Отчёт по лабораторной работе № 2

ОТЧЁТ

Саенко Ангелина Андреевна

Содержание

Список иллюстраций

Список таблиц

Цель работы

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

Задание

Создать базовую конфигурацию для работы с git.

Создать ключ SSH.

Создать ключ PGP.

Настроить подписи git.

Зарегистрироваться на GitHub.

Создать локальный каталог для выполнения заданий по предмету.

Выполнение лабораторной работы

Для начала установим git. В моём случае он уже установлен (рис. [-@fig:001]).

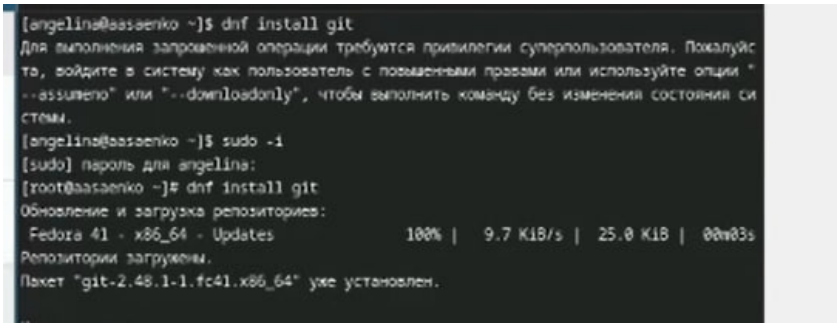

Рис. 1: Установка git.

Теперь установим gh.

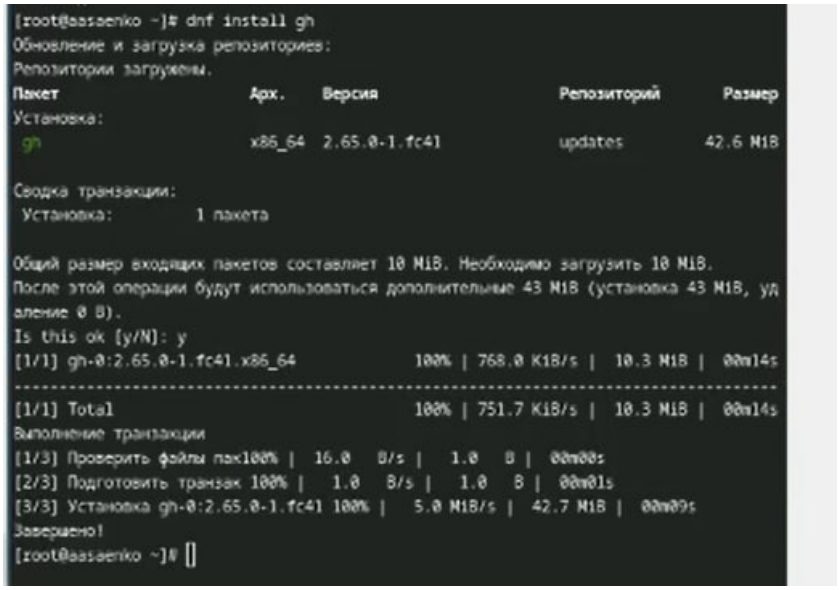

Рис. 2: Установка gh.

Далее, зададим имя для владельца репозитория. В данном случае это моё имя.

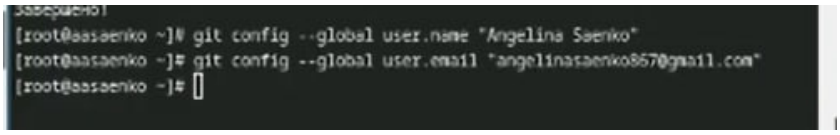

Рис. 3: Указание имени

Теперь зададим почту. Я задала почту, на которую у меня зарегистрирован аккаунт на GitHub.

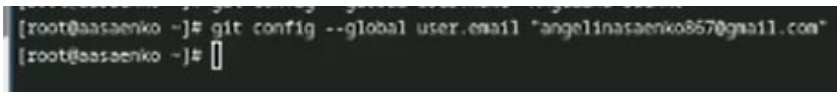

Рис. 4: Указание почты.

Настроим кодировку utf8 в выводе сообщений git.

