Отчёта по лабораторной работе

Лабораторная работа №2

Диана Садова Алексеевна

Содержание

# 1 Цель работы

Изучить идеологию и применение средств контроля версий.

Освоить 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 Теоретическое введение

Системы контроля версий (Version Control System, VCS) применяются при работе нескольких человек над одним проектом.Обычно основное дерево проекта хранится в локальном или удалённом репозитории, к которому настроен доступ для участников проекта. При внесении изменений в содержание проекта система контроля версий позволяет их фиксировать, совмещать изменения, произведённые разными участниками проекта, производить откат к любой более ранней версии проекта, если это требуется.

В классических системах контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов. Выполнение большинства функций по управлению версиями осуществляется специальным сервером. Участник проекта (пользователь) перед началом работы посредством определённых команд получает нужную ему версию файлов. После внесения изменений, пользователь размещает новую версию в хранилище. При этом предыдущие версии не удаляются из центрального хранилища и к ним можно вернуться в любой момент. Сервер может сохранять не полную версию изменённых файлов, а производить так называемую дельта-компрессию — сохранять только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных.

Системы контроля версий поддерживают возможность отслеживания и разрешения конфликтов, которые могут возникнуть при работе нескольких человек над одним файлом. Можно объединить (слить) изменения, сделанные разными участниками (автоматически или вручную), вручную выбрать нужную версию, отменить изменения вовсе или заблокировать файлы для изменения. В зависимости от настроек блокировка не позволяет другим пользователям получить рабочую копию или препятствует изменению рабочей копии файла средствами файловой системы ОС, обеспечивая таким образом, привилегированный доступ только одному пользователю, работающему с файлом.

Системы контроля версий также могут обеспечивать дополнительные, более гибкие функциональные возможности. Например, они могут поддерживать работу с несколькими версиями одного файла, сохраняя общую историю изменений до точки ветвления версий и собственные истории изменений каждой ветви. Кроме того, обычно доступна информация о том, кто из участников, когда и какие изменения вносил. Обычно такого рода информация хранится в журнале изменений, доступ к которому можно ограничить.

В отличие от классических, в распределённых системах контроля версий центральный репозиторий не является обязательным.

Среди классических VCS наиболее известны CVS, Subversion, а среди распределённых — Git, Bazaar, Mercurial. Принципы их работы схожи, отличаются они в основном синтаксисом используемых в работе команд.

# 4 Последовательность выполнения работы

## 4.1 Установка программного обеспечения

### 4.1.1 Установка git

Установим git:(рис. 1).

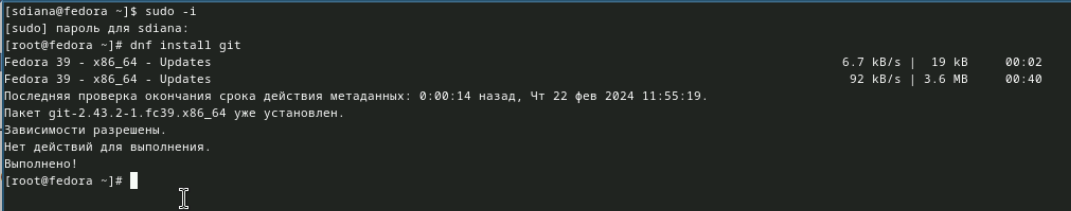


Рис. 1: Установка git

Установка завершилась, можем преступать к следующему пункту

### 4.1.2 Установка gh

Fedora: (рис. 2).

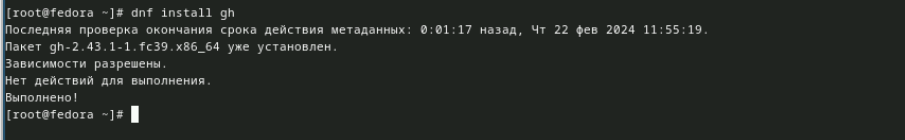


Рис. 2: Установка gh

После завершения этой установки мы можем начинать работу с git

## 4.2 Базовая настройка git

Зададим имя и email владельца репозитория: (рис. 3).

git config --global user.name "Name Surname"  
git config --global user.email "work@mail"

Настроим utf-8 в выводе сообщений git: (рис. 3).

git config --global core.quotepath false

Настройте верификацию и подписание коммитов git (см. Верификация коммитов git с помощью GPG).

