Отчет по лабораторной работе №2

Операционные системы

Ирина Васильевна Панявкина

Содержание

# 1 Цель работы

Цель данной лабораторной работы - изучение идеологии и применения средств контроля версий, освоение умения по работе с git.

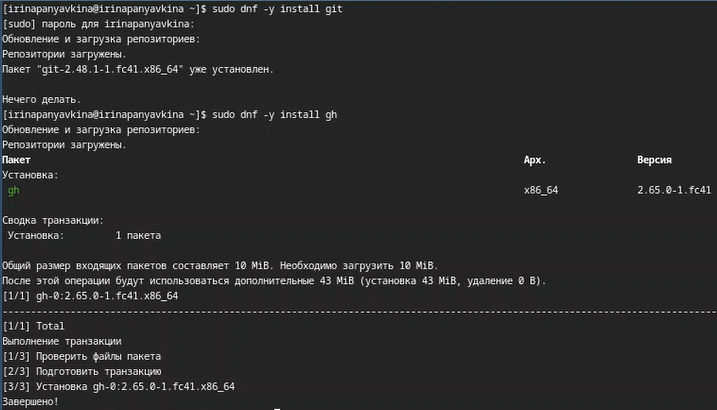
# 2 Задание

1. Создать базовую конфигурацию для работы с git
2. Создать ключ SSH
3. Создать ключ GPG
4. Настроить подписи Git
5. Зарегестрироваться на GitHub
6. Создать локальный каталог для выполнения заданий по предмету.

# 3 Выполнение лабораторной работы

## 3.1 Установка программного обеспечения

Устанавливаю необходимое программное обеспечение git и gh через терминал с помощью команд: dnf install git и dnf install gh (рис.[??]).



Установка git и gh

## 3.2 Базовая настройка git

Задаю в качестве имени и email владельца репозитория свои имя, фамилию и электронную почту (рис.[??]).

Задаю имя и email владельца репозитория

Задаю имя и email владельца репозитория

Настраиваю utf-8 в выводе сообщений git для их корректного отображения (рис.[??]).

Настройка utf-8 в выводе сообщений git

Настройка utf-8 в выводе сообщений git

Начальной ветке задаю имя master (рис.[??]).

Задаю имя начальной ветки

Задаю имя начальной ветки

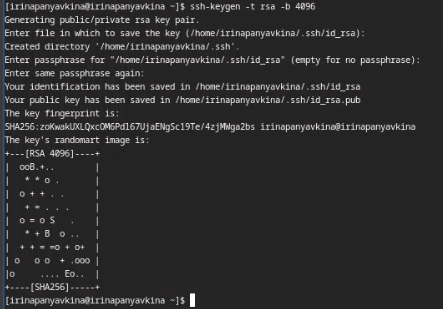
Задаю параметры autocrlf и safecrlf для корректного отображения конца строки (рис.[??]).

Задаю параметры autocrlf и safecrlf

Задаю параметры autocrlf и safecrlf

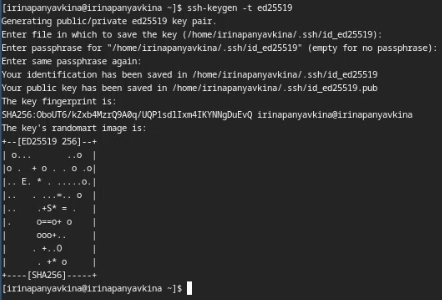
## 3.3 Создание ключа SSH

Создаю ключ ssh размером 4096 бит по алгоритму rsa (рис.[??]).