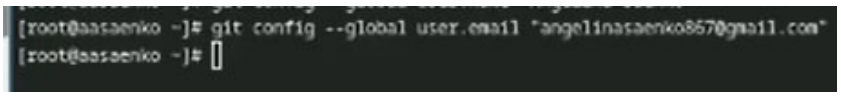

Рис. 5: Настройка кодировки utf8.

Зададим имя начальной ветки, настроим параметры autocrlf и safecrlf

Рис. 6: Настройка git

Создадим ключ RSA размером 4096 бит.

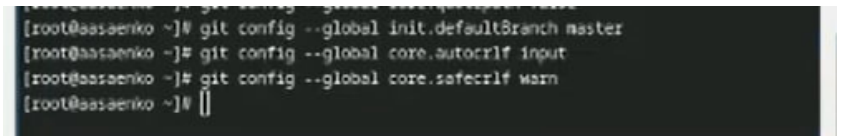

Рис. 7: Создание ключа RSA

Теперь создадим ключ по алгоритму ed25519.

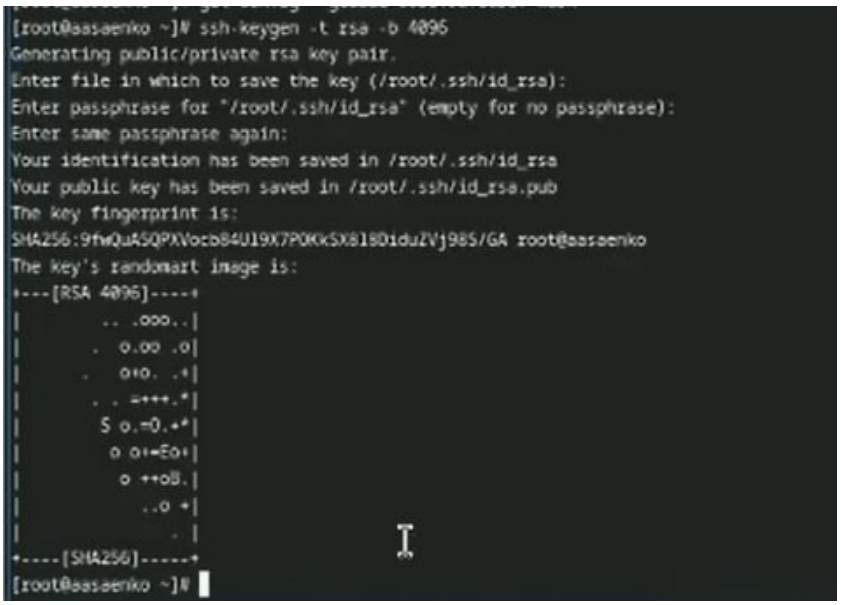

Рис. 8: Создание ключа ed25529.

Теперь создадим ключ gpg. Выбираем из предложенных вариантов первый тип (RSA and RSA), размер ключа задаём 4096 бит и делаем срок действия ключа неограниченным.

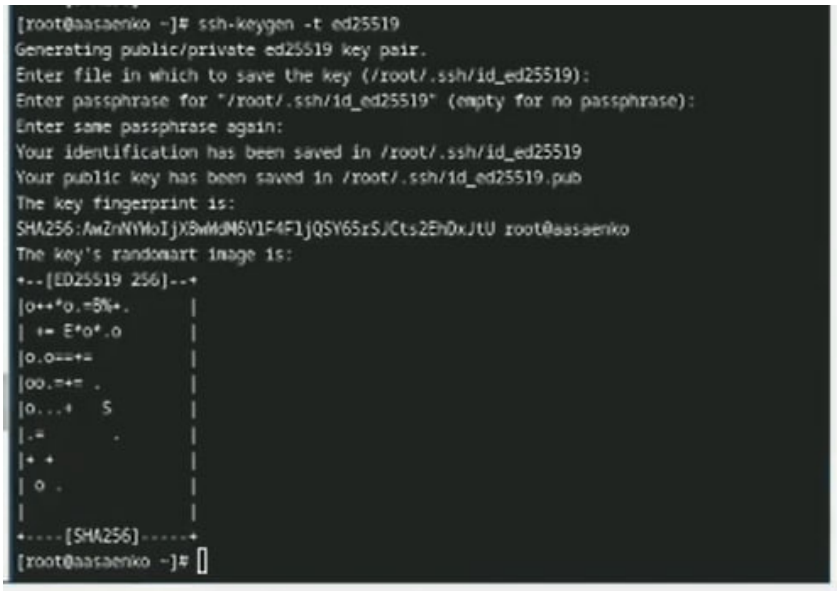

Рис. 9: Создание pgp ключа (1)

