Отчёт по лабораторной работе 1

Соболев Максим Сергеевич

Содержание

Список иллюстраций

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.

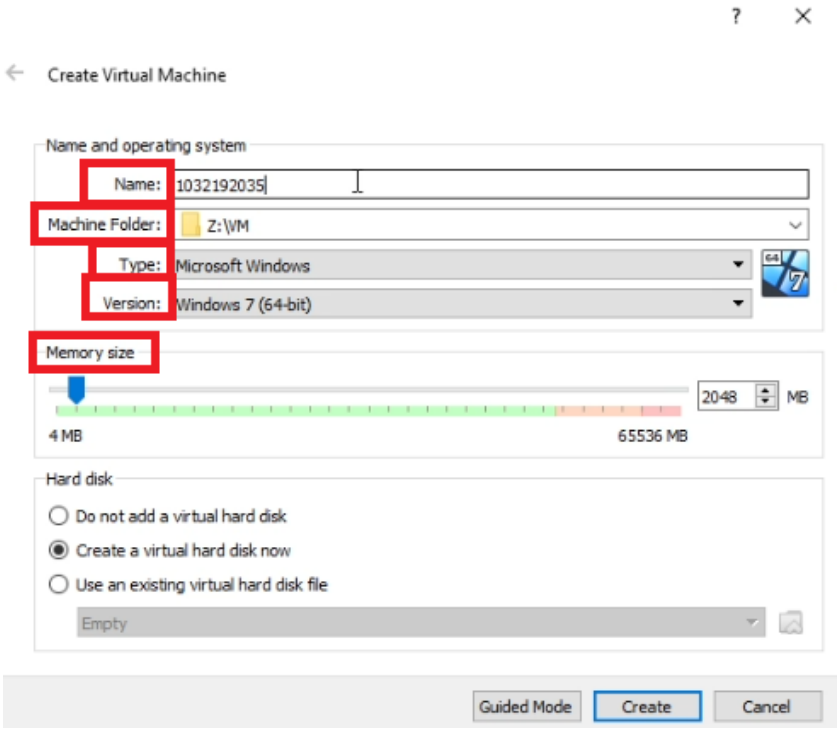

Рис. 4.1: Создание виртуальной машины

4.1.1 Настройка виртуального жесткого диска

Здесь указываем объем виртуального жесткого диска, место его хранения.

