Отчёт по лабораторной работе №2

Дисциплина: Операционные системы

Бызова Мария Олеговна

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Целью данной лабораторной работы является изучение идеологии и применение средств контроля версий, 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.

Уставливаю необходимое программное обеспечение: устанавливаю git (рис. 3.1).

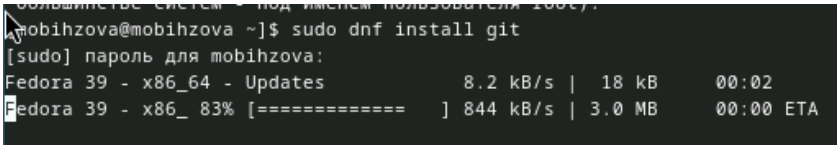

Рис. 3.1: Установка git

Устанавливаю gh (рис. 3.2).

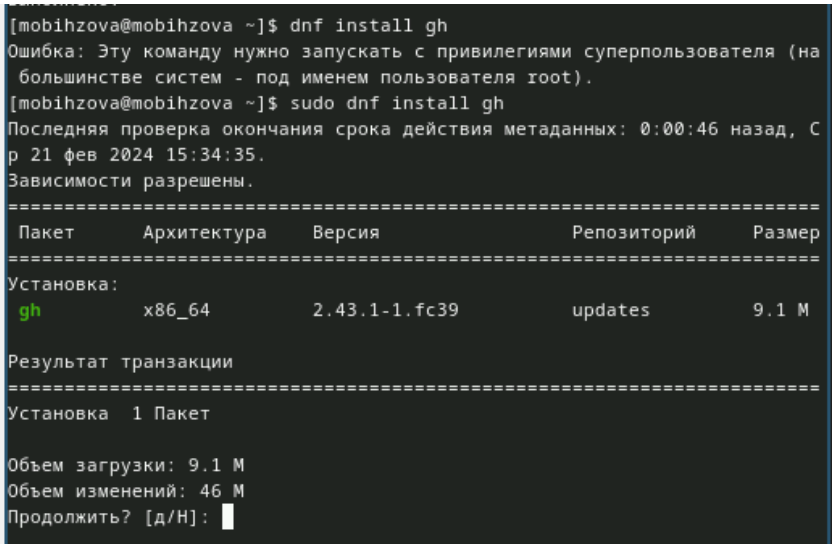

Рис. 3.2: Установка gh

3.2 Базовая настройка git.

Задаю имя и email владельца репозитория (рис. 3.3).

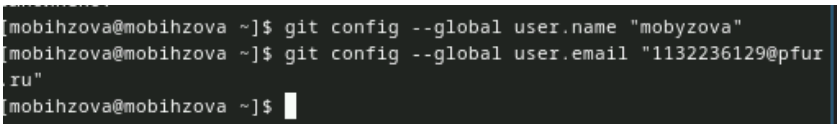

Рис. 3.3: Задаю имя и email владельца репозитория

Настраиваю utf-8 в выводе сообщений git (рис. 3.4).

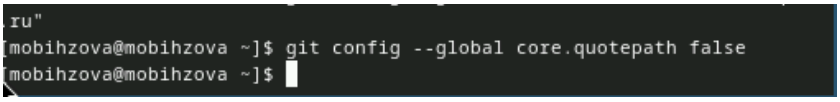

Рис. 3.4: Настройка utf-8 в выводе сообщений git

Зададаю имя начальной ветки (будем называть её master) (рис. 3.5).

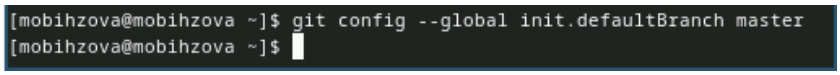

Рис. 3.5: Задаю имя начальной ветки

Задаю параметр autocrlf (рис. 3.6).

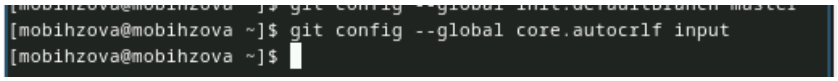

Рис. 3.6: Задаю параметр autocrlf

Задаю параметр safecrlf (рис. 3.7).

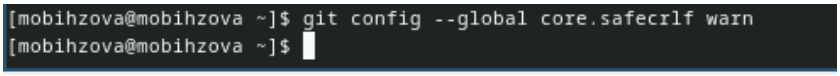

Рис. 3.7: Задаю параметр safecrlf

3.3 Создание ключа ssh.

