Лабораторная работа №2

Первоначальная настройка git

Юсупова Ксения Равилевна

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Изучить идеологию и применение средств контроля версий, и освоить умения по работе с git.

2 Выполнение лабораторной работы

2.1 Установка программного обеспечения

Установка git(рис. 2.1).

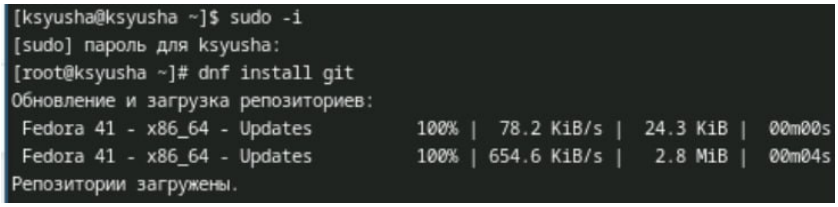

Рис. 2.1: Установили git

Установка gh(рис. 2.2).

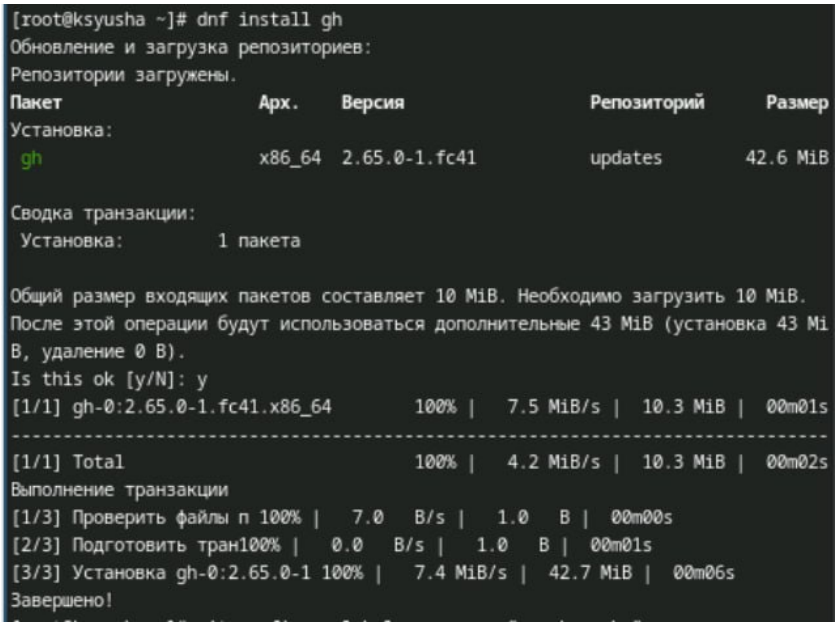

Рис. 2.2: Установили gh

2.2 Базовая настройка git

Зададим имя и email владельца репозитория(рис. 2.3).

```
[root@ksyusha ~]# git config --global user.name "yaneksyusha"
[root@ksyusha ~]# git config --global user.email "1132247531@pfur.ru"
```

Рис. 2.3: Задали имя и email владельца репозитория

Настроим utf-8 в выводе сообщений git и зададим имя начальной ветки (будем называть её master)(рис. 2.4).

```
[root@ksyusha ~]# git config --global core.quotepath false
[root@ksyusha ~]# git config --global init.defaultBranch master
```

Рис. 2.4: Настроили utf-8 в выводе сообщений git и задали имя начальной ветки

Настроим параметр autocrlf и параметр safecrlf(рис. 2.5).

```
[root@ksyusha ~]# git config --global core.autocrlf input
[root@ksyusha ~]# git config --global core.safecrlf warn
```

Рис. 2.5: параметр autocrlf и параметр safecrlf

2.3 Создание ключей ssh

Создаём ключи ssh по алгоритму rsa с ключём размером 4096 бит и по алгоритму ed25519(рис. 2.6).

Рис. 2.6: Создали ключи ssh

2.4 Создание ключей pgp

Генерируем ключ pgp с учетом необхомых опций(рис. 2.7).

