Отчёт по лабораторной работе № 2

ОТЧЁТ

Саенко Ангелина Андреевна

Содержание

Список иллюстраций

Список таблиц

# Цель работы

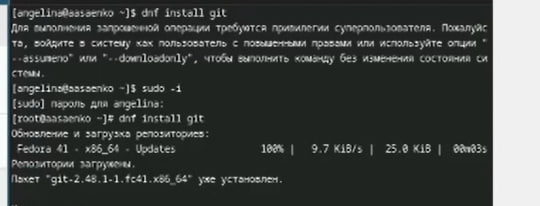
Изучить идеологию и применение средств контроля версий. Освоить умения по работе с git.

# 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.

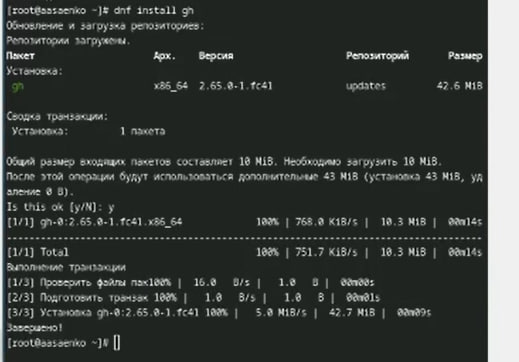
# Выполнение лабораторной работы

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



Установка git.

Теперь установим gh.



Установка gh.

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



Указание имени

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

Указание почты.

Указание почты.

Настроим кодировку utf8 в выводе сообщений git.

Настройка кодировки utf8.

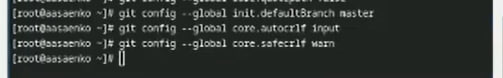
Настройка кодировки utf8.

Зададим имя начальной ветки, настроим параметры autocrlf и safecrlf

Настройка git

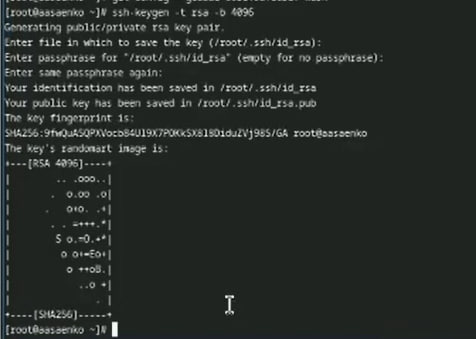
Настройка git

Создадим ключ RSA размером 4096 бит.



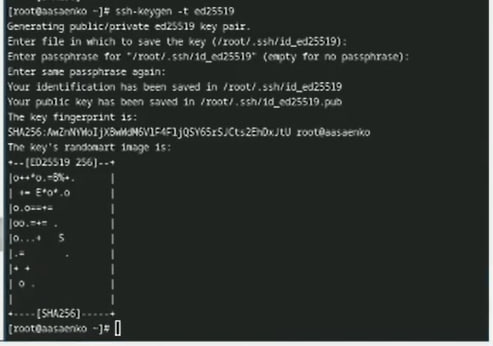
Создание ключа RSA

Теперь создадим ключ по алгоритму ed25519.



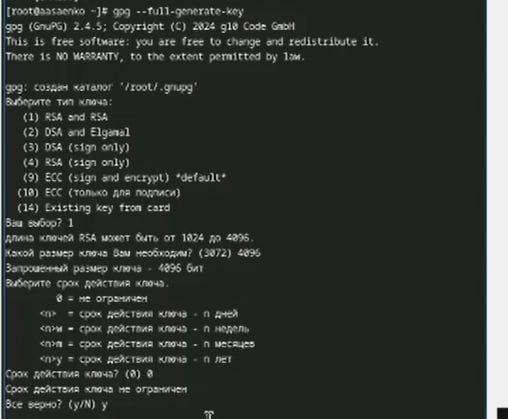
Создание ключа ed25529.

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



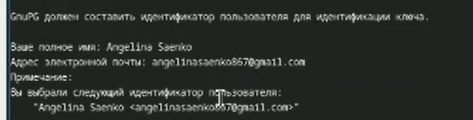
Создание pgp ключа (1)

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



Создание pgp ключа (2)

Далее, выводим список pgp ключей .