Создаю ключ ssh по алгоритму rsa с ключём размером 4096 бит (рис. 3.8).

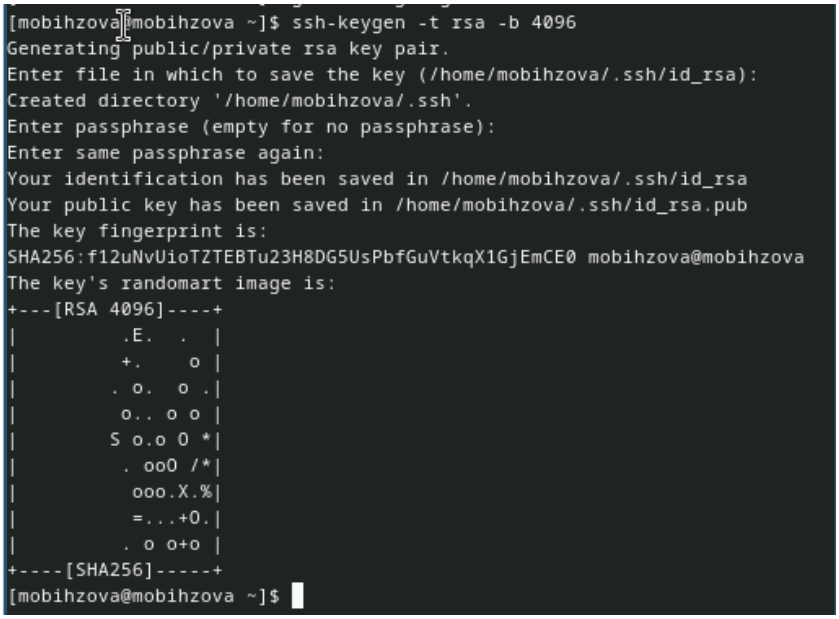

Рис. 3.8: Генерация ключа ssh по алгоритму rsa

Создаю ключ ssh по алгоритму ed25519 (рис. 3.9).

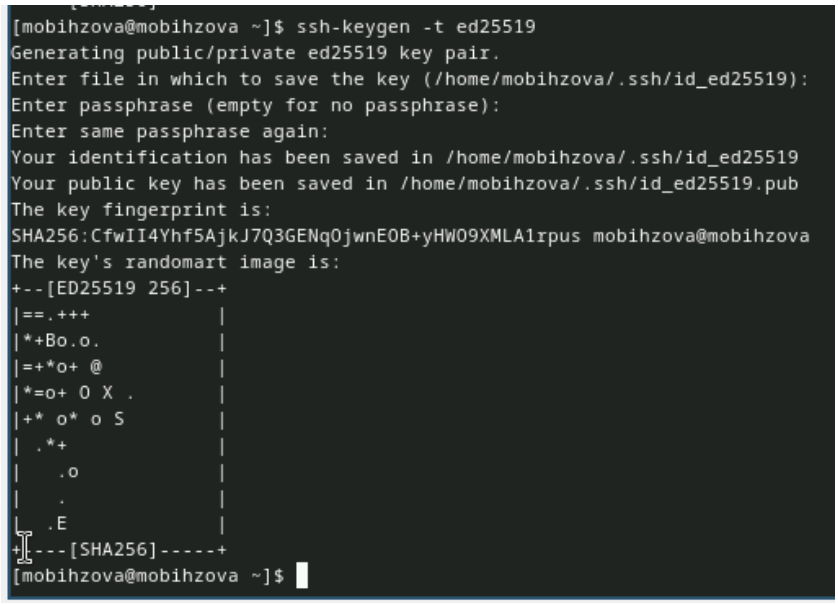

Рис. 3.9: Генерация ключа ssh по алгоритму ed25519

3.4 Создание ключа PGP.

Генерирую ключ PGP, затем выбираю тип ключа RSA and RSA, задаю максимальную длину ключа: 4096, оставляю неограниченный срок действия ключа. Далее отвечаю на вопросы программы о личной информации (рис. 3.10).