Зададим имя начальной ветки (будем называть её master): (рис. 3).

git config --global init.defaultBranch master

Параметр autocrlf: (рис. 3).

git config --global core.autocrlf input

Параметр safecrlf: (рис. 3).

git config --global core.safecrlf warn

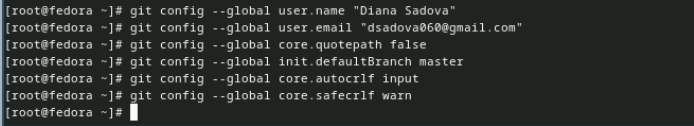


Рис. 3: Настраиваим git для дальнейшей работы

Заводим имя и email владельца, зададим имя начальной ветки, настраиваем параметры для дольнейшей работы

## 4.3 Создайте ключи ssh

по алгоритму rsa с ключём размером 4096 бит:(рис. 4).

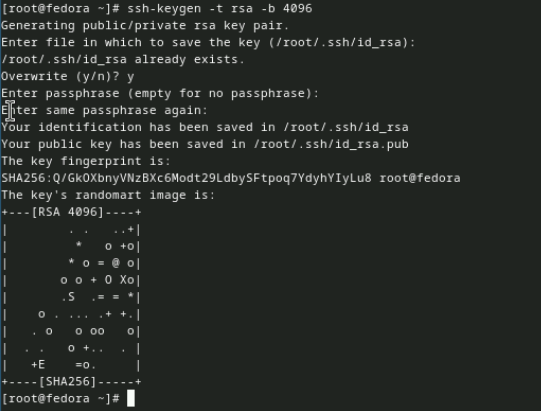


Рис. 4: Создаем ключ ssh с размером 4096 бит

по алгоритму ed25519:(рис. 5).

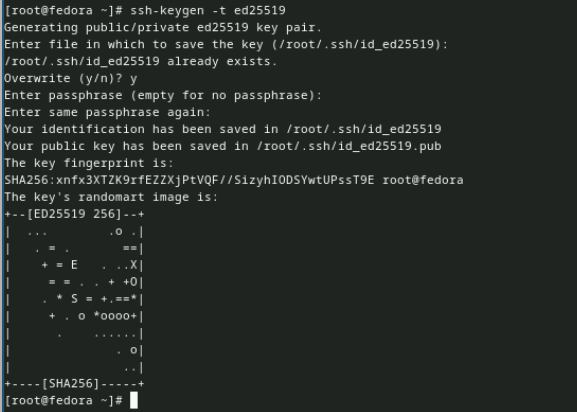


Рис. 5: Создаем ключ ssh c ed25519

Эти ключи нам понадобятся при дальнейшим выполнение лабороторной работы

## 4.4 Создайте ключи pgp

Генерируем ключ (рис. 6).

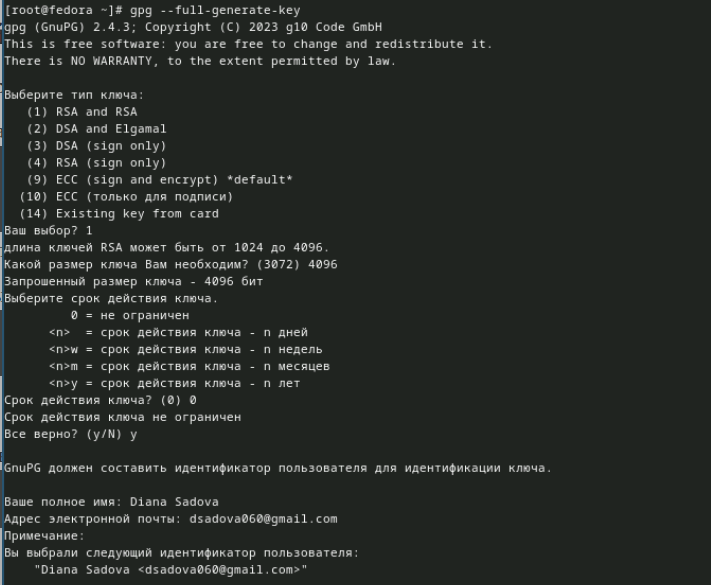


Рис. 6: Создаем ключи pgp

Из предложенных опций выбираем:

тип RSA and RSA;   
  
размер 4096;  
  
 выберите срок действия; значение по умолчанию — 0 (срок действия не истекает никогда).

GPG запросит личную информацию, которая сохранится в ключе:

Имя (не менее 5 символов).  
 Адрес электронной почты.  
  
 При вводе email убедитесь, что он соответствует адресу, используемому на GitHub.  
   
 Комментарий. Можно ввести что угодно или нажать клавишу ввода, чтобы оставить это поле пустым.   
 (рис. [-@fig:007]).