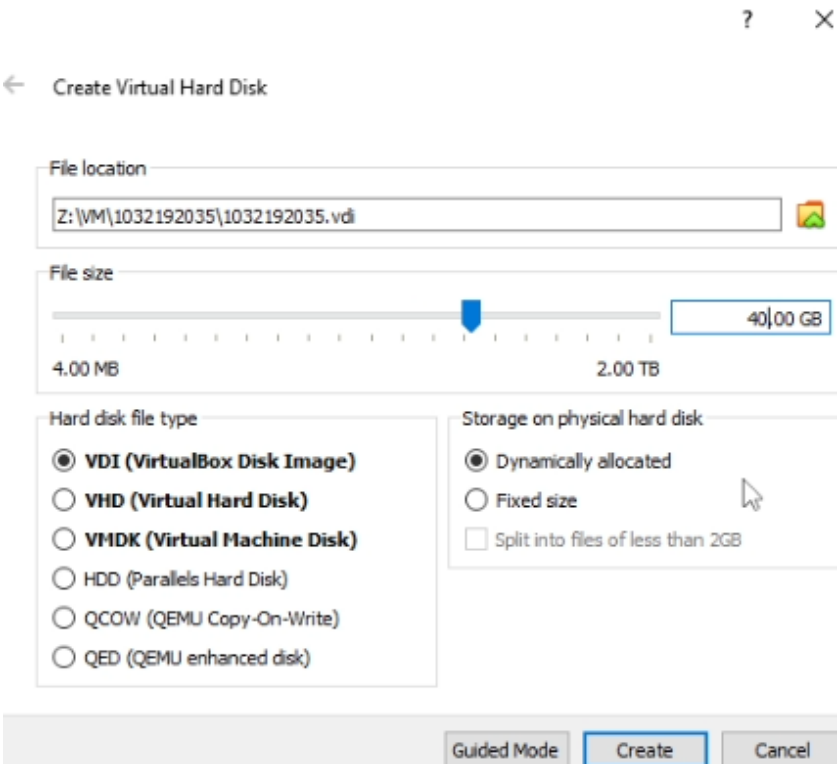

Рис. 4.2: Настройка виртуального ЖД

4.1.2 Настройка виртуального процессора

Добавляем виртуальные ядра, что бы ВМ шевелилась быстрее.

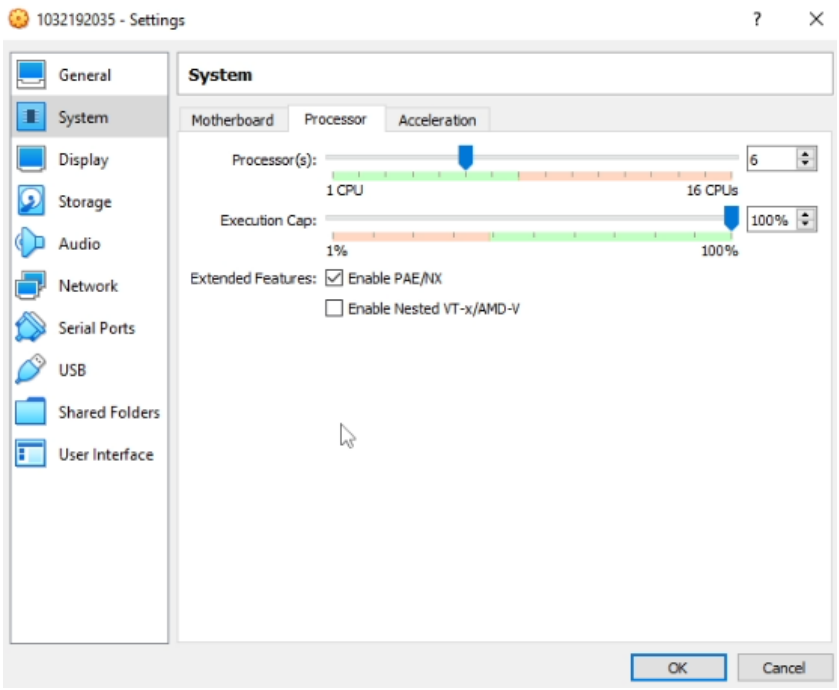

Рис. 4.3: Настройка виртуального ЦП

4.1.3 Запуск виртуальной машины

Нажимаем кнопку “Start”, виртуальная машина запускается.

