Отчёт по лабораторной работе №2

Операционные системы

Калашникова Ольга Сергеевна

Содержание

# 1 Цель работы

Целью данной работы является изучитение идеологии и применение средств контроля версий, а так же освоение умения по работе с git.

# 2 Задание

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.

# 3 Выполнение лабораторной работы

## 3.1 Установка программного обеспечения

Установим git при помощи dnf install git, а так же установим gh при помощи dnf install gh (в моём случае програмное обспечение уже установлено, так как вышло не с первого раза) (рис. 1).

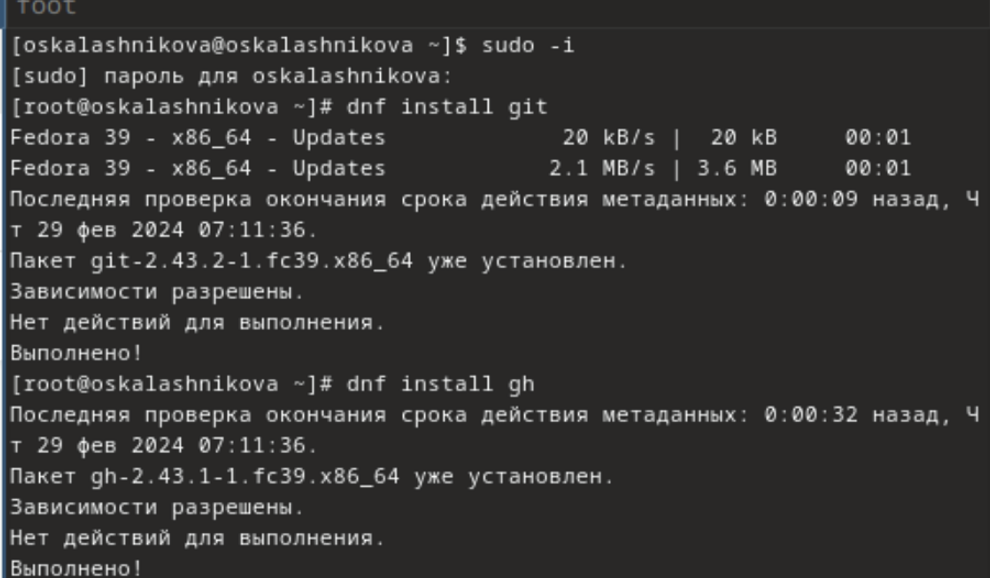


Рис. 1: Установка git и gh

## 3.2 Базовая настройка git

Зададим имя владельца репозитория при помощи git config –global user.name “lacrimell” и email при помощи git config –global user.email “lacrimell@yandex.by” (рис. 2).

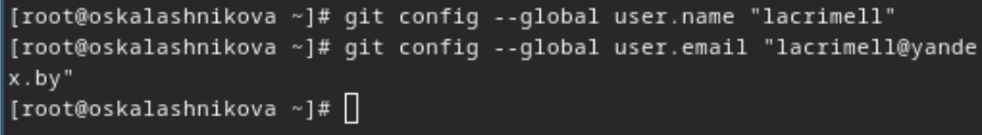


Рис. 2: Задаём имя и почту

Настроим utf-8 в выводе сообщений git при помощи git config –global core.quotepath false (рис. 3).

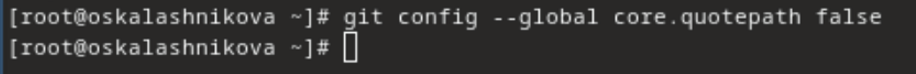


Рис. 3: utf-8

Зададим имя начальной ветки (будем называть её master) при помощи git config –global init.defaultBranch master, далее параметр autocrlf с помощью git config –global core.autocrlf input и параметр safecrlf при помощи git config –global core.safecrlf warn (рис. 4).

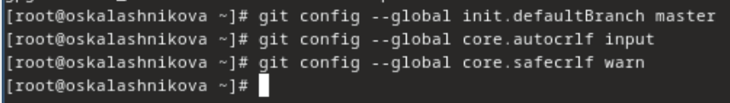


Рис. 4: Настройка верификации

## 3.3 Создайте ключи ssh

Gо алгоритму rsa с ключём размером 4096 бит создаём ключ ssh при помощи ssh-keygen -t rsa -b 4096, а по алгоритму ed25519 с ssh-keygen -t ed25519 (рис. 5).