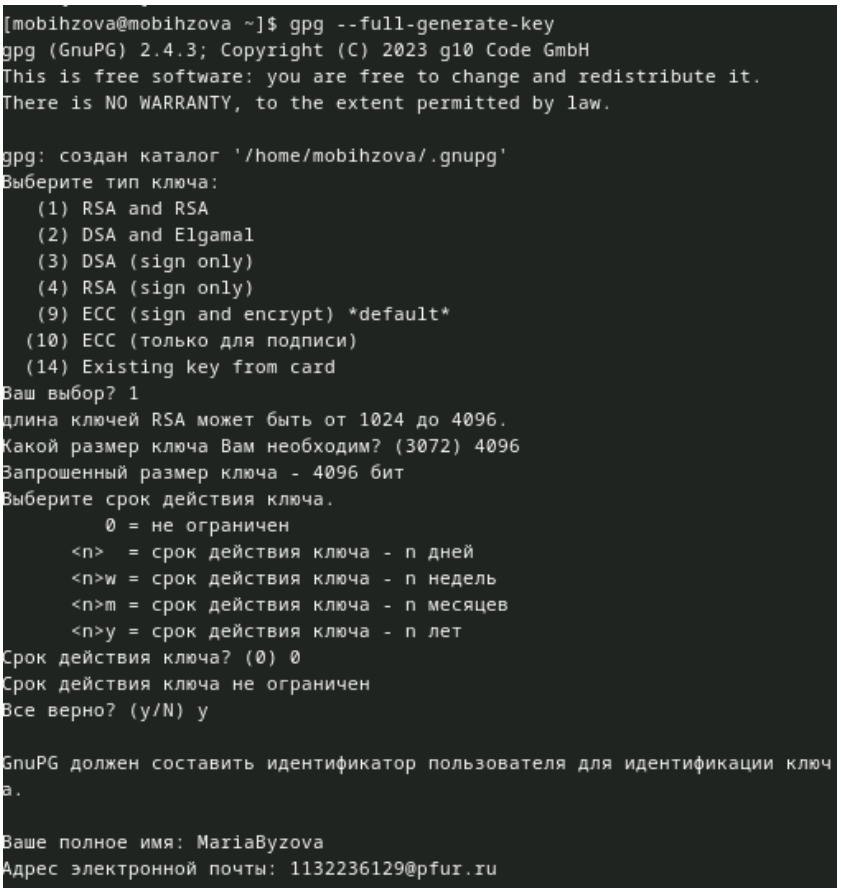

Рис. 3.10: Генерация ключа

3.5 Настройка github.

У меня уже был создан аккаунт на Github, соответсвенно, основные данные аккаунта я так же заполняла и проводила его настройку, поэтому просто вхожу в свой аккаунт (рис. 3.11).

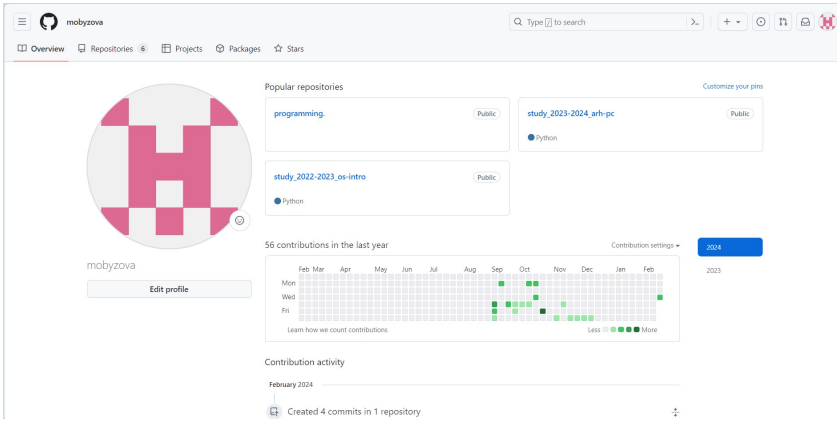

Рис. 3.11: Аккаунт Github

3.6 Добавление ключа PGP в Github

Вывожу список ключей и копирую отпечаток приватного ключ (рис. 3.12).

```
[mobihzova@mobihzova ~]$ gpg --list-secret-keys --keyid-format LONG
gpg: проверка таблицы доверия
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: глубина: 0  достоверных:   1  подписанных:   0  доверие: 0-, 0q, 0n
, 0m, 0f, 1u
[keyboxd]
---------
sec   rsa4096/94499505E2FAC668 2024-02-21 [SC]
      5C340D3AC5C7AA332BDFFE0B94499505E2FAC668
uid                 [  абсолютно ] MariaByzova <1132236129@pfur.ru>
ssb   rsa4096/87CE5F52F96B347D 2024-02-21 [E]

[mobihzova@mobihzova ~]$
```

Рис. 3.12: Вывод списка ключей

Скопирую мой сгенерированный PGP ключ в буфер обмена (рис. 3.13).

```
Выполнено!
[mobihzova@mobihzova ~]$ gpg --armor --export 94499505E2FAC668 | xclip -
sel clip
[mobihzova@mobihzova ~]$
```

Рис. 3.13: Копирование ключа в буфер обмена

Перейду в настройки GitHub, нажму на кнопку New GPG key и вставлю полученный ключ в поле ввода (рис. 3.14).

