIvGiOpcUSAIDRfE3VVqsvczignWYcZTj0poXas root@1032192035.local
The key's randomart image is:
+--[ED25519 256]--+
|=*++ . .000      |
|..+ o .o +       |
|   .o * o        |
|..= * =          |
|..o % S .        |
|. +.* B .        |
|. o.= + +        |
|o... o o . o     |
| o E o .         |
+-----[SHA256]-----+
```

Рис. 6.3: Создание ключей ssh

6.5.1 Указываем ключи ssh на GitHub

Заходим на сайт, в профиле указываем публичный ключ ssh.

6.6 Создание ключей gpg

Создаем ключи gpg.

```
[root@1032192035 gitflow]# gpg --full-generate-key
public and secret key created and signed.

pub   rsa4096 2022-09-06 [SC] [expires: 2022-12-05]
       1320C05ED558788A1A3D988D964FDB6455A76ABD
uid           Maksim Sobolev <sobolek322lorek@gmail.com>
sub   rsa4096 2022-09-06 [E] [expires: 2022-12-05]
```

Рис. 6.4: Создание ключей gpg

6.6.1 Указываем ключи gpg на GitHub

Аналогично п. 5.1. указываем ключи gpg.

6.7 Настройка автоматических подписей коммитов git

В соответствии с заданием включаем автоматическую подпись коммитов и указываем там свой ключ.

```
[root@1032192035 gitflow]# git config --global user.signingKey 964FDB6455A76ABD
[root@1032192035 gitflow]# git config --global commit.gpgSign true
[root@1032192035 gitflow]# git config --global gpg.program $(which gpg)
```

Рис. 6.5: Настройка автоматических подписей коммитов git

6.8 Настройка gh

Логинимся в GitHub с использованием утилиты командной строки gh.

```
[1032192035_pfur.ru@1032192035 ~]$ gh auth login
? What account do you want to log into? GitHub.com
? What is your preferred protocol for Git operations? SSH
? Generate a new SSH key to add to your GitHub account? No
? How would you like to authenticate GitHub CLI? Login with a web browser

! First copy your one-time code: 70D0-5A0C
Press Enter to open github.com in your browser...
✓ Authentication complete.
- gh config set -h github.com git_protocol ssh
✓ Configured git protocol
✓ Logged in as Soblya
```

Рис. 6.6: Настройка gh

6.9 Создание репозитория курса на основе шаблона

В соответствии с заданием подготавливаем папки и создаем репозиторий курса на основе шаблона.

```
[1032192035 pfur.ru@1032192035 ~]$ mkdir -p ~/work/study/2022-2023/"Операционные системы"
[1032192035 pfur.ru@1032192035 ~]$ cd ~/work/study/2022-2023/Операционные\ системы/
[1032192035 pfur.ru@1032192035 Операционные системы]$ gh repo create study_2022-2023_os-intro --template
=yamadharm/course-directory-student-template --public
GraphQL: Could not resolve to a Repository with the name 'yamadharm/course-directory-student-template'.
(repository)
[1032192035 pfur.ru@1032192035 Операционные системы]$ gh repo create study_2022-2023_os-intro --template
=yamadharm/course-directory-student-template --public
/ Created repository Soblya/study_2022-2023_os-intro on GitHub
```

Рис. 6.7: Создание репозитория

6.10 Настраиваем каталог курса, создаем необходимые каталоги, отправляем результат на сервер

```
[1032192035 pfur.ru@1032192035 infosec]$ make COURSE=infosec
[1032192035 pfur.ru@1032192035 infosec]$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    package.json

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        labs/
        prepare