После нас попросят ввести свои данные. Мы вводим имя и адрес электронной почты. После этого соглашаемся с генерацией ключа

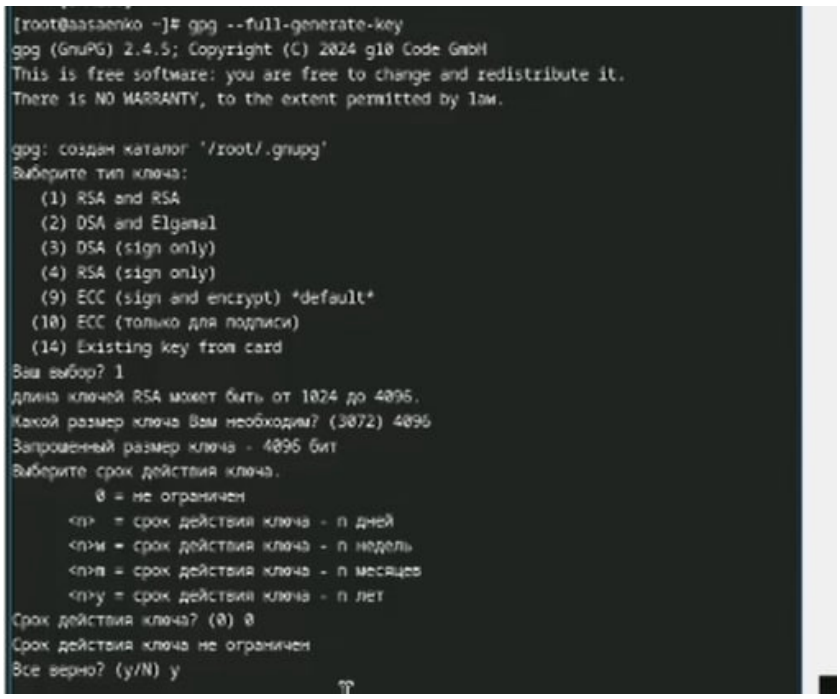

Рис. 10: Создание pgp ключа (2)

Далее, выводим список pgp ключей .

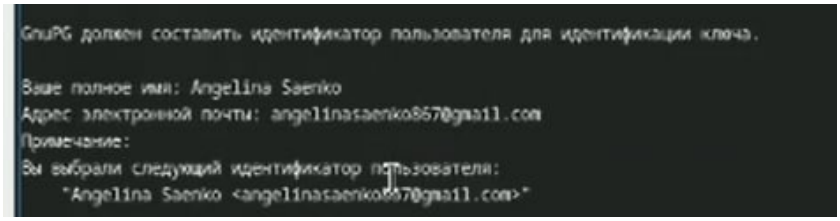

Рис. 11: Список pgp ключей.

Копируем наш ключ в буфер обмена

Рис. 12: Копирование ключа.

Вставляем этот ключ на гитхаб, и задаём ему имя. Я выбрала имя Sway.