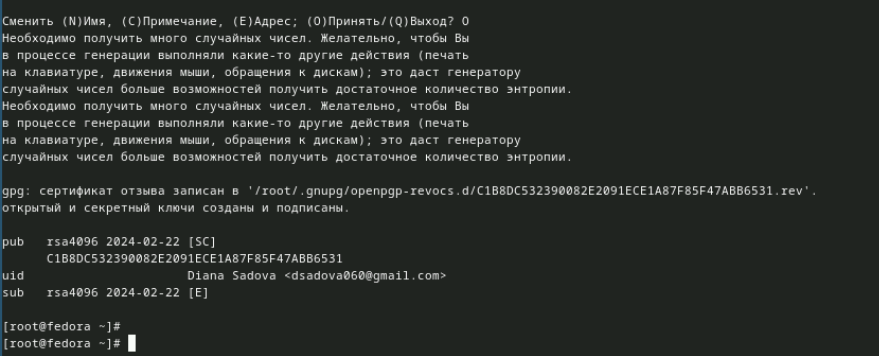


Рис. 7: Создаем ключи pgp

Задаем параметры для ключа и зашершаем процесс его создания

## 4.5 Настройка github

Создайте учётную запись на https://github.com.  
Заполните основные данные на https://github.com.  
(рис. [-@fig:008]).

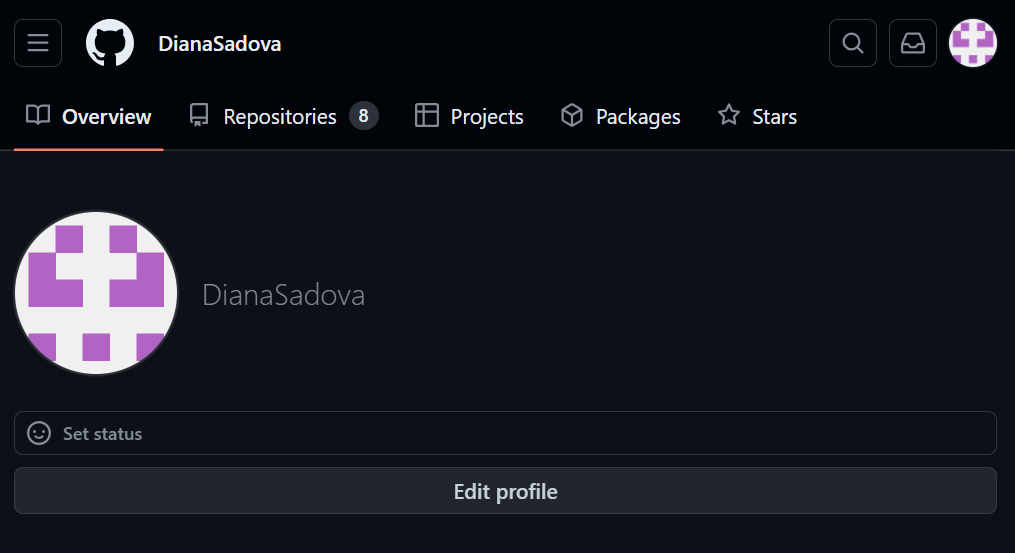


Рис. 8: Учетная запись

Моя учетная запись, которую я создала в прошлом семестре

## 4.6 Добавление PGP ключа в GitHub

Выводим список ключей и копируем отпечаток приватного ключа:(рис. 9).

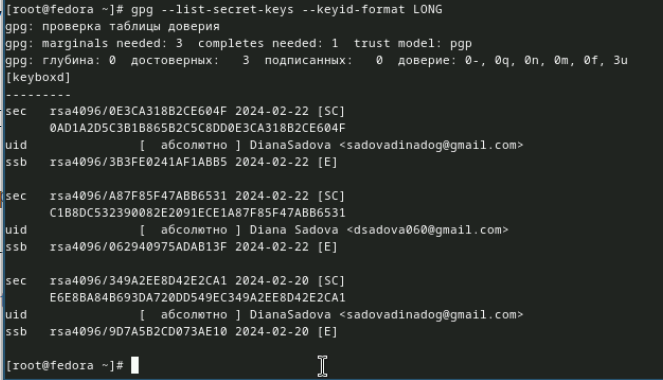


Рис. 9: Список ключей. Скопировали отпечаток приватного ключа

Отпечаток ключа — это последовательность байтов, используемая для идентификации более длинного, по сравнению с самим отпечатком ключа.