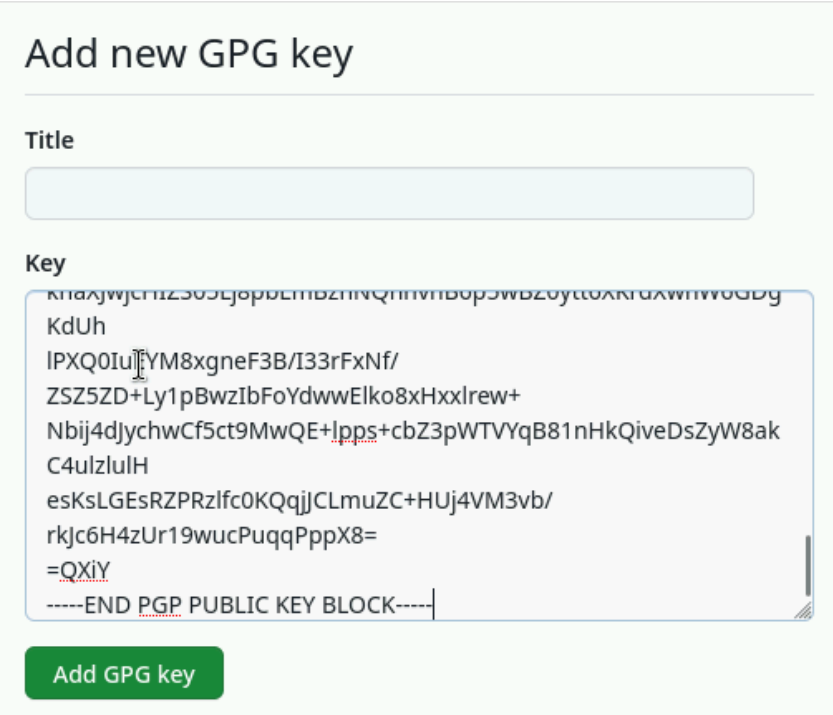

Рис. 3.14: Добавление нового ключа

3.7 Настройка автоматических подписей коммитов git.

Используя введёный email, укажу Git применять его при подписи коммитов (рис. 3.15).

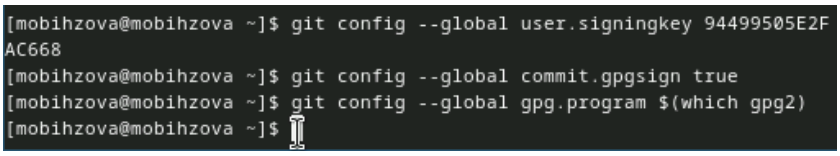

Рис. 3.15: Настройка подписей git

3.8 Настройка gh.

Для начала авторизируюсь: отвечаю на наводящие вопросы от утилиты, в конце выбираю авторизироваться через браузер (рис. 3.16).

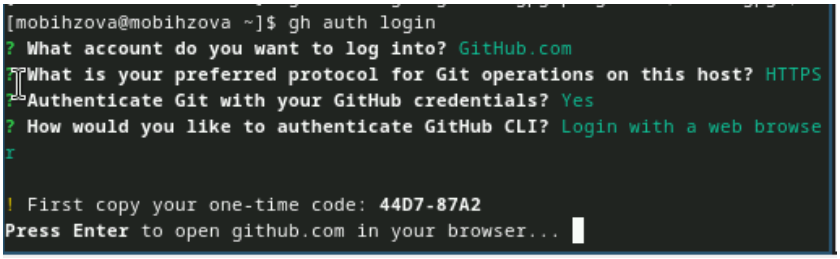

Рис. 3.16: Авторизация

Завершаю авторизацию на сайте (рис. 3.17).