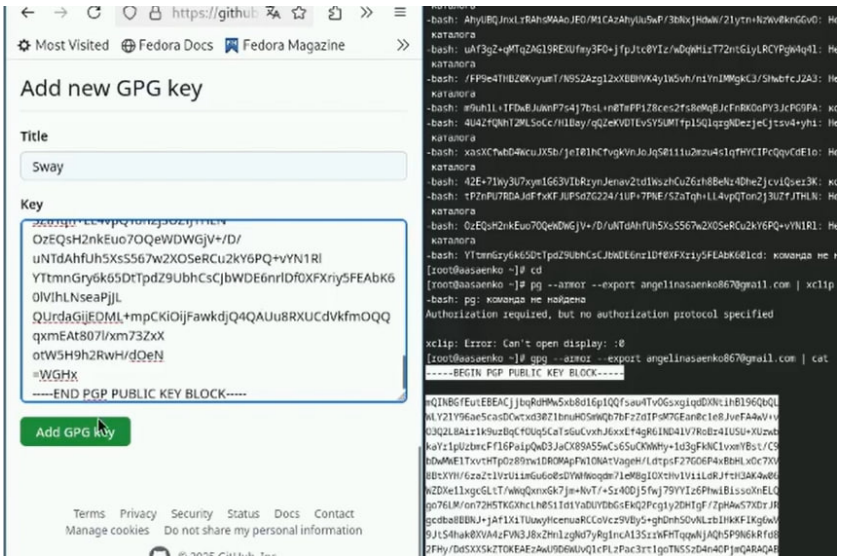

Рис. 13: Вставка ключа в GitHub

Теперь производим настройку автоматических подписей

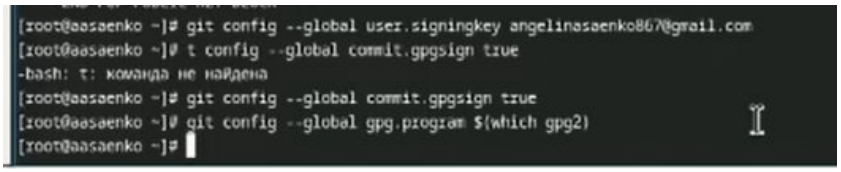

Рис. 14: Настройка автоматических подписей коммитов git.

После, нам нужно авторизоваться в github с помощью gh. Мы выбираем сайт для авторизации (GitHub.com) , после выбираем предпочитаемый протокол (SSH), публичный SSH ключ (id_rsa.pub) , и имя для ключа (Sway) . В качестве способа авторизации выбираем авторизацию через браузер .

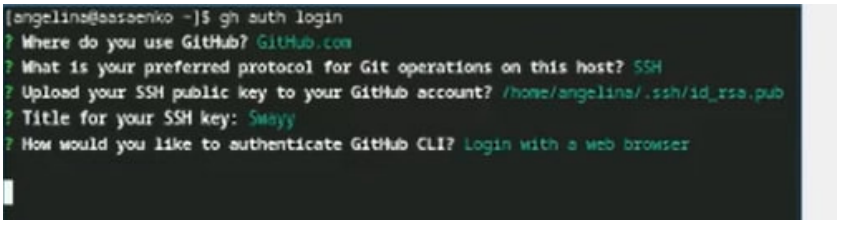

Рис. 15: Авторизация в gh.

Теперь создаём рабочую директорию курса и переходим в неё

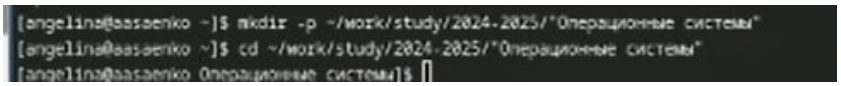

Рис. 16: Создание рабочей директории и переход в неё.

Далее, создаём репозиторий для лабораторных работ из шаблона.

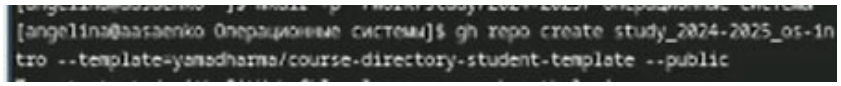

Рис. 17: Создание репозитория курса.

И клонируем его к себе на компьютер

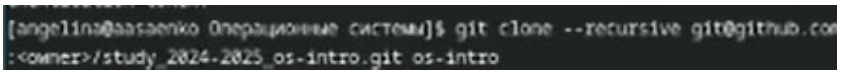

Рис. 18: Клонирование репозитория .

