Лабораторная работа №2

Опереционные системы

Софич Андрей Геннадьевич

Содержание

Список иллюстраций

1 Цель работы

Цель работы- изучение и применения средств контроля версии,освоение умения по работе с git.

2 Задание

1. Создание базовой конфигурации для работы с git.
2. Создание ключа SSH
3. Создание ключа GPG
4. Настройка подписи Git
5. Авторизация на GitHub
6. Создание репозитория
7. Ответы на вопросы

3 Выполнение лабораторной работы

3.1 Создание базовой конфигурации для работы с git

Устанавливаю программное обеспечение git и gh через терминал (рис. 3.1).

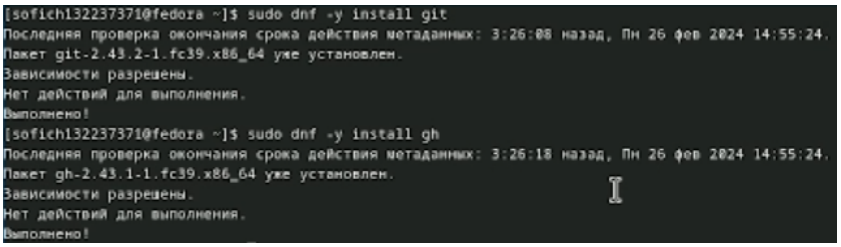

Рис. 3.1: Загрузка пакетов

Задаю имя и email владельца репозитория, настраиваю utf-8 для корректного отображения сообщения git. Задаю имя ветки. Задаю параметры autocrlf и safecrlf (рис. 3.1).

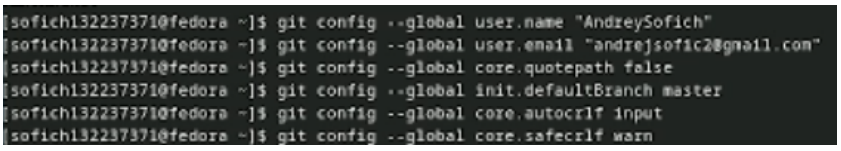

Рис. 3.2: Создание базовой конфигурации

##Создание ключа ssh

Создаю ключ ssh по алгоритму ed25519 (рис. 3.3).

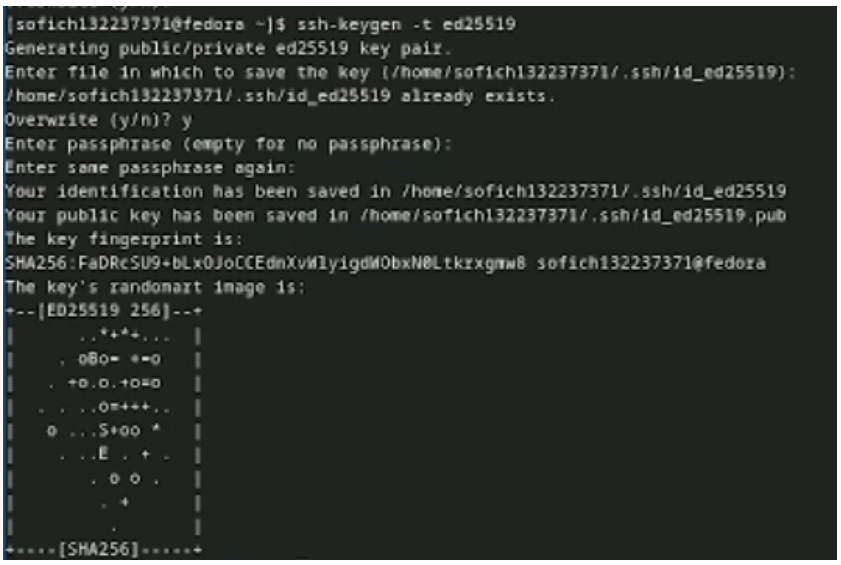

Рис. 3.3: Создание ssh ключа

Создаю ключ ssh по алгоритму rsa (рис. 3.4).