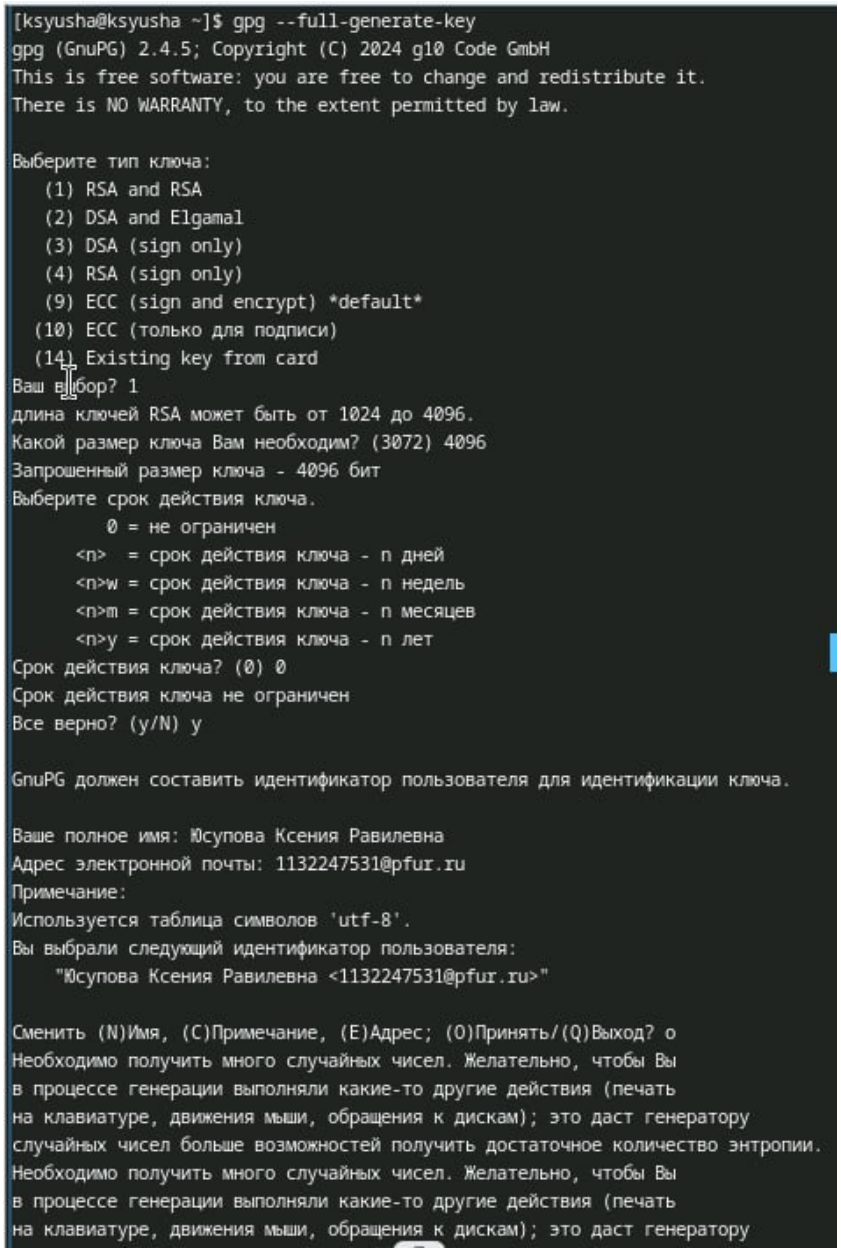

Рис. 2.7: Сгенерировали ключ pgp

2.5 Настройка github

Создаём учётную запись и заполнили основные данные на https://github.com.(рис. 2.8).