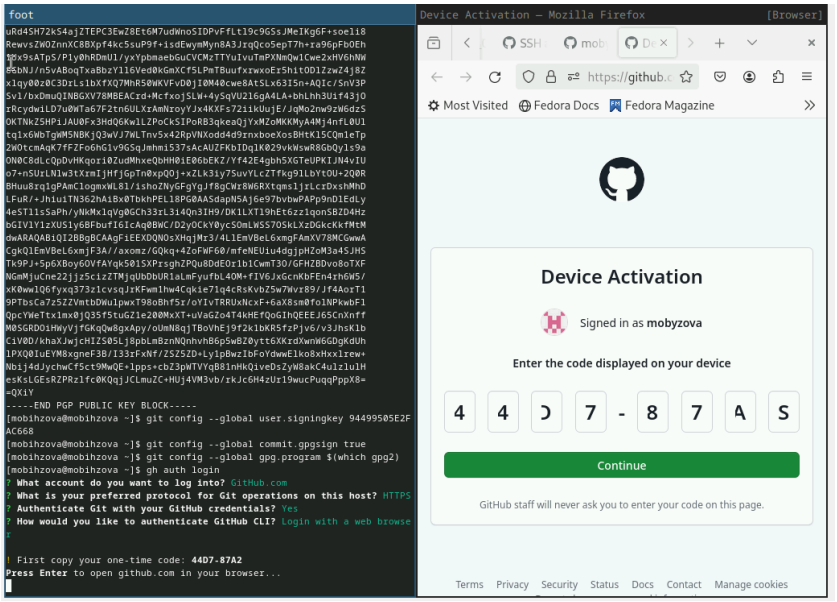

Рис. 3.17: Завершение авторизации

Авторизация прошла успешно (рис. 3.18).

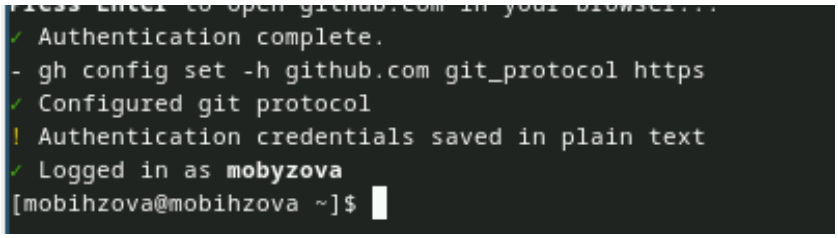

Рис. 3.18: Результат авторизации

3.9 Создание репозитория курса на основе шаблона.

Сначала создаю директорию с помощью утилиты mkdir и флага -p, который позволяет установить каталоги на всем указанном пути. После этого с помощью утилиты cd перехожу в только что созданную директорию (рис. 3.19).

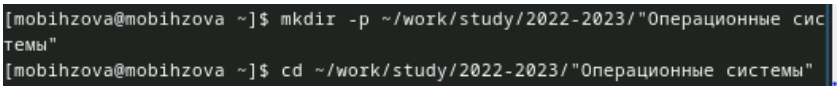

Рис. 3.19: Создание и перемещение между директориями

Далее создаю репозиторий на основе шаблона репозитория (рис. 3.20).

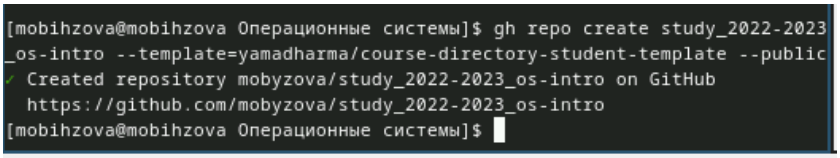

Рис. 3.20: Создание репозитория на основе шаблона репозитория

Клонирую репозиторий (рис. 3.21).