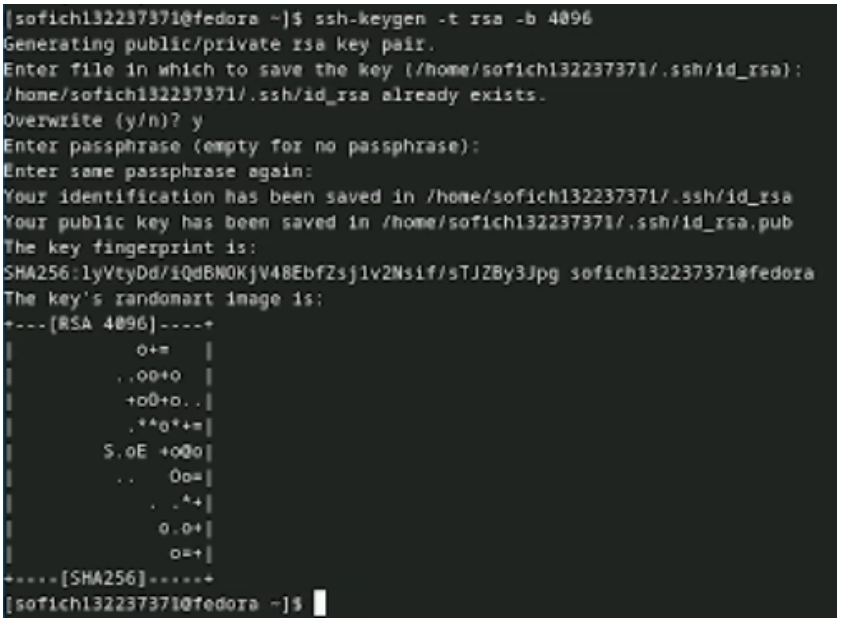

Рис. 3.4: Создание ssh ключа

Добавляю ключ в репозиторий на GitHub (рис. 3.5).

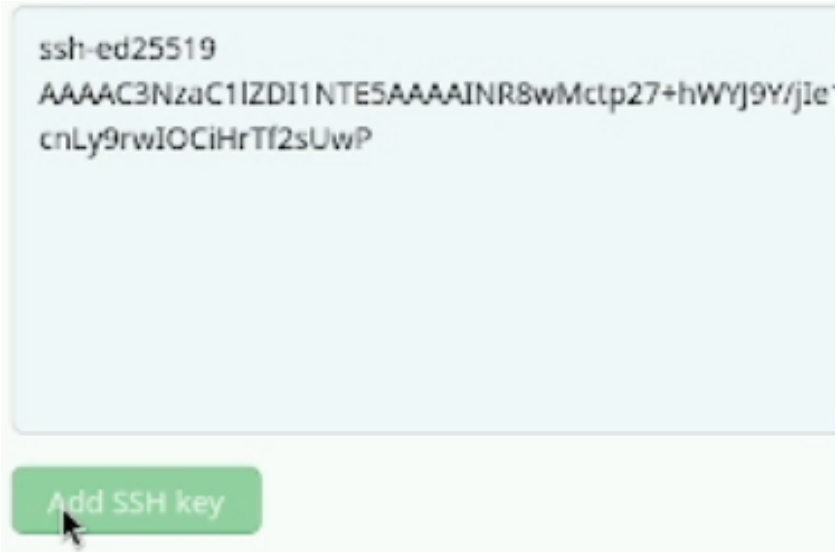

Рис. 3.5: Создание ssh ключа

##Создание ключа GPG

Генерирую ключ GPG, тип ключа RSA, задаю максимальную длину ключа 4096 и ставлю неограниченный срок действия ключа (рис. 3.6).