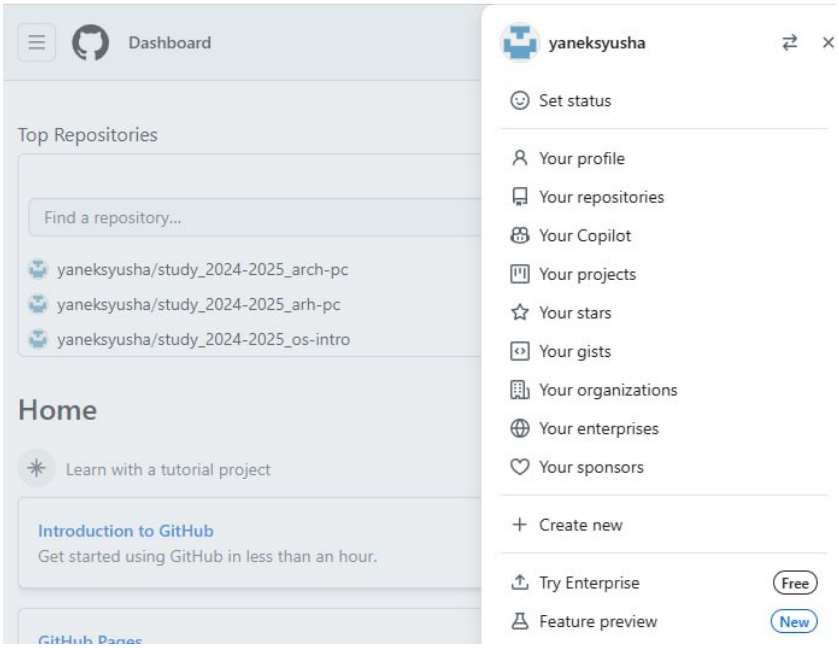

Рис. 2.8: Создали учетную запись

2.6 Добавление PGP ключа в GitHub

Выводим список ключей и копируем отпечаток приватного ключа(рис. 2.9).

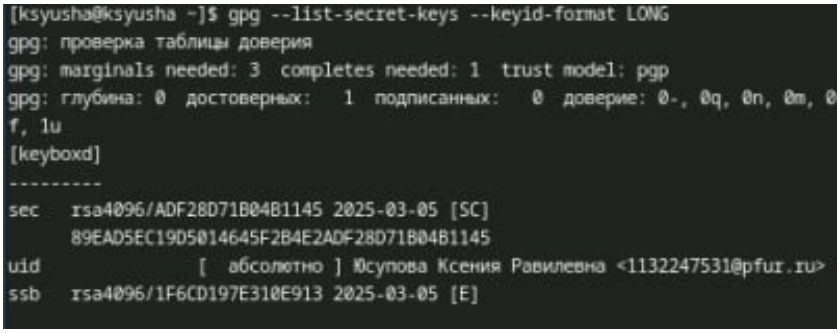

Рис. 2.9: Вывели список ключей и скопировали отпечаток приватного ключа

Скопировали наш сгенерированный PGP ключ в буфер обмена. Перешли в настройки GitHub, нажали на кнопку New GPG key и вставили полученный ключ в поле ввода.(рис. 2.10).

Рис. 2.10: Скопировали наш сгенерированный PGP ключ в буфер обмена и далее создали его на GitHub

2.7 Настройка автоматических подписей коммитов git

Используя введёный email, указали Git применять его при подписи коммитов(рис. 2.11).

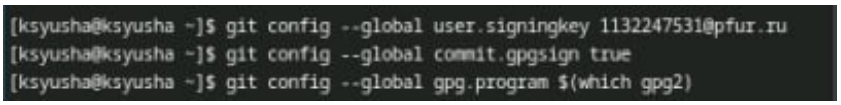

Рис. 2.11: Используя email, указали Git применять его при подписи коммитов

2.8 Настройка gh

Для начала авторизируемся(рис. 2.12).