Переходим в него с помощью cd и удаляем ненужные файлы (package.json) и создаём необходимые каталоги , записав в файл COURSE строку os-intro (это нам текущий курс) и прописываем make prepare для того , чтобы нужные нам каталоги создались

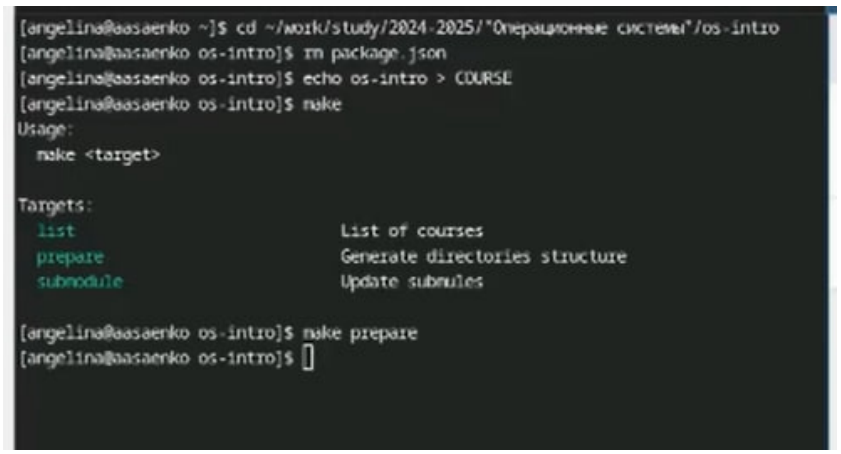

Рис. 19: Удаление ненужных файлов и использование make.

Теперь добавляем нашу папку для отправки .

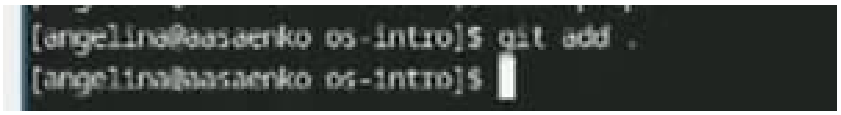

Рис. 20: Использование git add.

Делаем коммит ,в котором указываем , что мы сделали структуру курса.