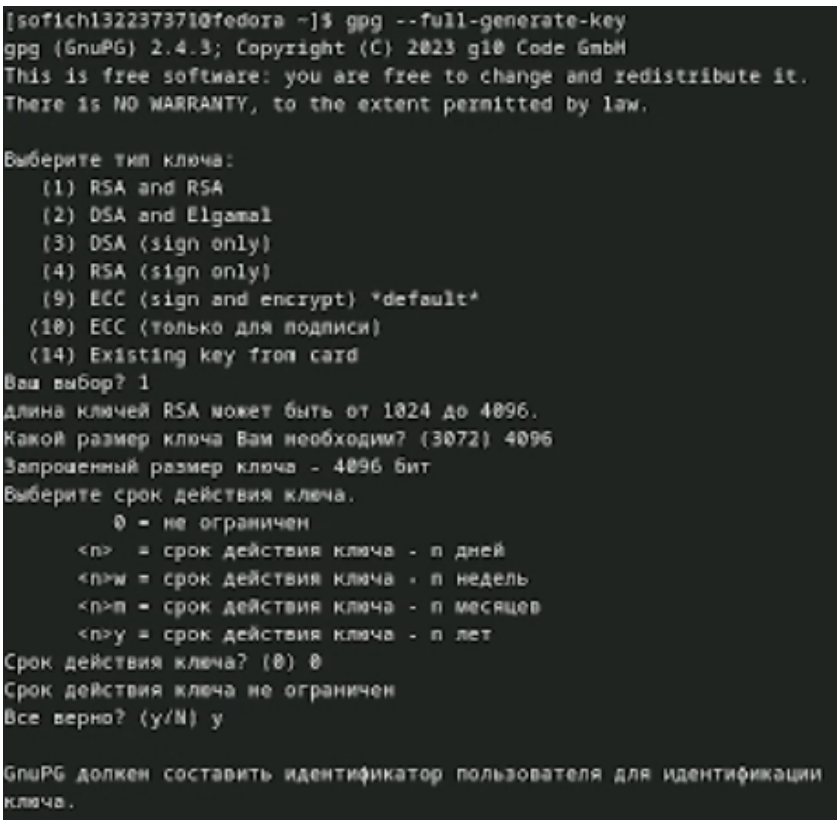

Рис. 3.6: Создание ключа GPG

Вывожу список созданных ключей в терминал, ищу отпечаток ключа и копирую его {#fig:007 width=70%}