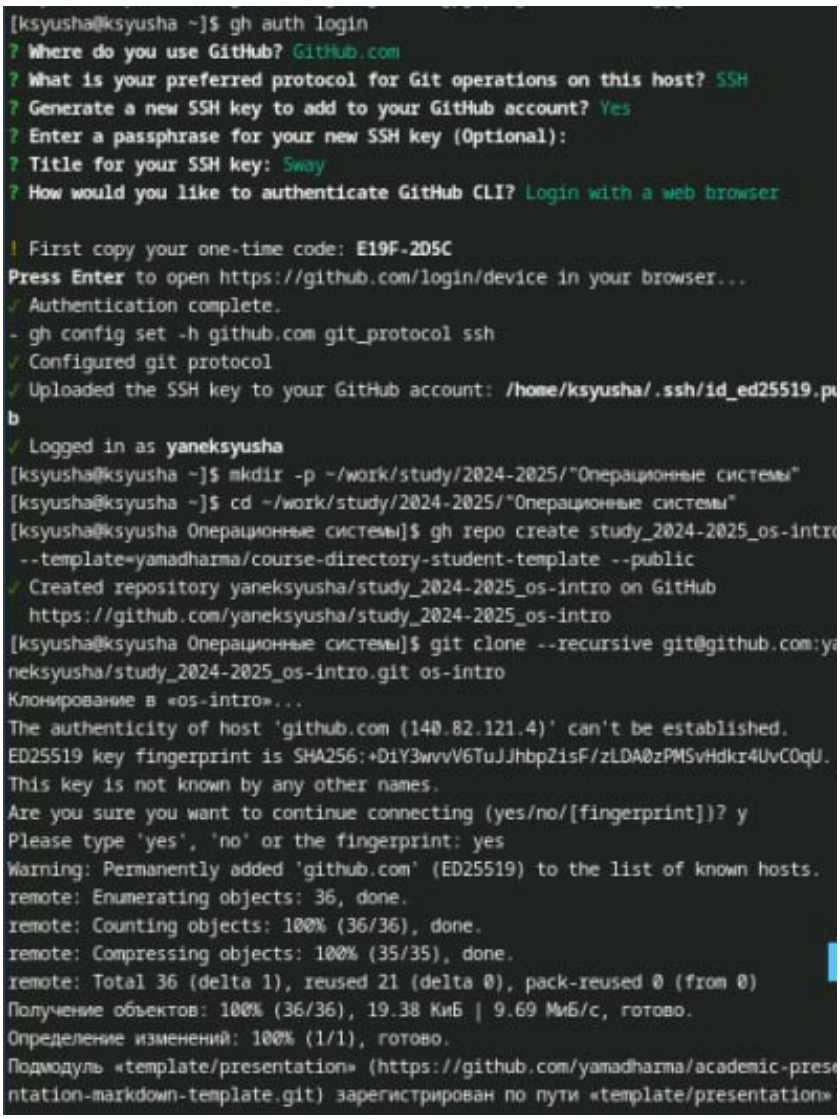

Рис. 2.12: авторизовались

2.9 Создание репозитория курса на основе шаблона

Создаём репозиторий курса на основе шаблона для 2024–2025 учебного года и предмета «Операционные системы» (рис. 2.13).

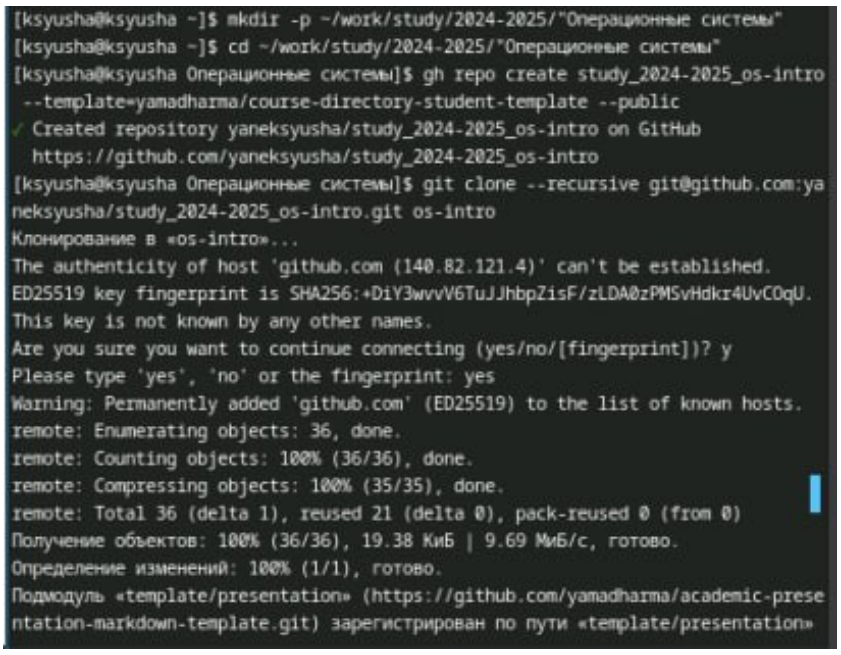

Рис. 2.13: Создаём необходимый репозиторий

2.10 Настройка каталога курса

Переходим в каталог курса (рис.2.14).