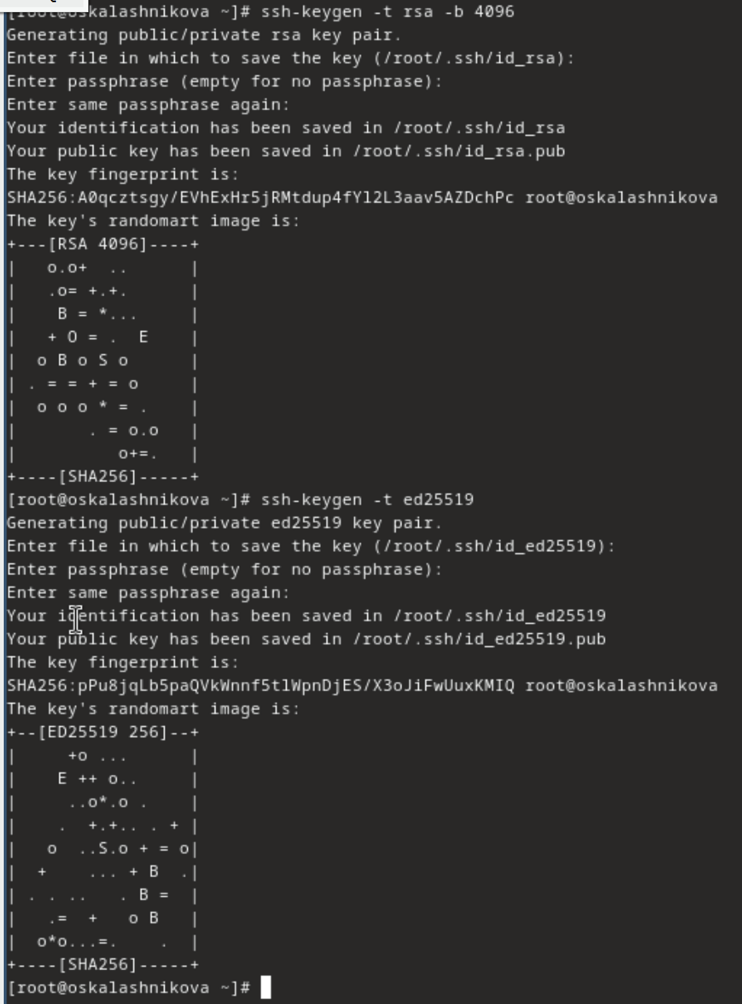


Рис. 5: Создание ключей ssh

## 3.4 Создайте ключи pgp

Генерируем ключ при помощи gpg –full-generate-key и выбираем тип RSA and RSA, размер 4096,срок действия не истекает никогда (рис. 6).