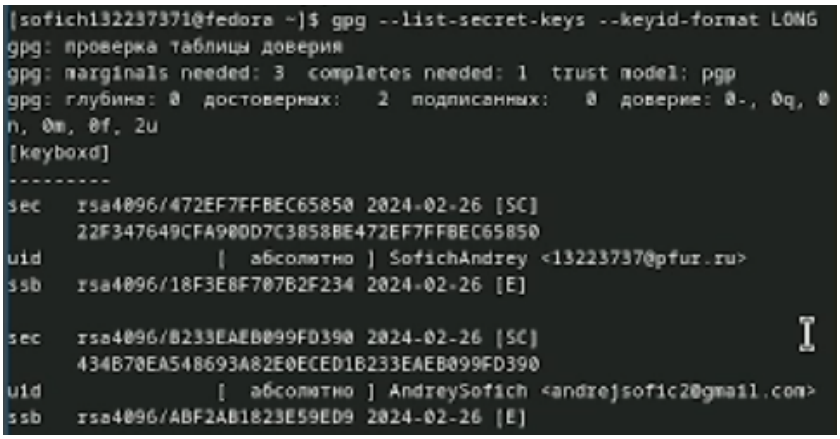

Рис. 3.7: Создание ключа GPG

Копирую ключ {#fig:008 width=70%}

Рис. 3.8: Копирование ключа GPG

Добавляю ключ GPG на GitHub {#fig:009 width=70%}

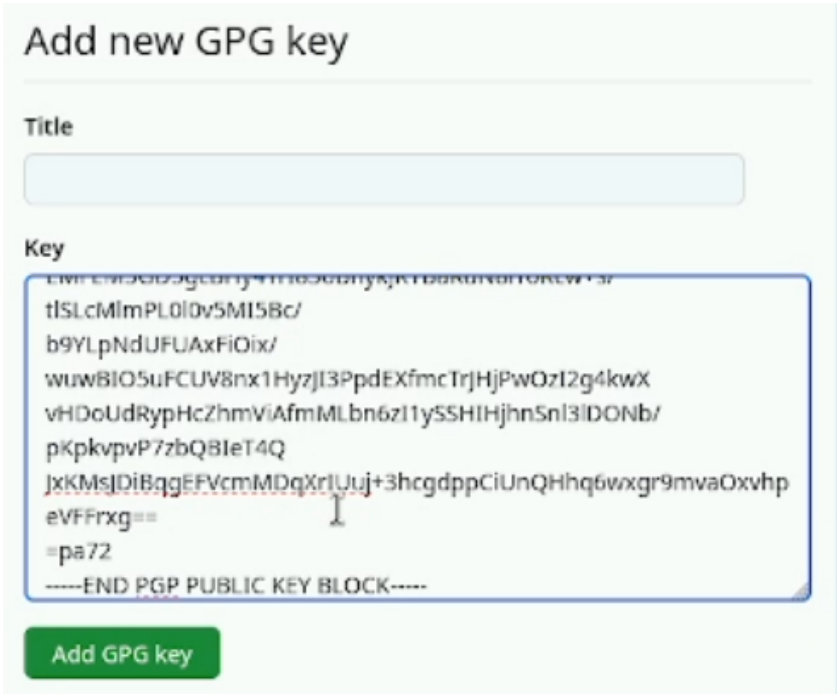

Рис. 3.9: Добавление GPG

3.2 Настройка подписи Git

Настраиваю автоматические подписи коммитов git: используя введенный ранее email {#fig:010 width=70%}