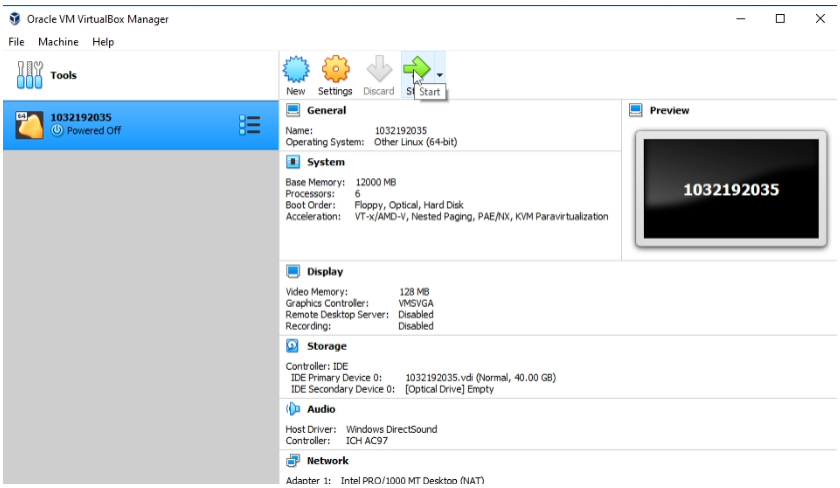

Рис. 4.4: Кнопка запуска ВМ

4.2 Начало установки

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

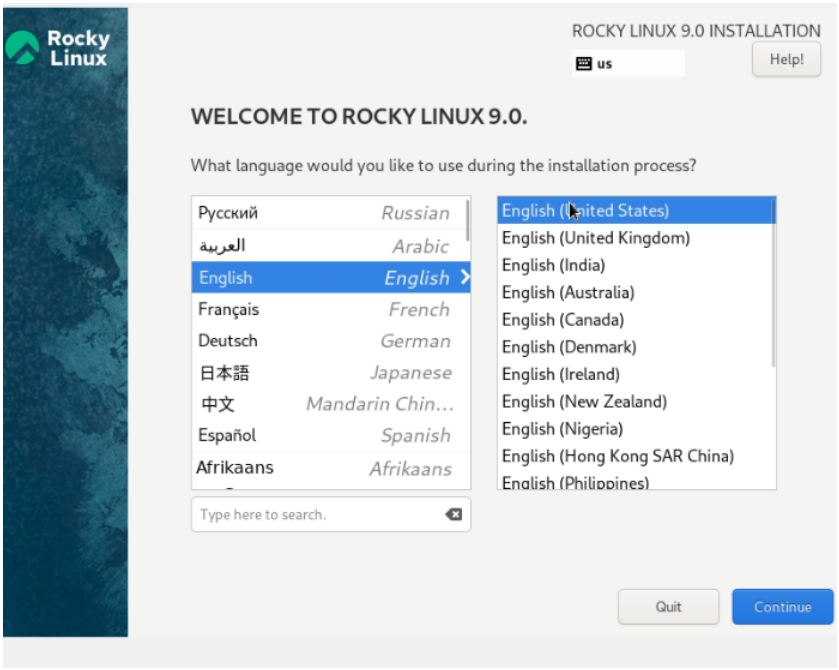

Рис. 4.5: Название рисунка

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.

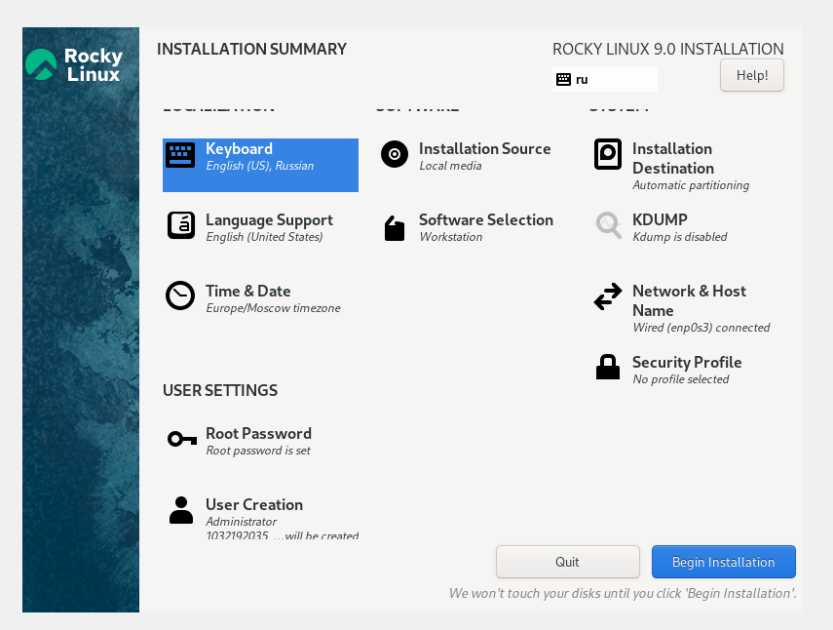

Рис. 4.6: Сведения об установке

Настраиваем место установки, время, параметры раскладки клавиатуры и т.д.