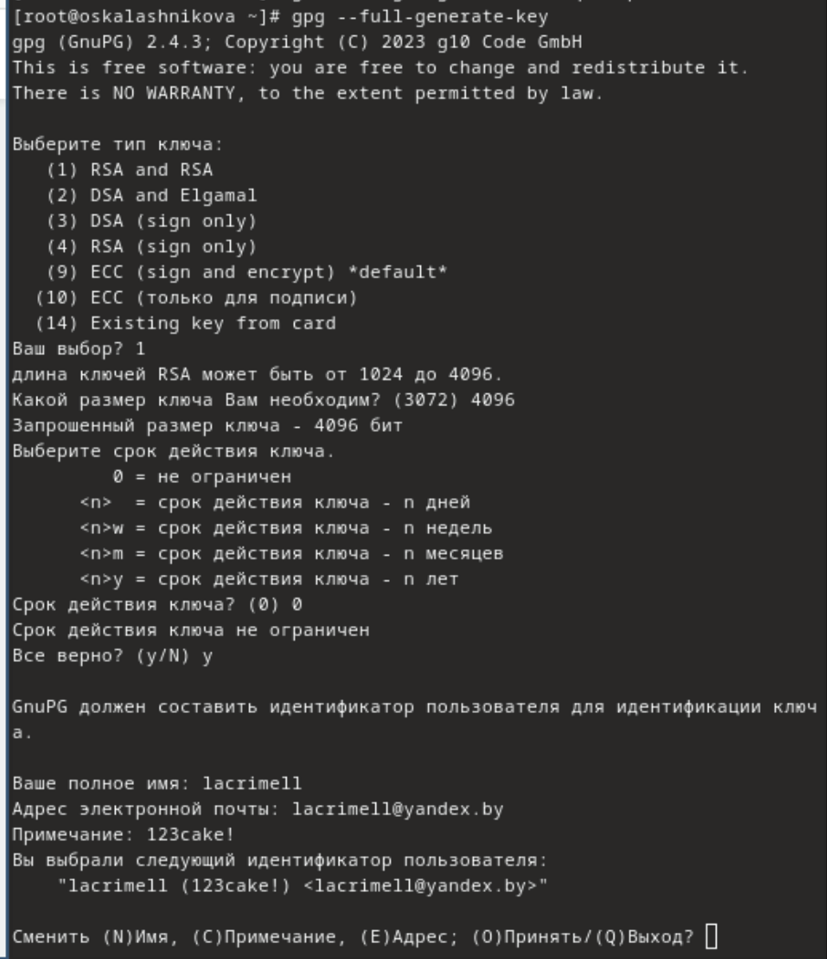


Рис. 6: Генерация ключа pgp

Вводим имя, адрес почты используемый на GitHub.Я так же ввела комментарий, чтобы не забыть пароль (рис. 7).

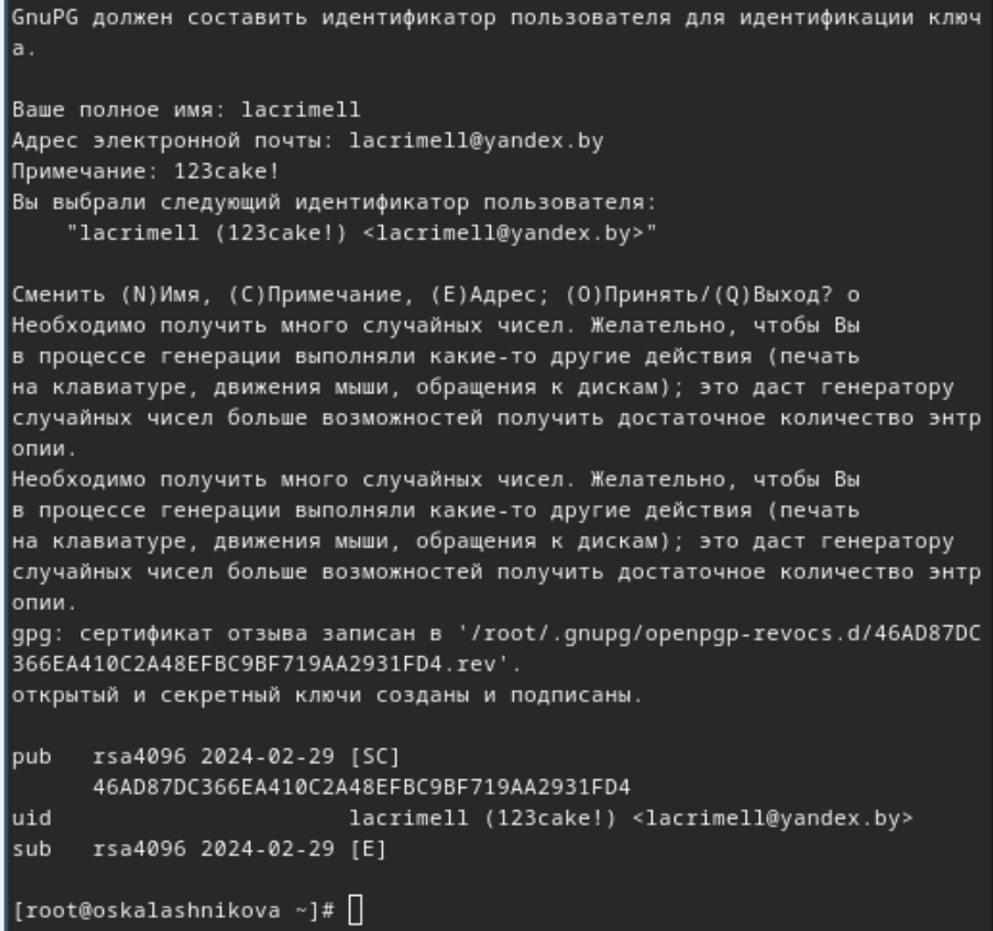


Рис. 7: Генерация ключа pgp 2

## 3.5 Добавление PGP ключа в GitHub

У меня уже была создана учетная запись и заполнены основные данные на https://github.com, так что я перешла сразу к выводу список ключей при помощи gpg –list-secret-keys –keyid-format LONG (рис. 8).