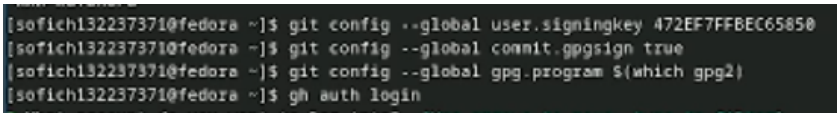

Рис. 3.10: Настройка подписей

3.3 Авторизация на GitHub

Завершаю авторизацию на сайте через браузер {#fig:011 width=70%}

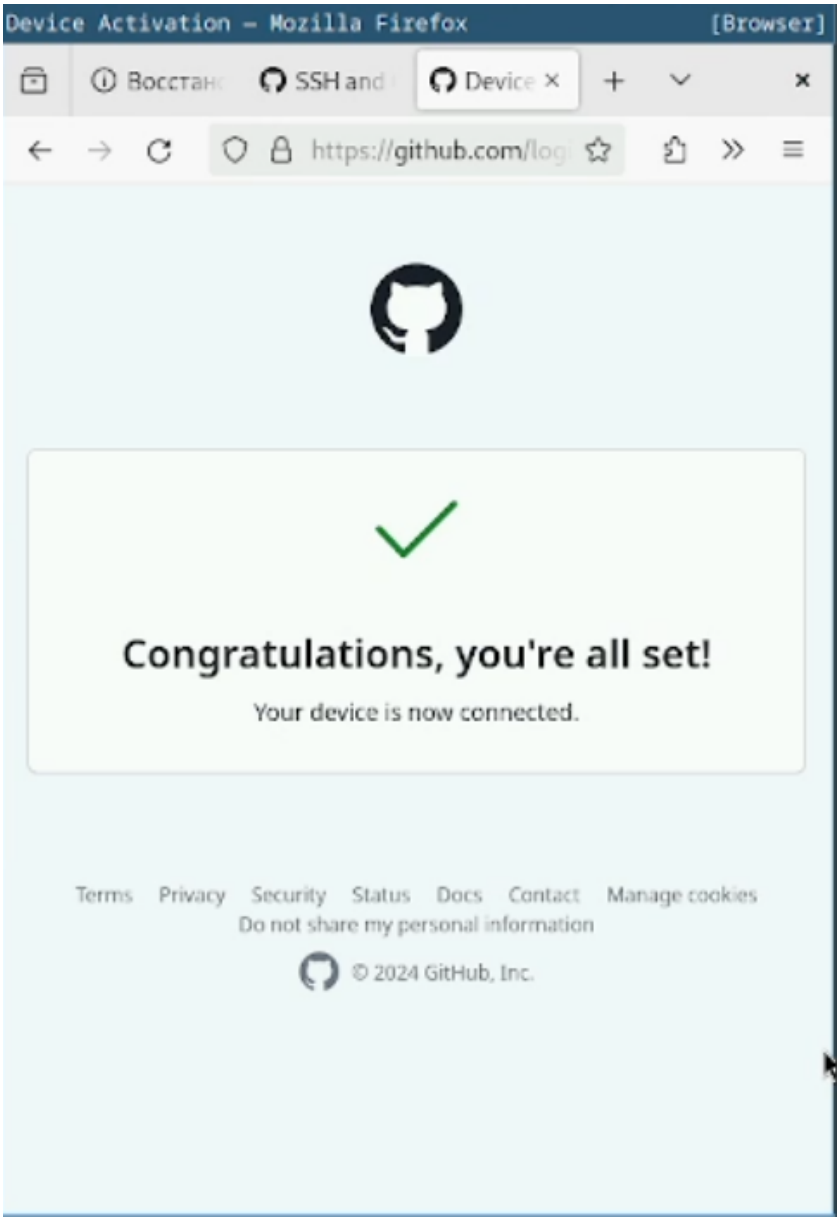

Рис. 3.11: Авторизация

##Создание репозитория

Перехожу в созданную директорию {#fig:012 width=70%}

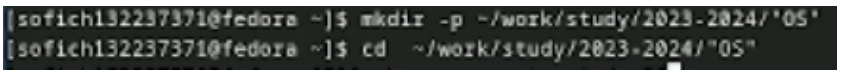

Рис. 3.12: Создание директории

Создаю репозиторий на основе шаблона {#fig:013 width=70%}

Рис. 3.13: Создание репозитория

Клонирую репозиторий в свою директорию {#fig:014 width=70%}

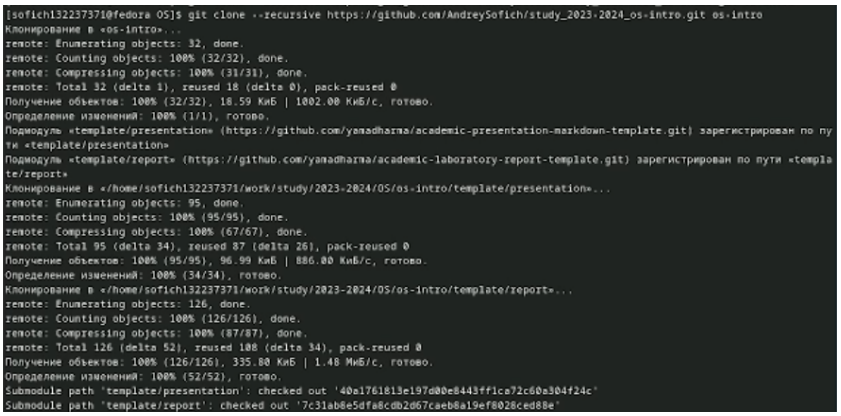

Рис. 3.14: Клонирование репозитория

Перехожу в каталог и проверяю наличие нужных папок {#fig:015 width=70%}

Рис. 3.15: Проверка репозитория

Удаляю ненужные файлы и создаю необходимые {#fig:016 width=70%}

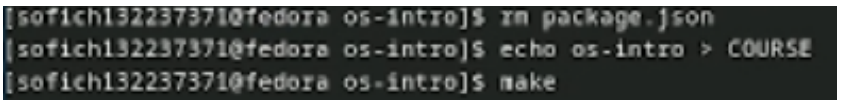

Рис. 3.16: Удаление файлов и создание каталогов

Загружаю всё на GitHub {#fig:016 width=70%}

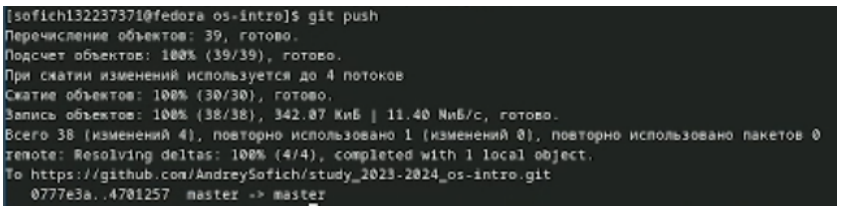

Рис. 3.17: Загрузка

##Ответы на 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 (децентрализованных) 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. Можно прописать шаблоны игнорируемых при добавлении в репозиторий типов файлов в файл .gitignore с помощью сервисов.

4 Выводы

При выполнении работы я изучил применение средств контроля версий и освоил умение по работе с git