Формат строки:

sec Алгоритм/Отпечаток\_ключа Дата\_создания [Флаги] [Годен\_до]  
 ID\_ключа

Cкопируйте ваш сгенерированный PGP ключ в буфер обмена:

gpg --armor --export <PGP Fingerprint> | xclip -sel clip

Перейдите в настройки GitHub (https://github.com/settings/keys), нажмите на кнопку New GPG key и вставьте полученный ключ в поле ввода. (рис. 10),(рис. 11).



Рис. 10: Копируем ключ из терминала

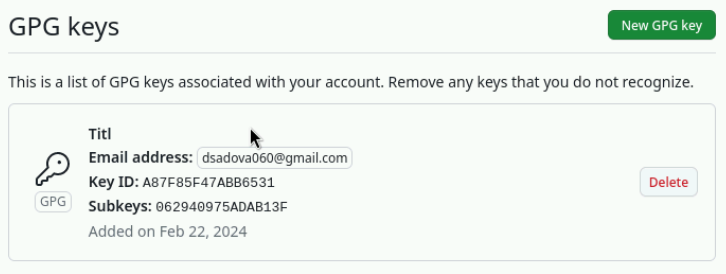


Рис. 11: Ключ на сайте Github

После того как мы убидились, что ключ на сайте Github, мы можем продолжить работу

## 4.7 Настройка автоматических подписей коммитов git

Используя введёный email, укажите Git применять его при подписи коммитов:(рис. 12),(рис. 13),(рис. 14).

Настройка автоматических подписей коммитов git

Рис. 12: Настройка автоматических подписей коммитов git

Настройка автоматических подписей коммитов git

Рис. 13: Настройка автоматических подписей коммитов git

Настройка автоматических подписей коммитов git

Рис. 14: Настройка автоматических подписей коммитов git

## 4.8 Настройка gh

Для начала необходимо авторизоваться(рис. 15).

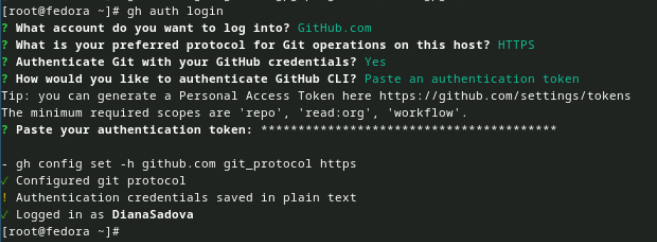


Рис. 15: Авторизация

Авторизуемся с помощью token

Утилита задаст несколько наводящих вопросов.

Авторизоваться можно через броузер.

## 4.9 Шаблон для рабочего пространства

Рабочее пространство для лабораторной работы

Репозиторий: https://github.com/yamadharma/course-directory-student-template. (рис. 16),(рис. 17).

Скачиваем репозиторий

Рис. 16: Скачиваем репозиторий

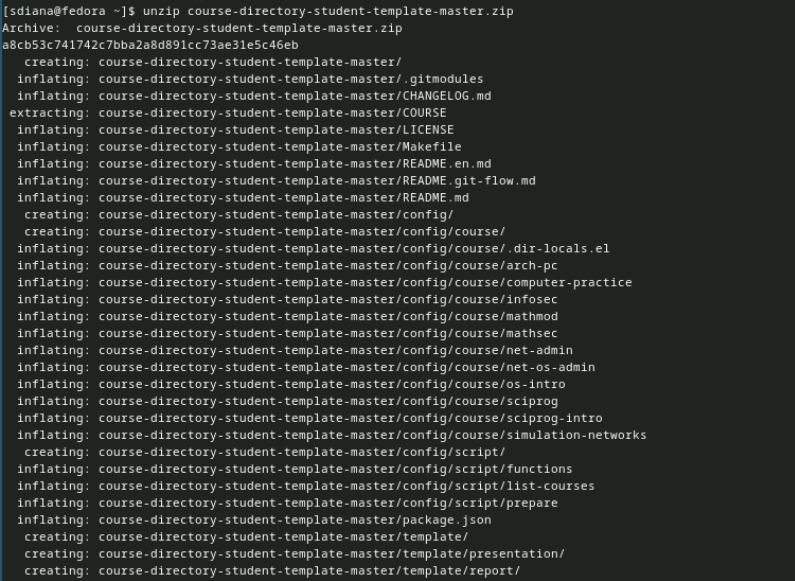


Рис. 17: Открываем zip фаил

Мы создали рабочее пространство, что поможет в дальнейшей деятельности

### 4.9.1 Сознание репозитория курса на основе шаблона

Необходимо создать шаблон рабочего пространства (см. Рабочее пространство для лабораторной работы).