Генерация ssh ключа по алгоритму rsa

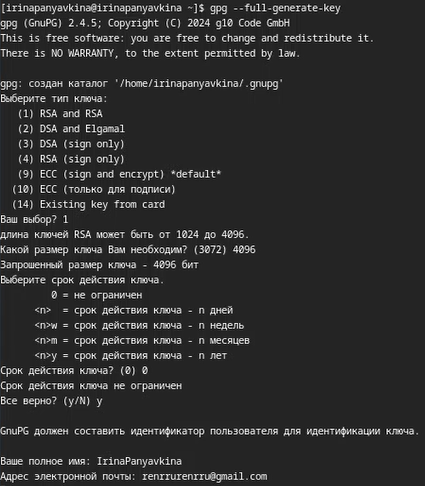
Создаю ключ ssh по алгоритму ed25519 (рис.[??]).



Генерация ssh ключа по алгоритму ed25519

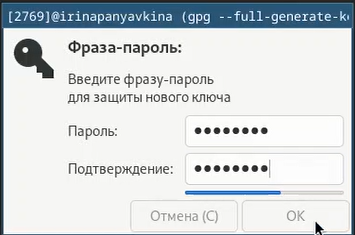
## 3.4 Создание ключа GPG

Генерирую ключ GPG, затем выбираю тип ключа RSA и RSA, задаю максимальную длину ключа 4096, оставляю неограниченный срок действия ключа. Затем даю ответы на вопросы программы о личной информации (рис.[??]).



Генерация ключа GPG

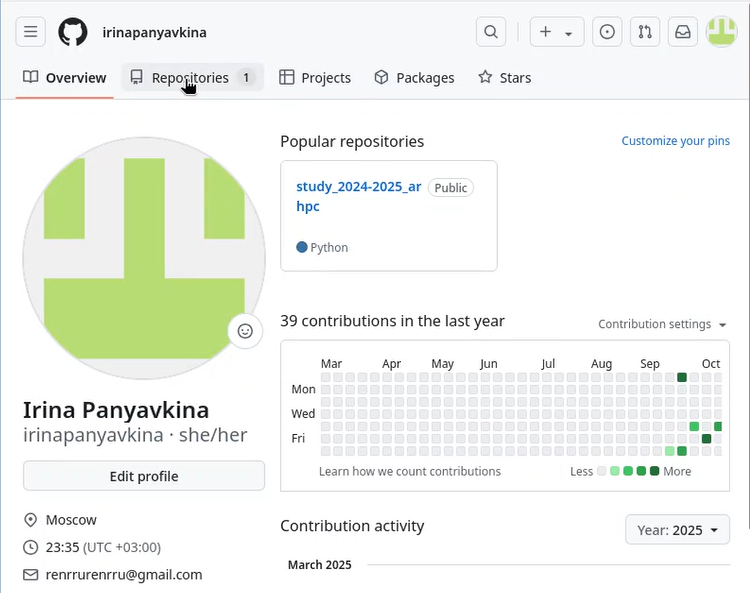
Ввожу фразу-пароль для защиты нового ключа (рис.[??]).



Защита ключа GPG

## 3.5 Регистрация на GitHub

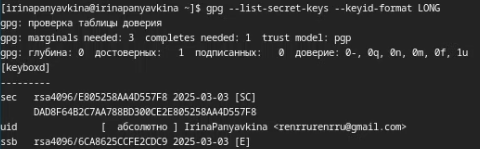
Аккаунт на GitHub я уже создавала, поэтому основные данные уже заполнены и проведена его настройка, тогда просто вхожу в свой аккаунт (рис.[??]).