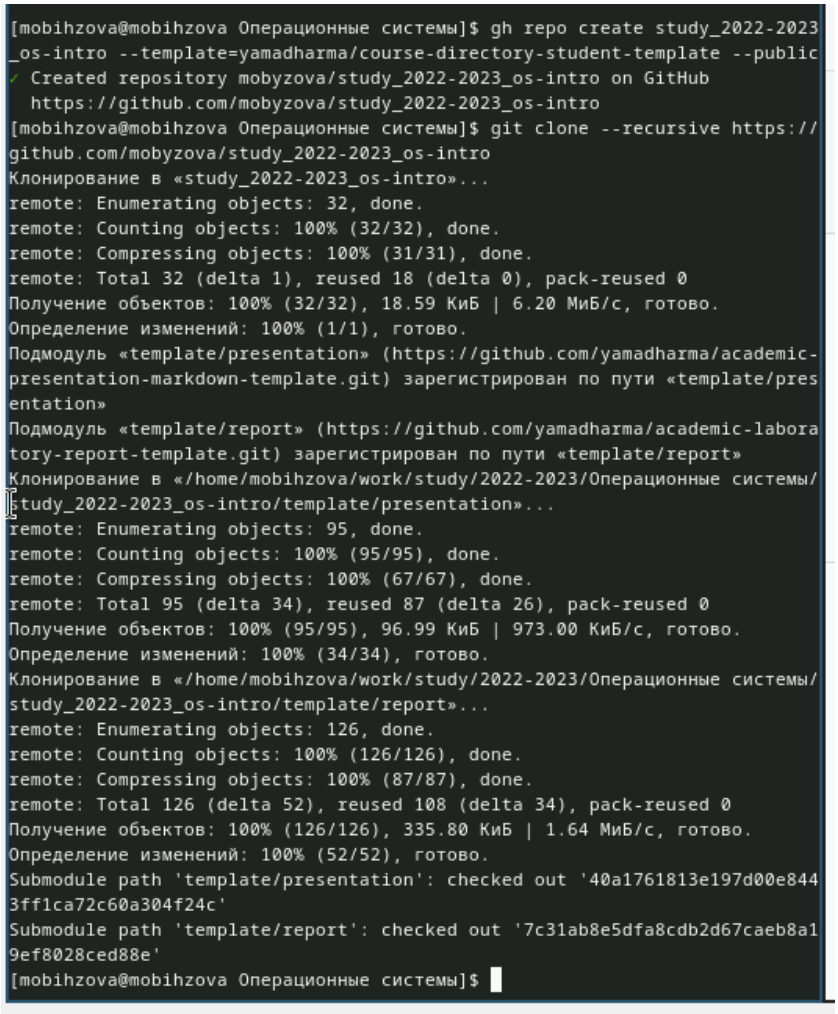

Рис. 3.21: Клонирование репозитория

3.10 Настройка каталога курса.

Перехожу в каталог курса (рис. 3.22).

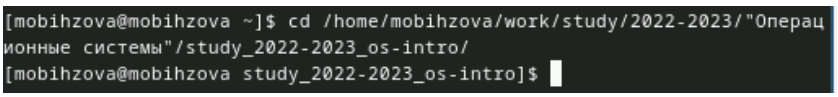

Рис. 3.22: Перемещение между директориями

Удаляю лишние файлы (рис. 3.23).

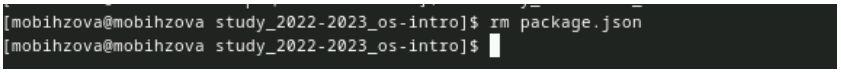

Рис. 3.23: Удаление лишних файлов

Создаю необходимые каталоги (рис. 3.24).

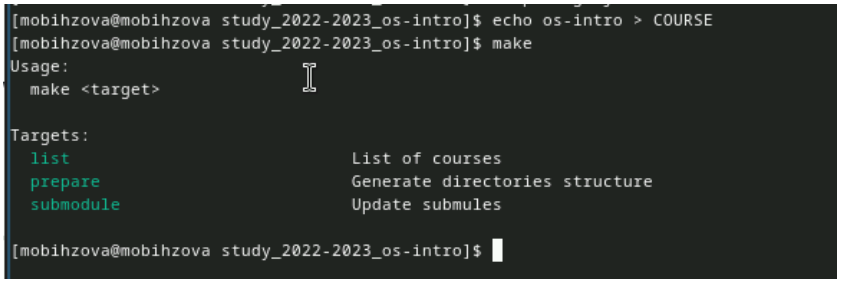

Рис. 3.24: Создание необходимых каталогов

Отправляю файлы на сервер (рис. 3.25).