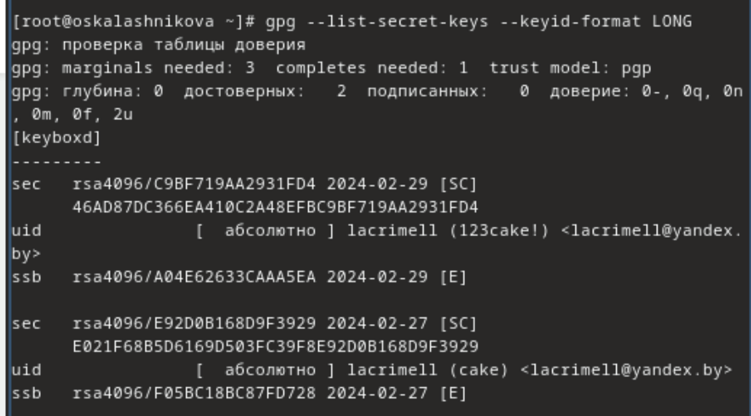


Рис. 8: Список ключей

Далее мы должны скопируйте сгенерированный PGP ключ в буфер обмена при помощи gpg –armor –export | xclip -sel clip, но данная команда у меня не сработала и я копировала вручную (рис. 9).

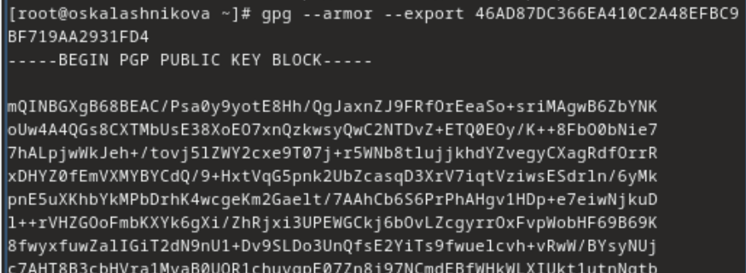


Рис. 9: Копирование ключа

Переходим в настройки GitHub (https://github.com/settings/keys), нажимаем на кнопку New GPG key и вставляем полученный ключ в поле ввода (рис. 10).

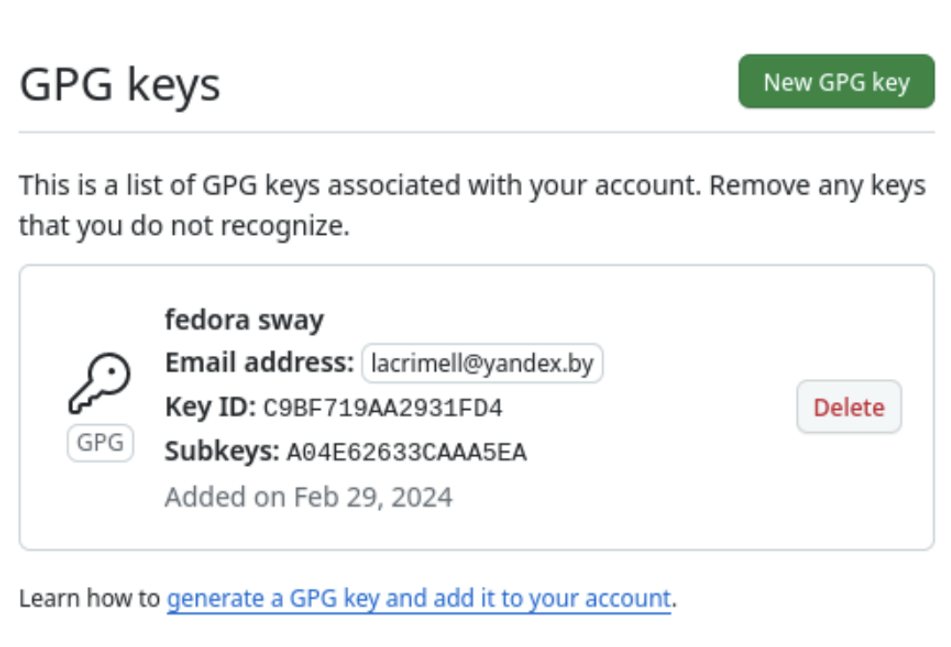


Рис. 10: Готовый ключ

##Настройка автоматических подписей коммитов git

Используя введёный email, указываем Git применять его при подписи коммитов с помощью git config –global user.signingkey , git config –global commit.gpgsign true,git config –global gpg.program $(which gpg2) (рис. 11).

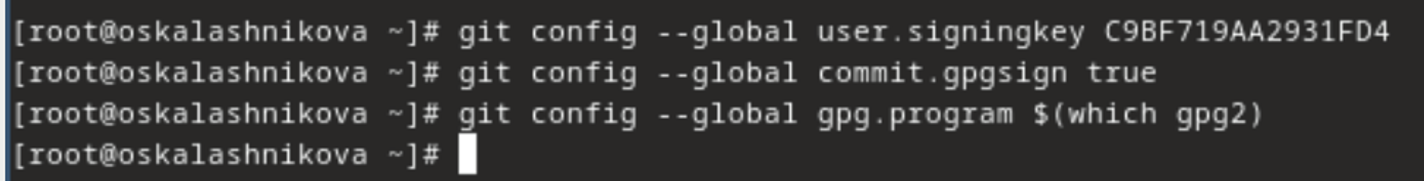


Рис. 11: Подпись коммитов

## 3.6 Настройка gh

Авторизируемся при помощи gh auth login. Утилита задаёт несколько наводящих вопросов, после авторизируемся через браузер (рис. 12).