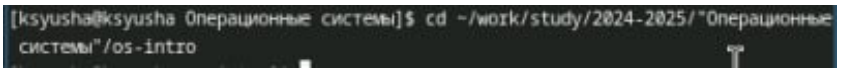

Рис. 2.14: Переходим в каталог курса

Удаляем лишние файлы, создаём необходимые каталоги и отправляем файлы на сервер(рис. 2.15).

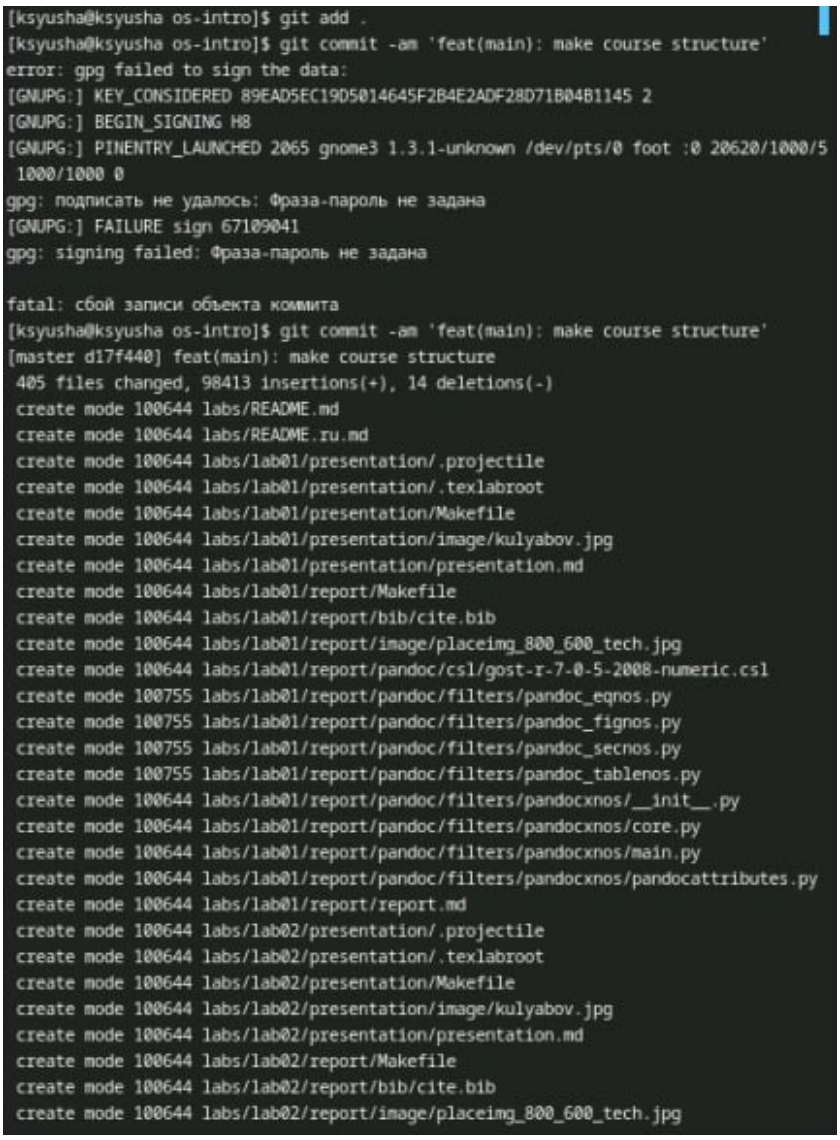

Рис. 2.15: Удаляем лишние файлы, создаём необходимые каталоги и отправляем файлы на сервер

Отправляем файлы на сервер(рис. 2.16).