Аккаунт на GitHub

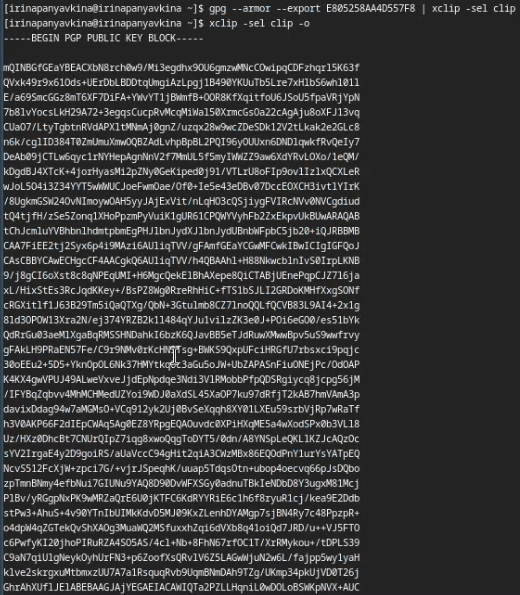
## 3.6 Добавление ключа GPG на GitHub

Вывожу список созданных ключей в терминал, ищу в результате запроса отпечаток ключа (последовательность байтов для идентификации более длинного, по сравнению с самим отпечатком, ключа), он находится после знака слеш, копирую его в буфер обмена.(рис.[??]).



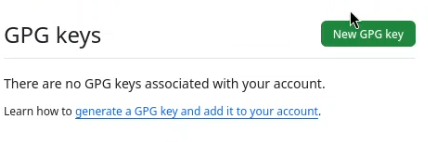
Вывод списка ключей

К сожалению, скопировать ключ с помощью утилиты xclip, введя следующую команду gpg –armor –export отпечаток ключа | xclip -sel clip, мне не удалось, как бы я не пыталась (происходила вставка предыдущего скопированного элемента), поэтому пришлось сделать это вручную с помощью команды для просмотра буфера обмена xclip -sel clip -o. (рис.[??]).



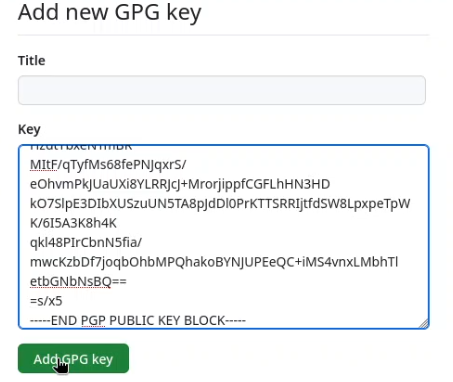
Вывод и копирование ключа в буфер обмена

Открываю настройки GitHub, ищу среди них добавление GPG ключа (рис.[??]).



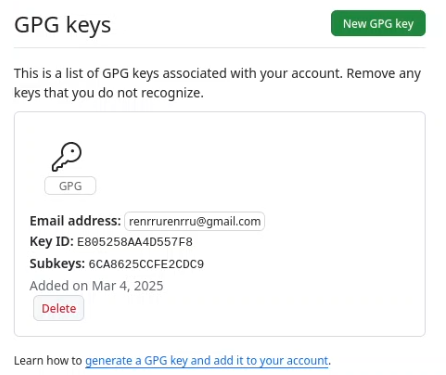
Настройки GitHub

Нажимаю на “New GPG key” и вставляю в поле ключ из буфера обмена (рис.[??]).



Добавление нового GPG ключа

Ключ GPG успешно добавлен на GitHub (рис.[??]).



Добавленный GPG ключ

## 3.7 Настроить подписи Git

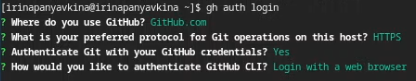
Настраиваю автоматические подписи коммитов git: используя введённый ранее email, указываю git использовать его при создании подписей коммитов (рис.[??]).