no changes added to commit (use "git add" and/or "git commit -a")
[1032192035 pfur.ru@1032192035 infosec]$ git add .
[1032192035 pfur.ru@1032192035 infosec]$ git commit -am 'feat(main): make course
structure'
[1032192035 pfur.ru@1032192035 infosec]$ git push
Enumerating objects: 20, done
```

Рис. 6.8: Настройка репозитория курса

7 Контрольные вопросы (часть 2)

7.1 Что такое системы контроля версий (VCS) и для решения каких задач они предназначены?

Системы контроля версий – это специализированное программное обеспечение используемое для хранения и контроля за изменениями исходных кодов. Позволяет оптимизировать работу команды, значительно сократить объём занимаемого кодом пространства.

7.2 Объясните следующие понятия VCS и их отношения:

7.2.1 Хранилище

Хранилище предназначено для хранения(sic!) исходных кодов и различных ресурсов проекта. В централизованных VCS хранилище находится на главном сервере, а пользователь копирует себе некоторую часть данных из хранилища, так же называемую рабочей копией, после внесения и фиксации изменений пользователь отправляет данные обратно на сервер. В децентрализованных, таких как git, при стандартной конфигурации каждый пользователь имеет полную копию хранилища и метаданных, при этом централизованное хранилище отсутствует, а такие порталы как GitHub хранят такую же копию хранилища как и все пользователи.

7.2.2 Commit

Commit(коммит) это снимок некоторого состояния хранилища в определённый момент времени.

7.2.3 История

История – это граф всех находящихся в репозитории снимков (коммитов)

7.2.4 Рабочая копия

Файлы из репозитория, находящиеся в локальной директории.

7.3 Что представляют собой и чем отличаются централизованные и децентрализованные VCS

В централизованных VCS полная копия репозитория находится на одном главном сервере, пользователи загружают данные для определённого состояния. В децентрализованных VCS главный сервер отсутствует, каждый пользователь имеет полную локальную копию хранилища, истории коммитов и т.д. Самым ярким примером централизованной VCS можно назвать Subversion(SVN), децентрализованной – git.

7.4 Опишите действия с VCS при единоличной работе с хранилищем.

- Создаём репозиторий `git init && git add . && git commit -m "initial commit"`
- Производим изменения в коде

- Фиксируем изменения `git add . && git commit -m "Commit message"`

7.5 Опишите порядок работы с общим хранилищем VCS.

- Клонировем репозиторий из удалённого источника `git clone <url>`
- Переходим в папку, в которой содержится склонированный репозиторий
- При необходимости создаём новую ветку и переходим в неё `git branch <name> && git checkout <name>`
- Вносим изменения в код
- Фиксируем изменения `git add . && git commit -m "Commit message"`
- Отправляем изменения в общее хранилище `git push`

7.6 Каковы основные задачи, решаемые инструментальным средством git?

Управление локальным репозиторием, создание коммитов, отправка их в общее хранилище.

7.7 Назовите и дайте краткую характеристику командам git.

- `git init` инициализирует репозиторий в каталоге
- `git clone` позволяет клонировать удалённый репозиторий
- `git add` позволяет пометить файлы для добавления в коммит
- `git commit` зафиксировать изменения

- `git pull` позволяет получить последние изменения из удалённого репозитория
- `git push` позволяет отправить локальные изменения в удалённый репозиторий

7.8 Приведите примеры использования при работе с локальным и удалённым репозиториями.

Примеры приведены в ответах на вопросы 4 и 5.

7.9 Что такое и зачем могут быть нужны ветви (branches)?

Ветви нужны для организации комфортной работы в репозитории. Хорошим тоном является вести работу над каждым тикетом в отдельной ветке. Так же, зачастую, производится разделение на стабильную ветвь(`master/main`) в которой находится стабильный, работоспособный код, готовый для распространения/установки у клиентов, ветвь текущей разработки (`develop/future`), в которую сливаются изменения из других ветвей при внедрении нового функционала/фиксе багов.

7.10 Как и зачем можно игнорировать некоторые файлы при commit?

Игнорируемые файлы в `git` перечисляются в файле `.gitignore`. Это необходимо для того, что бы в репозиторий не попадали `user-specific` файлы, такие как файлы настроек локального окружения, бинарные файлы создаваемые при сборке и т.п.

8 Выводы

По итогу выполненной лабораторной работы я научился устанавливать операционную систему, с использованием виртуальной машины. Так-же я ознакомился с работой в markdown.

Приобрел практические навыки установки операционной системы на виртуальную машин и настройки минимально необходимых для дальнейшей работы сервисов.

Изучил идеологию и применение средств контроля версий.

Освоил умения по работе с git.

Научился оформлять отчёты с помощью легковесного языка разметки Markdown.

Список литературы

1. Rocky Linux // Rocky Linux Official site URL: <https://rockylinux.org/ru/> (дата обращения: 08.09.2022).
2. Microsoft // GitHub URL: <https://github.com/> (дата обращения: 08.09.2022).
3. wereturtle // ghostwriter Official site URL: <https://wereturtle.github.io/ghostwriter/> (дата обращения: 08.09.2022).
4. Software Freedom Conservancy, Inc. // git - Book URL: <https://git-scm.com/book/en/v2/> (дата обращения: 08.09.2022).