Данные пользователя в соответствии с соглашением:

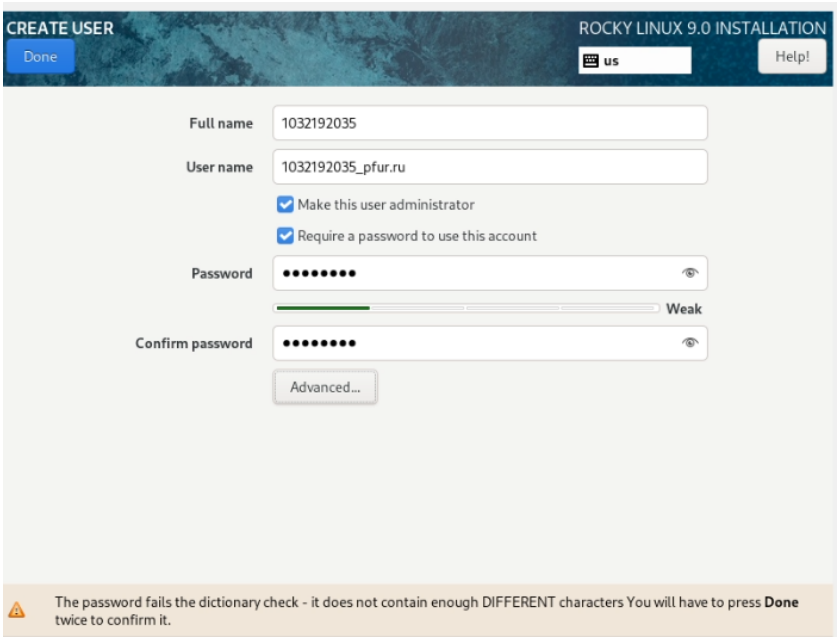

Рис. 4.7: Окно создания пользователя

4.4 Наблюдаем за процессом установки

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

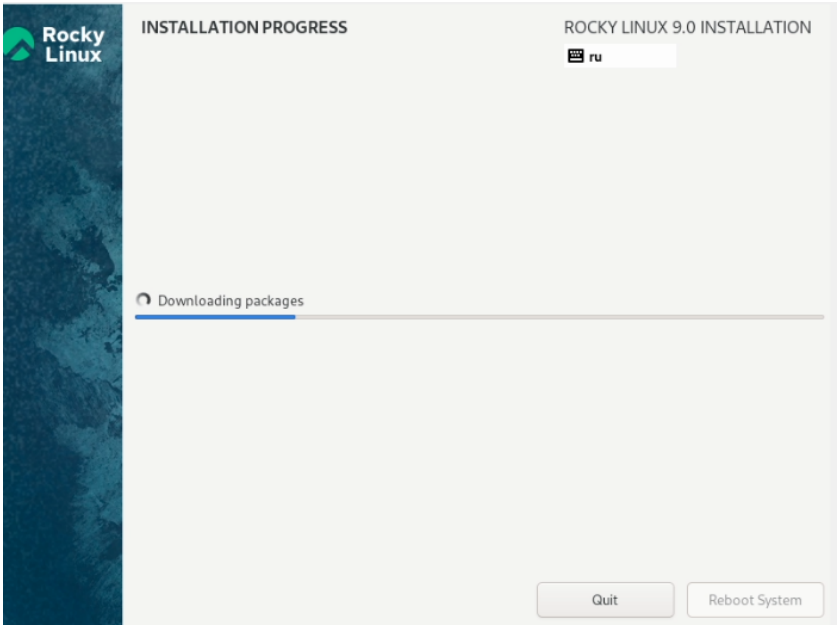

Рис. 4.8: Синяя полоса прогресса установки

4.5 После окончания процесса установки

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

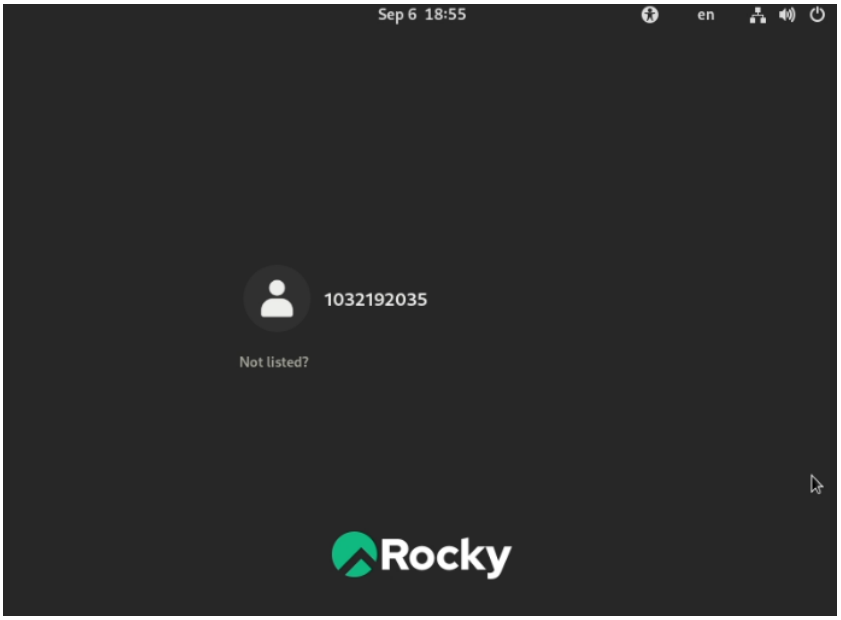

Рис. 4.9: Окончание процесса установки

4.6 Вывод dmesg | less

Начинаем выполнять задания.