Например, для 2022–2023 учебного года и предмета «Операционные системы» (код предмета os-intro) создание репозитория примет следующий вид:(рис. 18),(рис. 19),(рис. 20).

Создаем и открываем нужную нам папку

Рис. 18: Создаем и открываем нужную нам папку

Создаем новый репридиторий

Рис. 19: Создаем новый репридиторий



Рис. 20: Клонируем репридиторий

На последним шаге мы кланировали недавно созданный репридиторий “…so-intro”

### 4.9.2 Настройка каталога курса

Перейдите в каталог курса: (рис. 21).

cd ~/work/study/2022-2023/"Операционные системы"/os-intro

Удалите лишние файлы: (рис. 21).

rm package.json

Создайте необходимые каталоги: (рис. 21).

echo os-intro > COURSE

Следуем командам описанным выше

Рис. 21: Следуем командам описанным выше

make

(рис. 22),(рис. 23),(рис. 24).

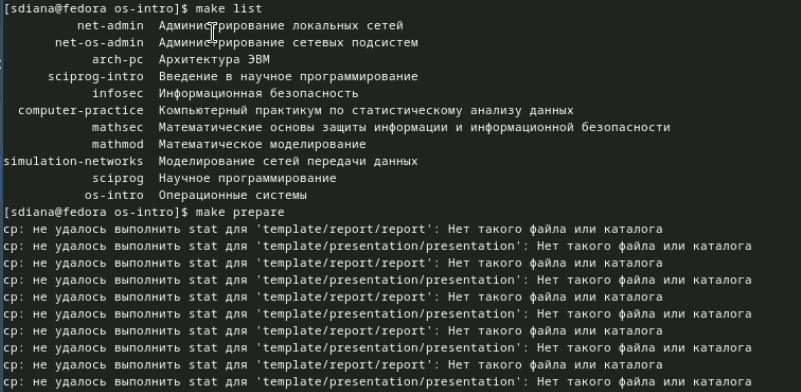


Рис. 22: Установка make list



Рис. 23: Установка make prepare

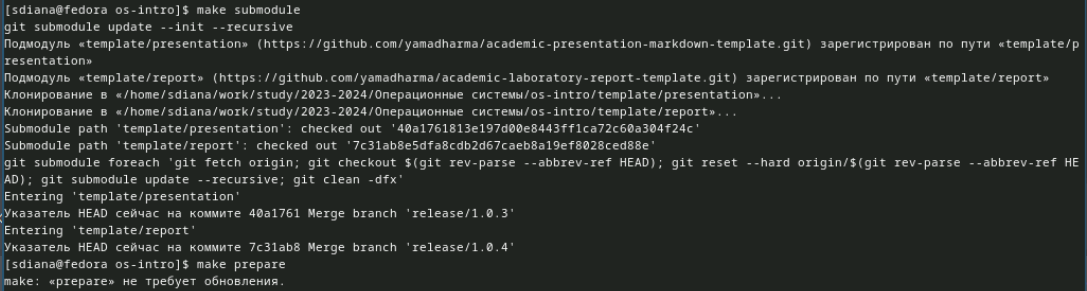


Рис. 24: Установка make submodule

По исструкции нужно было устанавить 3 разных каталога для дольнейшей работы

Отправьте файлы на сервер: (рис. 25).

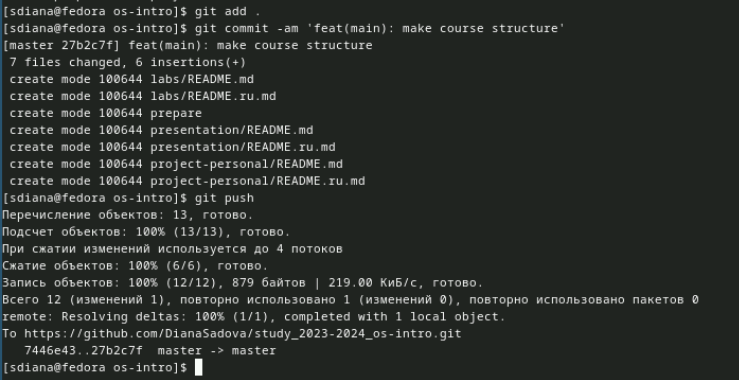


Рис. 25: Отправляем на Github

Завершаем нашу работу тем, что отправляем ее в наш репридиторий на Github

# 5 Выводы

Изучили идеологию и применение средств контроля версий. Освоили умения по работе с git.

# Список литературы