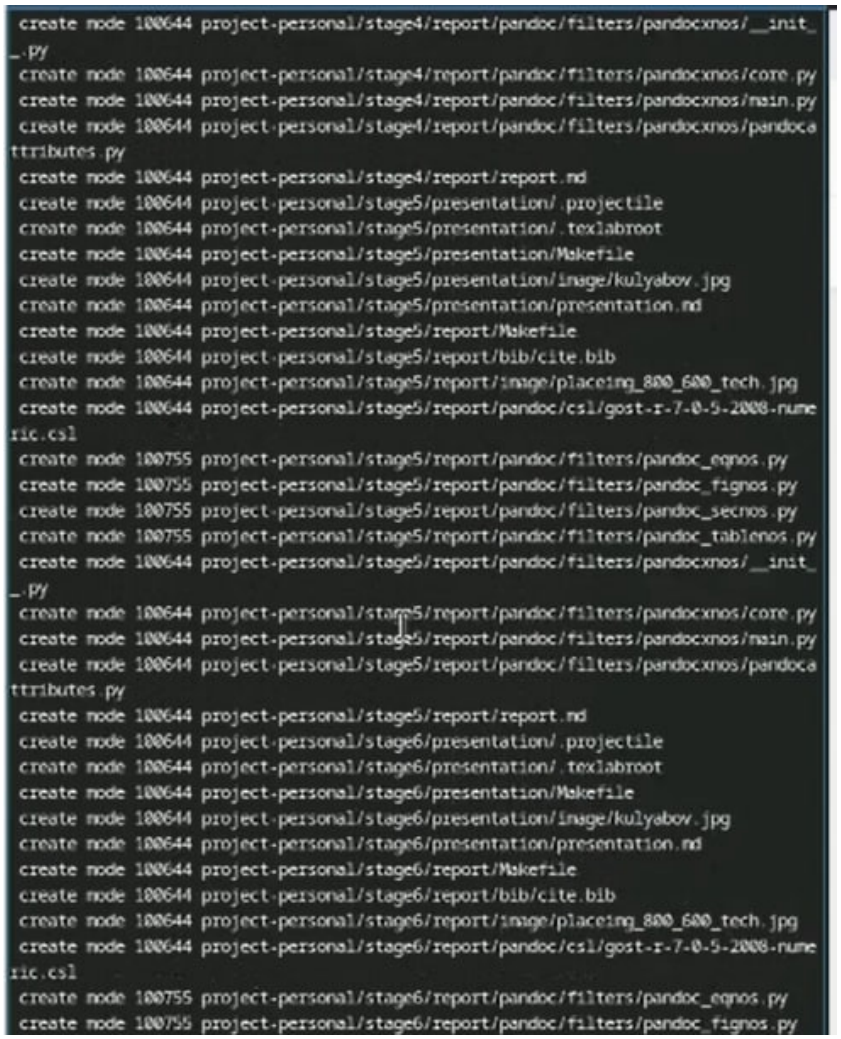

Рис. 21: Использование git commit

И отправляем файлы на сервер GitHub с помощью команды push

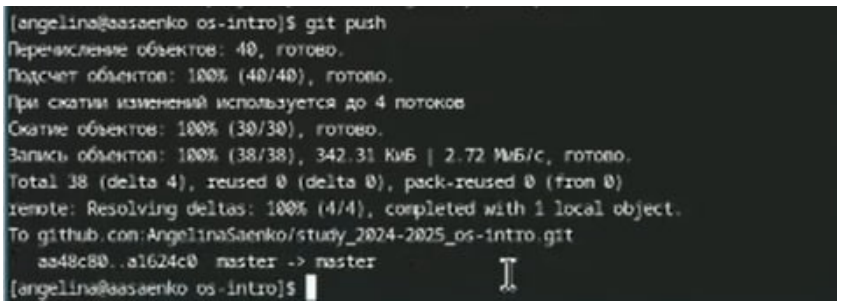

Рис. 22: Использование git push

Выводы

Была произведена установка git , проведена его первоначальная настройка, были созданы ключи для авторизации и подписи ,а также создан репозиторий курса из предложенного шаблона .

Ответы на контрольные вопросы

1. Системы контроля версий - это системы, в которых мы можем хранить свои проекты и выкладывать их обновления, контролируя релизы и каждые внесённые изменения. Эти системы нужны для работы над проектами, чтобы иметь возможность контролировать версии проектов и в случае командной работы контролировать изменения, внесённые всеми участниками. Также, VCS позволяет откатываться на более ранние версии.
2. Хранилище - репозиторий, в нём хранятся все файлы проекта и все его версии commit - внесённые изменения в репозитории история - это история изменений файлов проекта рабочая копия - копия, сделанная из версии репозитория, с которой непосредственно работает сам разработчик
3. Централизованные системы контроля версий имеют один центральный репозиторий, с которым работают все разработчики. Примером является CVS , который является уже устаревшей системой. В децентрализованных системах же используется множество репозиториев одного проекта у каждого из разработчиков, при этом репозитории можно объединять брать из каждого только то, что нужно. Примером является знакомый нам Git.
4. Создаётся репозиторий, и разрабатывается проект. При внесении изменений файлы отправляются на сервер.
5. Разработчик клонирует репозиторий к себе на компьютер, и после внесения изменений выгружает их на сервер в качестве отдельной версии. После этого разработчики с более высокими правами могут, например , объединить его версию с текущей

6. . Хранение файлов проекта, а также обеспечение командной работы, и контроль за версиями проекта
7. git clone - клонирует проект с сервера на компьютер git commit - добавляет папку для выгрузки на сервер git push - фиксирует изменения репозитория git pull - выгружает изменения на сервер git pull - получить изменения с сервера git rm - удалить файл git status - получить статус репозитория
8. С локальным: git commit -am “added files” - создаёт коммит С удаленным: git push - загрузить данные на удалённый сервер
9. Ветки - это несколько независимых копий проекта, в каждой из которых ведётся разработка какой-то конкретной функции, при этом ветки существуют параллельно. Они нужны, когда нужно параллельно вести разработку нескольких функций, а в конце их можно объединить в одну.
10. Игнорировать файлы можно, внося их в файл .gitignore. игнорировать файлы нужно, когда их не нужно добавлять в репозиторий. Например, это могут быть файлы виртуального окружения (venv).

Список литературы

[@tanenbaum_book_modern-os_ru][@robbins_book_bash_en][@zarrelli_book_mastering-bash_en] [@nam_ewhbook_learning-bash_en] ::: :::