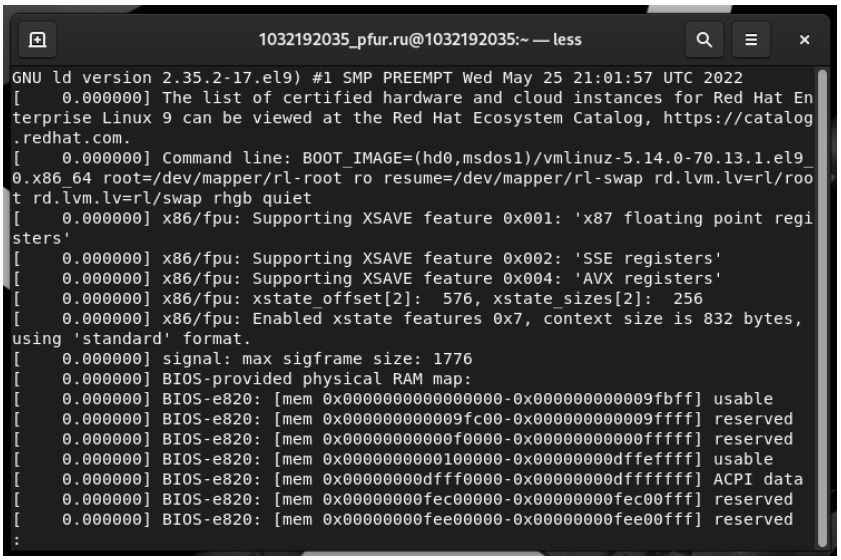

Рис. 4.10: dmesg | less

4.7 Версия ядра Linux

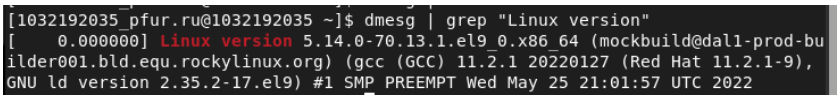

Рис. 4.11: Версия ядра Linux

4.8 Сведения о частоте и модели процессора

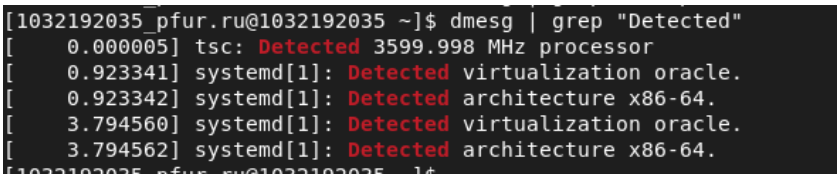

Рис. 4.12: Частота процессора

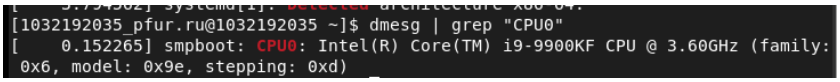

Рис. 4.13: Модель процессора

4.9 Сведения о доступной ОЗУ

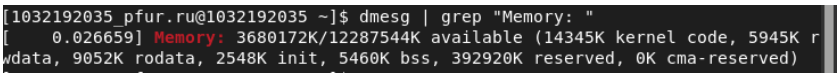

Рис. 4.14: Доступная ОЗУ

4.10 Сведения об используемом гипервизоре

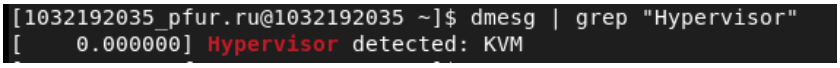

Рис. 4.15: Используемый гипервизор

4.11 Сведения о типе файловой системы корневого раздела

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

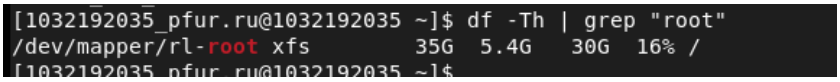