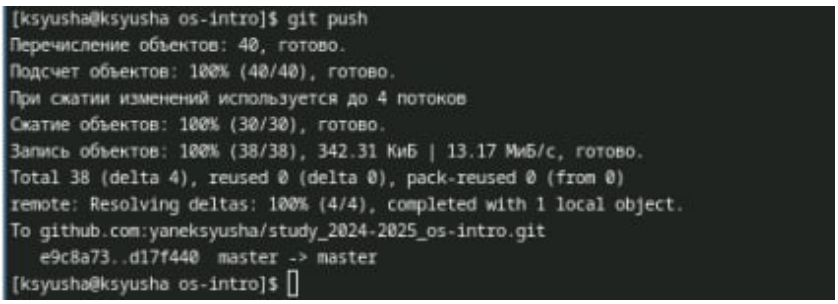

Рис. 2.16: Отправили файлы на сервер

3 Выводы

В ходе лабораторной работы мы изучили идеологию и применение средств контроля версий, а также освоили умения по работе с git.

4 Ответы на контрольные вопросы.

1. Системы контроля версий (VCS)- это инструменты, которые позволяют отслеживать изменения в файлах и координировать работу нескольких людей над этим проектом. Они предназначаются для отслеживания изменений в коде и файлах, удобно для использования в совместной работе над проектами для нескольких разработчиков.

2. Хранилище - это репозиторий, в котором хранятся все версии проекта и его файлы Commit - это операция, которая сохраняет текущее состояние фвйлов в хранилища с комментарием, описывающим эти изменения. История- это последовательность всех коммитов репозитория. Рабочая копия- копия, сделанная из версий репозтория, с которй непосредственно работает сам разработчик.

3. Централизованные VCS представляют собой все изменения, которые хранятся на одном центральном сервере, с ним работают все разработчики. В пример можно привести CVS.

Децентрализованные VCS представляют собой системы, в которых используется множество репозиториев одного проекта у каждого из разработчиков. Пример: Git, Mercurial.

4. Порядок работы с хранилищем при единоличной работе: Создание репозитория, разработка проекта и при внесении изменений в файлы отправляются на сервер

5. Порядок работы с общим хранилищем VCS.

Клонирование репозитория,создание новой ветки, внесение изменений: Изменение файлов в рабочей копии. Добавление и коммит изменений, синхронизация и отправка изменений.

6. Основные задачи Git.

Отслеживание изменений: Git отслеживает все изменения в файлах проекта.

Ведение истории: Git сохраняет историю всех коммитов, что позволяет видеть, кто и когда вносил изменения.

правление ветвями и слияниями: Git позволяет легко создавать ветки для новых функций и сливать их обратно в основную ветку после завершения работы.

Поддержка совместной работы: Git упрощает работу нескольких разработчиков над одним проектом, позволяя им синхронизировать свои изменения.

7. Команды git.

git init: Инициализация нового репозитория.
git clone : Клонирование удалённого репозитория.
git add : Добавление изменённых файлов в индекс.
git commit -m “message”: Фиксация изменений с сообщением.
git status: Проверка статуса рабочей копии.
git log: Просмотр истории коммитов.
git branch: Управление ветками.
git checkout : Переключение между ветками.
git merge : Слияние указанной ветки с текущей.
git pull: Получение обновлений из удалённого репозитория.
git push origin : Отправка изменений в удалённый репозиторий.

8. Примеры работы с локальным и удалённым репозиториями.

Локальный репозиторий: Инициализация нового репозитория git init

Создание файла и добавление текста echo “Hello World” > hello.txt

Добавление файла в индекс git add hello.txt

Фиксация изменений git commit -m “Initial commit”

Удалённый репозиторий: Клонирование удалённого репозитория git clone https://github.com/user/repo.git

Переход в директорию клонированного репозитория cd repo

Создание нового файла echo “New feature” » feature.txt

Добавление нового файла в индекс git add feature.txt

Фиксация изменений git commit -m “Add new feature”

Отправка изменений в удалённый репозиторий git push origin main

9. Ветви (branches).

Ветви в Git позволяют разработчикам работать над новыми функциями или исправлениями ошибок, не затрагивая основную (обычно это ветка main или master). Это позволяет изолировать изменения до тех пор, пока они не будут готовы к интеграции.

10. Игнорирование файлов при commit.

Файл .gitignore используется для указания файлов и директорий, которые не должны отслеживаться системой контроля версий. Это полезно для исключения временных файлов, конфиденциальной информации или других данных, которые не должны попадать в репозиторий.