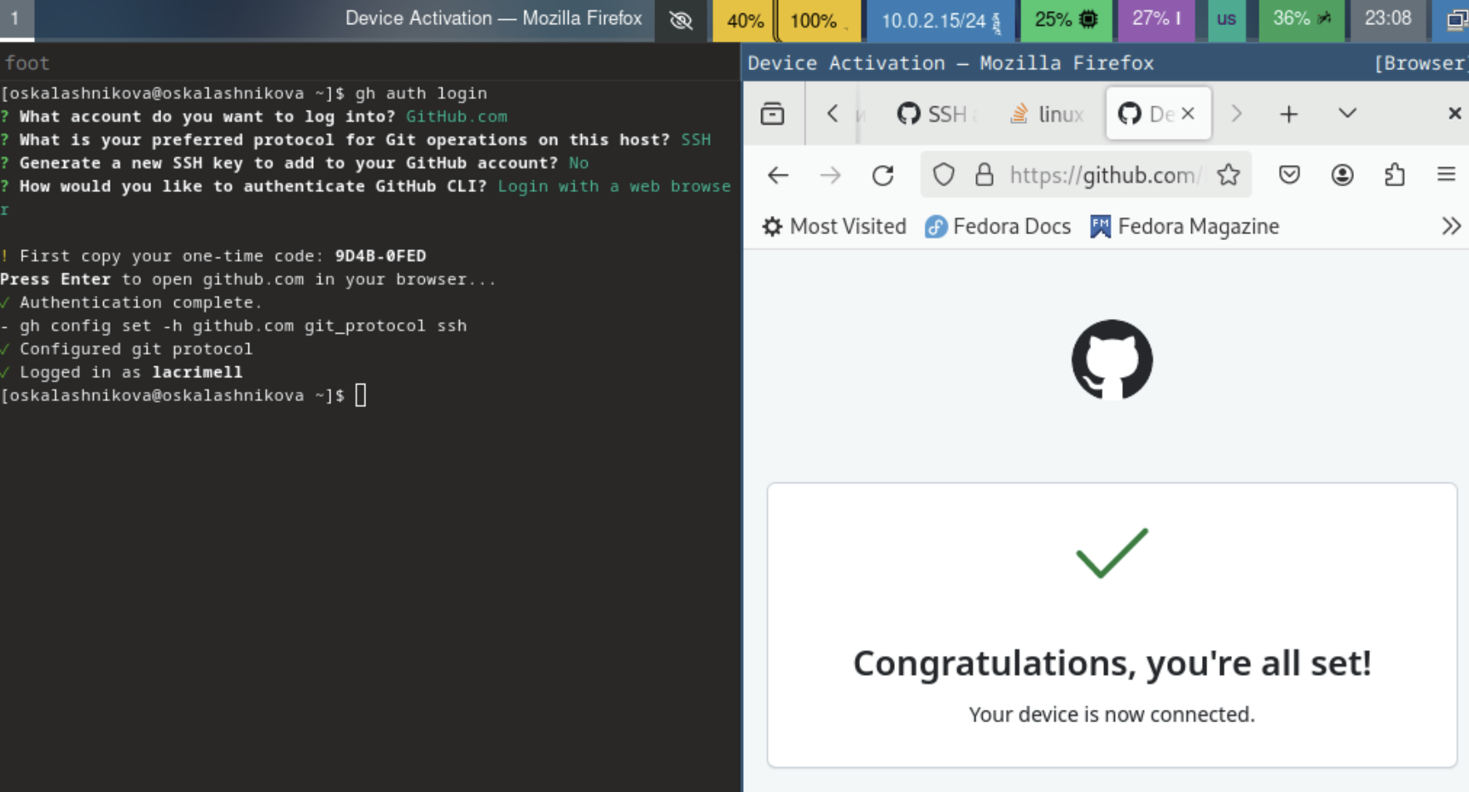


Рис. 12: Авторизация

## 3.7 Сознание репозитория курса на основе шаблона

Cоздаём папку при помощи mkdir -p ~/work/study/2023-2024/“Операционные системы” и переходим в неё cd ~/work/study/2023-2024/“Операционные системы” (рис. 13).