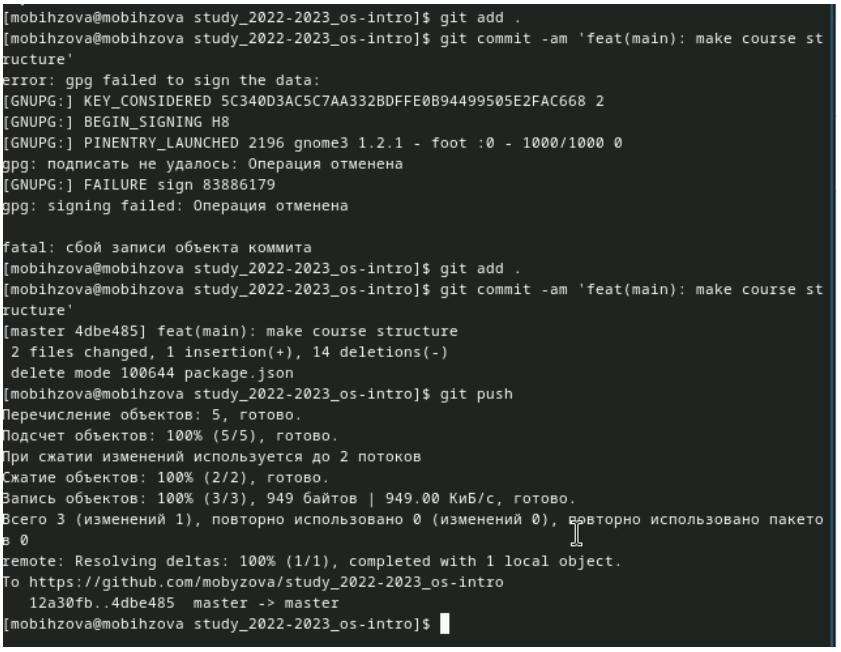

Рис. 3.25: Отправка файлов на сервер

4 Выводы

В ходе выполнения данной лабораторной работы я изучила идеологию и применение средств контроля версий, освоила умения по работе с git.

5 Ответы на контрольные вопросы.

1. Системы контроля версий (VCS) - программное обеспечение для облегчения работы с изменяющейся информацией. Они позволяют хранить несколько версий изменяющейся информации, одного и того же документа, может предоставить доступ к более ранним версиям документа. Используется для работы нескольких человек над проектом, позволяет посмотреть, кто и когда внес какое-либо изменение и т. д. VCS рриме-няются для: Хранения понлой истории изменений, сохранения причин всех изменений, поиска причин изменений и совершивших изменение, совместной работы над проектами.

2. Хранилище – репозиторий, хранилище версий, в нем хранятся все документы, включая историю их изменения и прочей служебной информацией. commit – отслеживание изменений, сохраняет разницу в изменениях.История – хранит все изменения в проекте и позволяет при необходимости вернуться/обратиться к нужным данным. Рабочая копия – копия проекта, основанная на версии из хранилища, чаще всего последней версии.

3. Централизованные VCS (например: CVS, TFS, AccuRev) – одно основное хранилище всего проекта. Каждый пользователь копирует себенеобходимые ему файлы из этого репозитория, изменяет, затем добавляет изменения обратно в хранилище. Децентрализованные VCS (например: Git, Bazaar) – у каждого пользователя свой вариант репозитория (возможно несколько вариантов), есть возможность добавлятьи забирать

изменения из любого репозитория. В отличие от классических, в распределенных (децентралиованных) системах контроля версий центральный репозиторий не является обязательным.

4. Сначала создается и подключается удаленный репозиторий, затем по мере изменения проекта эти изменения отправляются на сервер.

5. Участник проекта перед началом работы получает нужную ему версию проекта в хранилище, с помощью определенных команд, после внесения изменений пользователь размещает новую версию в хранилище. При этом предыдущие версии не удаляются. К ним можно вернуться в любой момент.

6. Хранение информации о всех изменениях в вашем коде, обеспечение удобства командной работы над кодом.

7. 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) отправка изменений конкретной ветки в центральный репозиторий: git push origin имя_ветки слияние ветки с текущим 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. git push -all отправляем из локального репозитория все сохраненные изменения в центральный репозиторий, предварительно создав локальный репозиторий и сделав предварительную конфигурацию.

9. Ветвление - один из параллельных участков в одном хранилище, исходящих из одной версии, обычно есть главная ветка. Между ветками, т. е. их концами возможно их слияние. Используются для разработки новых функций.

10. Во время работы над проектом могут создаваться файлы, которые не следуют добавлять в репозиторий. Например, временные файлы. Можно прописать шаблоны игнорируемых при добавлении в репозиторий типов файлов в файл .gitignore с помощью сервисов.

Список литературы

1. Лабораторная работа №2 [Электронный ресурс] URL: https://esystem.rudn.ru/mod/page/vi