Рис. 4.16: Файловая система корневого раздела

4.12 Последовательность монтирования файловых систем

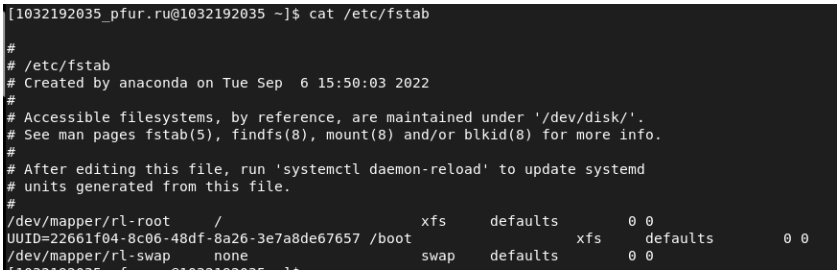

Рис. 4.17: Последовательность монтирования ФС

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 созданий. 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.

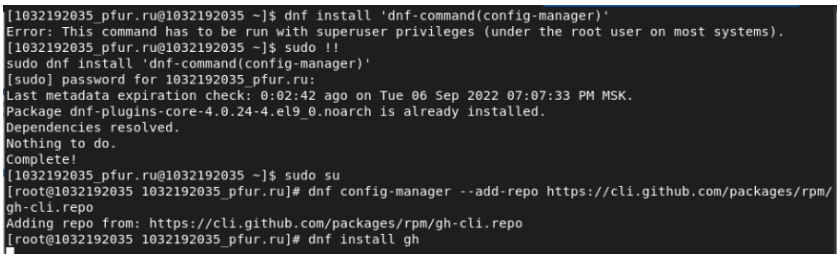

Рис. 6.1: Установка gh

6.4 Базовая настройка git

Вводим свои данные. Настраиваем в соответствие 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:WuVEsIvGiOpcUSAIDRfE3VVqsvczignWYcZTjOpoXas root@1032192035.local
The key's randomart image is:
+--[ED25519 256]--+
|=*++ . .ooo      |
|..+ 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.