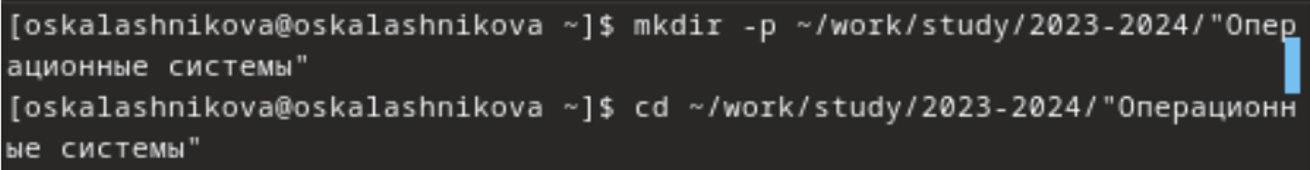


Рис. 13: Создание папки

Создаём репозиторий gh repo create study\_2023-2024\_os-intro –template=yamadharma/course-directory-student-template –public (рис. 14).

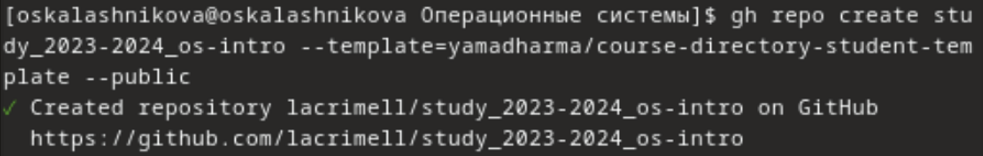


Рис. 14: Создание репозитория

Клонируем его на виртуальную машину git clone –recursive git@github.com:oskalashnikova/study\_2023-2024\_os-intro.git os-intro (рис. 15).

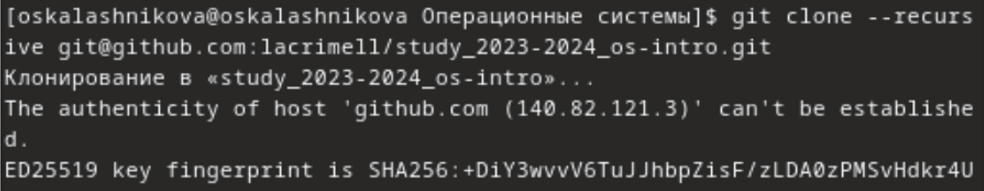


Рис. 15: Клонирование

Проверяем с помощью mc (рис. 16).

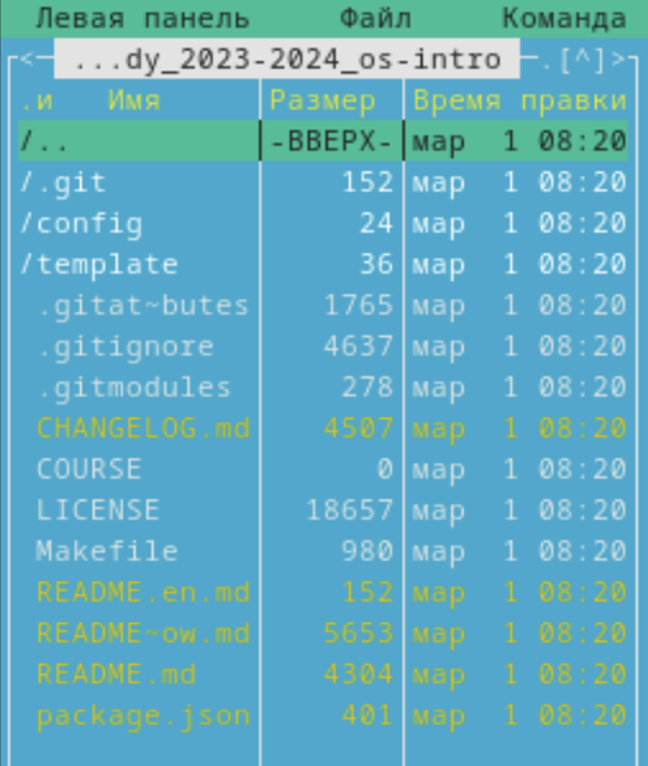


Рис. 16: Проверка

## 3.8 Настройка каталога курса

Переходим в каталог курса при помощи cd ~/work/study/2022-2023/“Операционные системы”/os-intro и удаляем лишние файлы с помощью rm package.json (рис. 17).