Настройка подписей Git

Настройка подписей Git

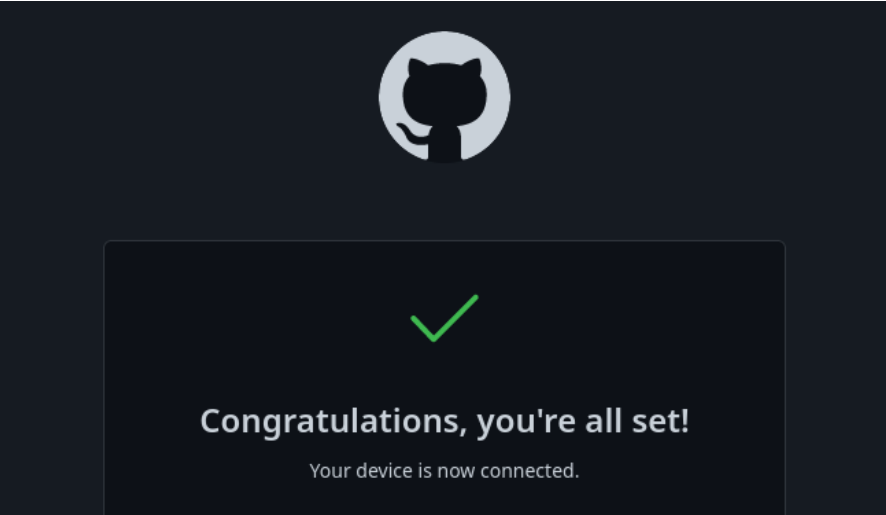
## 3.8 Настройка gh

Начинаю авторизацию в gh, отвечаю на наводящие вопросы от утилиты, в конце выбираю авторизоваться через браузер (рис.[??]).



Авторизация в gh

Завершаю авторизацию на сайте (рис.[??]).

.

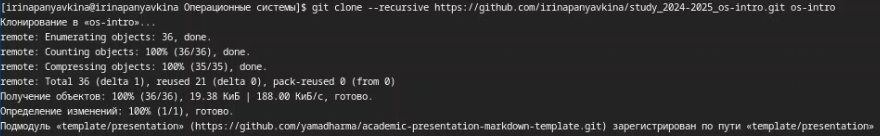
Вижу сообщение о завершении авторизации под именем irinapanyavkina (рис.[??]).

Завершение авторизации

Завершение авторизации

## 3.9 Создание репозитория курса на основе шаблона

Сначала я создаю директорию с помощью утилиты mkdir и флага -p, который позволяет установить каталоги на всем указанном пути. Затем с помощью утилиты cd перемещаюсь в созданную директорию “Операционные системы”. После этого в терминале ввожу команду gh repo create study\_2024-2025\_os-intro-template yamadharma/course =-directory-template –public, для тогог чтобы создать репозиторий на основе шаблона репозитория. Далее клонирую репозиторий к себе в директорию, указываю ссылку с протоколом https, а не ssh, потому что во время авторизации в gh, я выбрала протокол https (рис.[??]).



Создание репозитория

Перехожу в каталог курса с помощью утилиты cd, проверяю содержание каталога с помощью утилиты ls (рис.[??]).

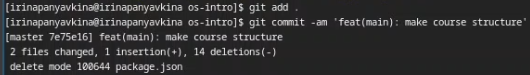
Перемещение между директориями

Перемещение между директориями

Удаляю лишние файлы с помощью утилиты rm, затем создаю необходимые каталоги используя makefile (рис.[??]).

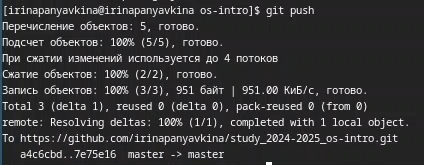
Удаление файлов и создание каталогов.

Добавляю все новые файлы для отправки на сервер (сохраняю добавленные изменения) с помощью команды git add и комментирую их с помощью команды git commit (рис.[??]).



Добавление и комментирование файлов для отправки на сервер

Отправляю файлы на сервер с помощью git push (рис.[??]).



Отправка файлов на сервер

# 4 Выводы

При выполнении лабораторной работы я изучила идеологию и применение средств контроля версий, освоила умение 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 ррименяются 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 себе необходимые 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 добавлять и 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.

# Список литературы

::: {#esystem.rudn.ru/mod/page/view.php?id=1224311}