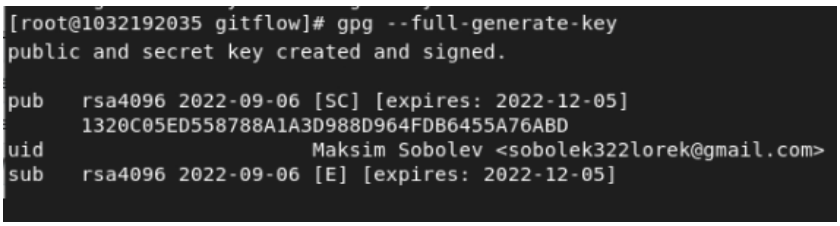

Рис. 6.4: Создание ключей gpg

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.

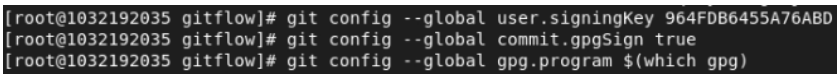

Рис. 6.5: Настройка автоматических подписей коммитов git

6.8 Настройка gh

Логинимся в GitHub с использованием утилиты командной строки gh.

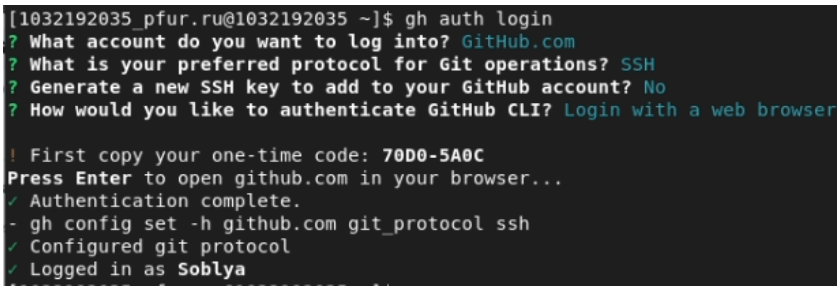

Рис. 6.6: Настройка gh

6.9 Создание репозитория курса на основе шаблона

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

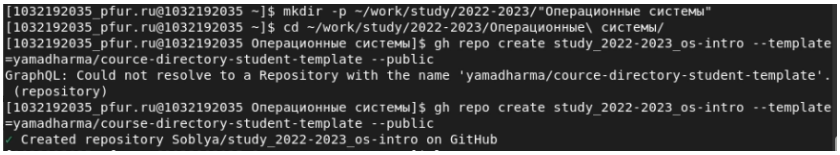

Рис. 6.7: Создание репозитория

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

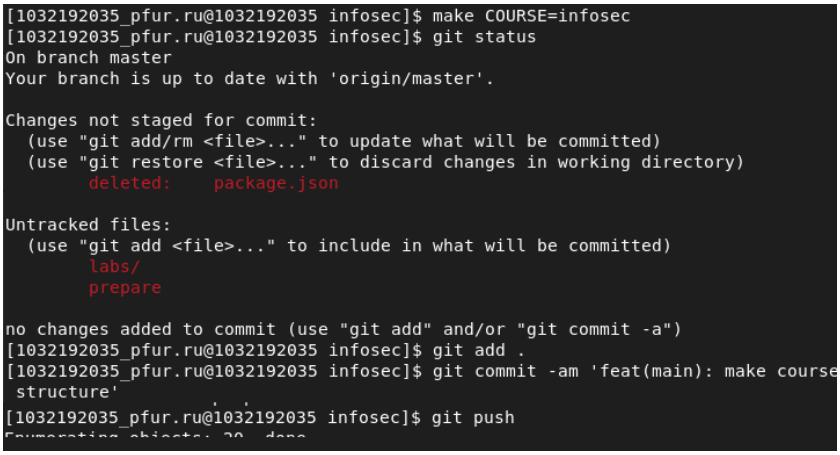

Рис. 6.8: Настройка репозитория курса

7 Контрольные вопросы (часть 2)

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

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.

- Клонируем 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 склонировать 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 Offlcial 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 Offlcial 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).