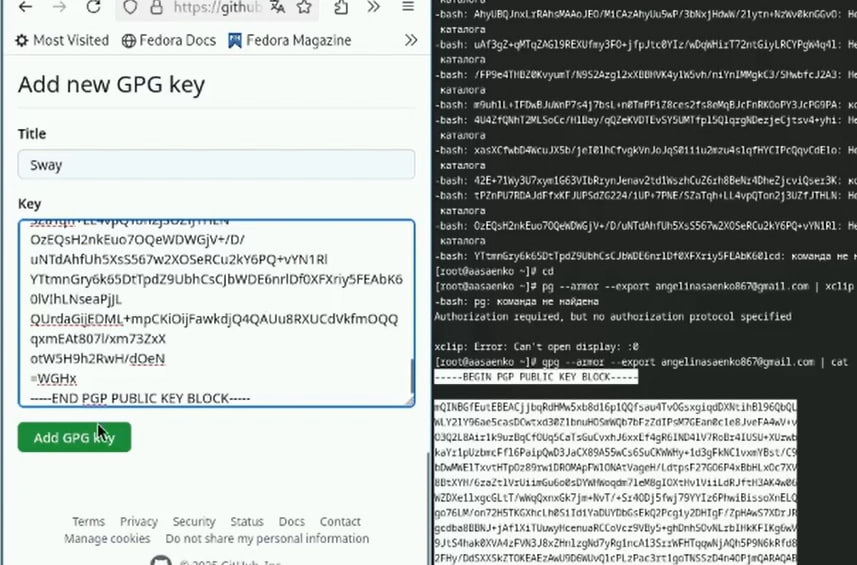
Список pgp ключей.

Копируем наш ключ в буфер обмена

Копирование ключа.

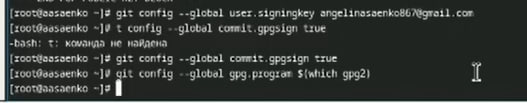
Копирование ключа.

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



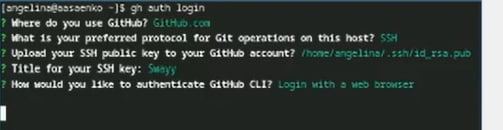
Вставка ключа в GitHub

Теперь производим настройку автоматических подписей



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 .



Авторизация в gh.

Теперь создаём рабочую директорию курса и переходим в неё

Создание рабочей директории и переход в неё.

Создание рабочей директории и переход в неё.

Далее, создаём репозиторий для лабораторных работ из шаблона.

Создание репозитория курса.

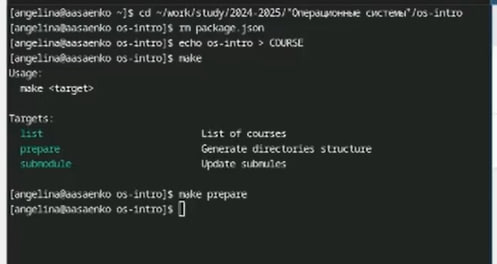
Создание репозитория курса.

И клонируем его к себе на компьютер

Клонирование репозитория .

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



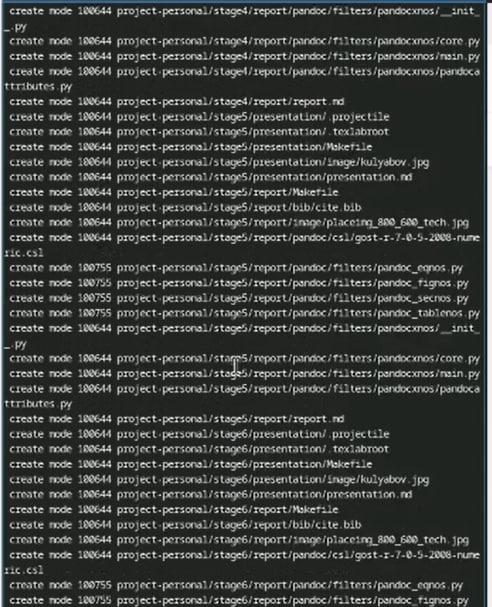
Удаление ненужных файлов и использование make.

Теперь добавляем нашу папку для отправки .

Использование git add.

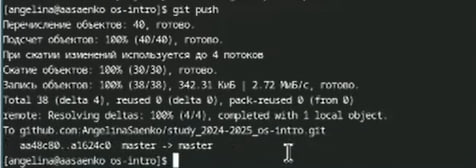
Использование git add.

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



Использование git commit

И отправляем файлы на сервер GitHub с помощью команды push



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 СVS , 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] ::: :::