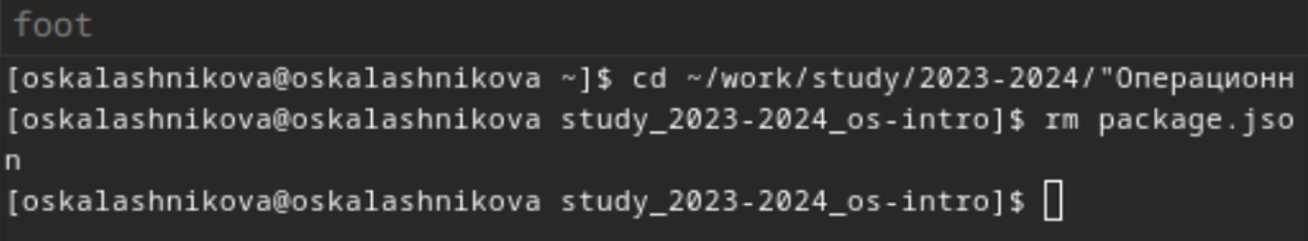


Рис. 17: Удаление файла

Проверяем, удалился ли файл (рис. 18).

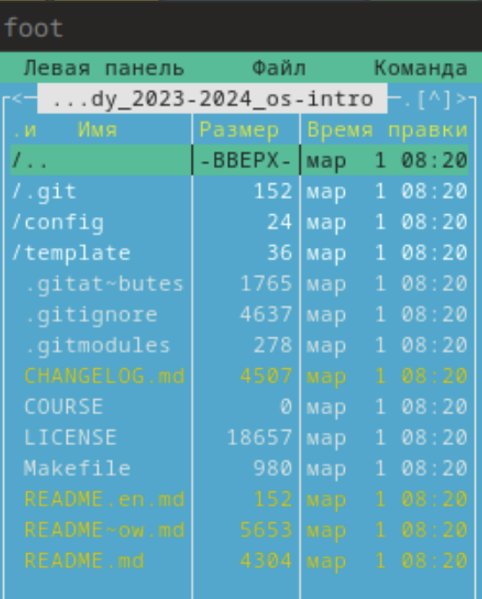


Рис. 18: Проверка

Создаём необходимые каталоги при помощи echo os-intro > COURSE (рис. 19).

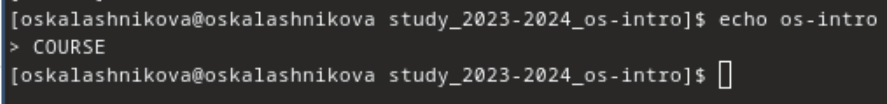


Рис. 19: Создание

Проверяем создался ли (рис. 20).

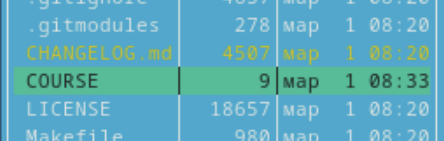


Рис. 20: Проверка

Используем make (рис. 21).

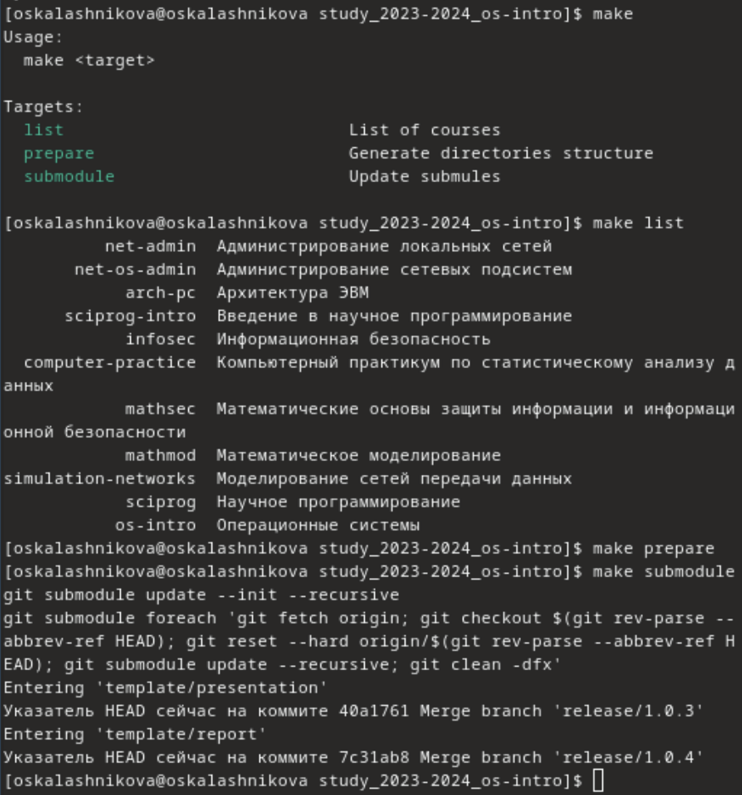


Рис. 21: make

Отправляем файлы на сервер (рис. 22), (рис. 23).

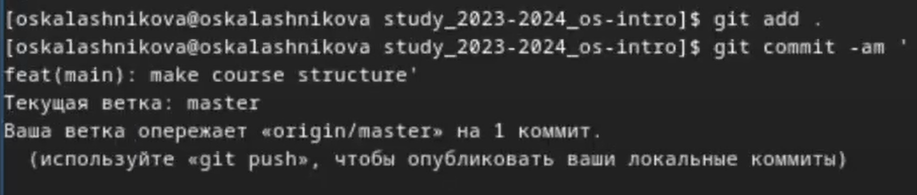


Рис. 22: git add . , git commit

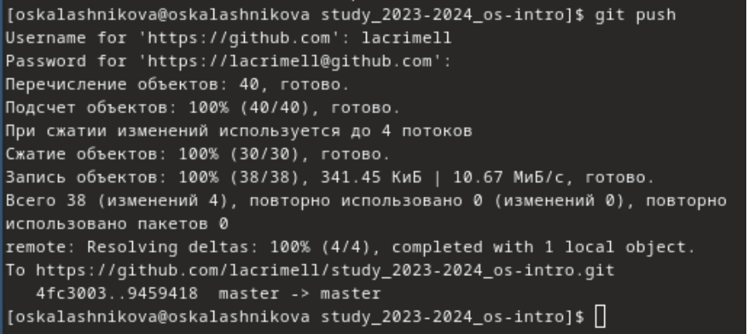


Рис. 23: git push

# 4 Контрольные вопросы

1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются? Системы контроля версий (VCS) позволяют хранить проекты, отслеживать изменения, управлять релизами и контролировать версии проектов. Они могут использоваться для индивидуальной и командной работы, обеспечивая контроль над изменениями и возможность отката к предыдущим версиям.
2. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия.

* Хранилище: это репозиторий, где хранятся все файлы и версии проекта.
  + Commit: это фиксация изменений в репозитории.
  + История: представляет собой запись изменений файлов проекта.
  + Рабочая копия: это копия репозитория, с которой работает разработчик.

1. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида. Централизованные VCS имеют один центральный репозиторий, в то время как децентрализованные VCS позволяют использовать несколько репозиториев и объединять изменения между ними. Примеры: CVS (централизованная) и Git (децентрализованная).
2. Опишите действия с VCS при единоличной работе с хранилищем. Создается репозиторий, в котором разрабатывается проект. После внесения изменений файлы добавляются в репозиторий.
3. Опишите порядок работы с общим хранилищем VCS. Разработчик клонирует репозиторий, вносит изменения, фиксирует их, и выгружает на сервер. Разработчики могут объединять свои версии в общем репозитории.
4. Каковы основные задачи, решаемые инструментальным средством git? Git предназначен для хранения файлов проекта, контроля версий, обеспечения командной работы и отслеживания изменений.
5. Назовите и дайте краткую характеристику командам git.

* git clone: клонирует проект с сервера на компьютер.
* git add: добавляет файлы для выгрузки на сервер.
* git commit: фиксирует изменения репозитория.
* git push: выгружает изменения на сервер.
* git pull: получает изменения с сервера.
* git rm: удаляет файл.
* git status: отображает статус репозитория.

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

* Локальный: git commit -am “added files” — создает коммит.
* Удаленный: git push — отправляет данные на удаленный сервер.

1. Что такое и зачем могут быть нужны ветви (branches)? Как и зачем можно игнорировать некоторые файлы при commit?

Ветви (branches) представляют собой независимые линии разработки, позволяющие параллельно вносить изменения в проект, которые могут быть объединены позднее.

1. Как и зачем можно игнорировать некоторые файлы при commit? Игнорировать файлы можно, добавив их в .gitignore. Это полезно, когда не нужно добавлять определенные файлы в репозиторий, например, файлы виртуального окружения (например, venv).

# 5 Выводы

Я изучила идеологию и применение средств контроля версий, а так же освоила умения по работе с git

# Список литературы

https://git-scm.com/book/ru/v2/Основы-Git-Работа-с-удалёнными-репозиториями

https://devpractice.ru/git-for-beginners-part-1-what-is-vcs/

https://blog.skillfactory.ru/glossary/git/

туис