Отчёт по лабораторной работе №2

Операционные системы

Пономарева Татьяна Александровна

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Изучить идеологию и применение средств контроля версий и 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.

4 Выполнение лабораторной работы

4.1 Установка программного обеспечения

Сначала устанавливаю git, потом gh (рис. 4.1).

Рис. 4.1: Установка git и gh

4.2 Базовая настройка git

Задаю имя и email владельца репозитория, настраиваю utf-8 в выводе сообщений git (рис. 4.2).

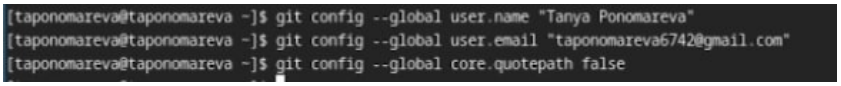

Рис. 4.2: Задание имени владельца и email, настройка utf-8

4.3 Верификация коммитов с помощью PGP

4.3.1 Создание ключа

Генерирую ключ, задаю следующие параметры: RSA и RSA, размер 4096, срок действия бессрочен, имя: tanya, email: taponomareva6742@gmail.com, comment: test (рис. 4.3).

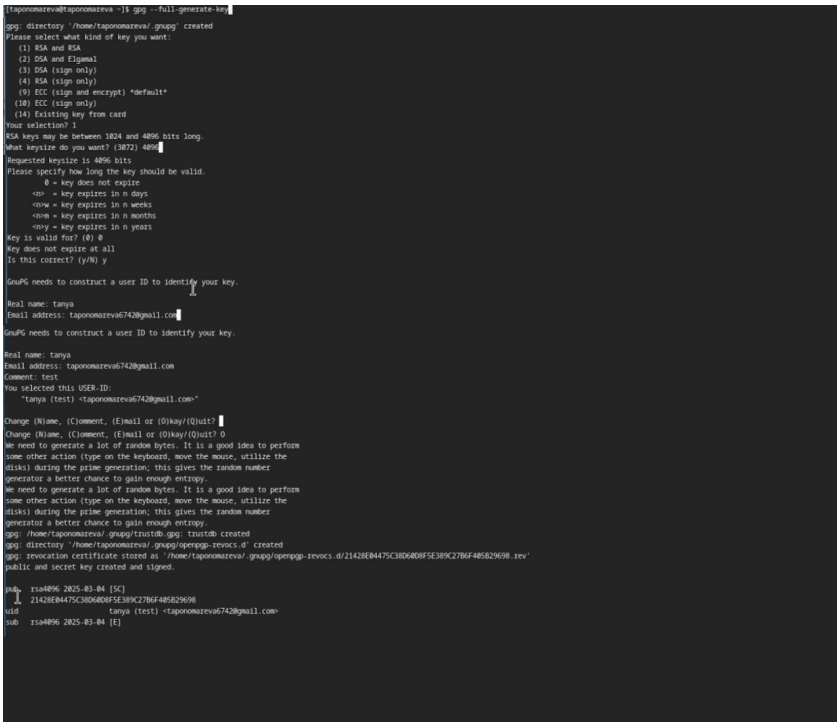

Рис. 4.3: Генерация ключа

4.3.2 Экспорт ключа

Вывожу список ключей (рис. 4.4).

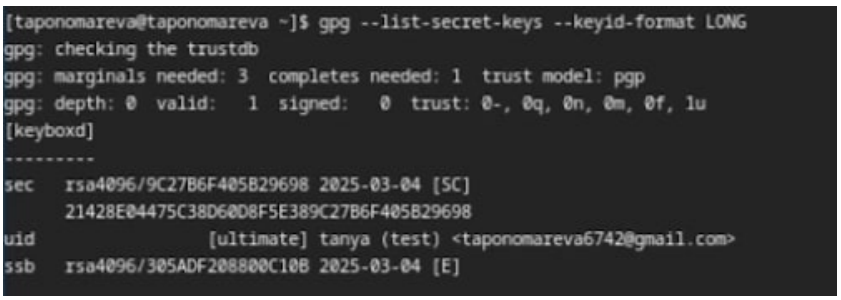

Рис. 4.4: Вывод списка ключей

Экспортирую ключ по его отпечатку (рис. 4.5).

Рис. 4.5: Экспорт ключа

4.3.3 Добавление PGP ключа в GitHub

Копирую ключ и добавляю его в настройках профиля на GitHub (рис. 4.6).

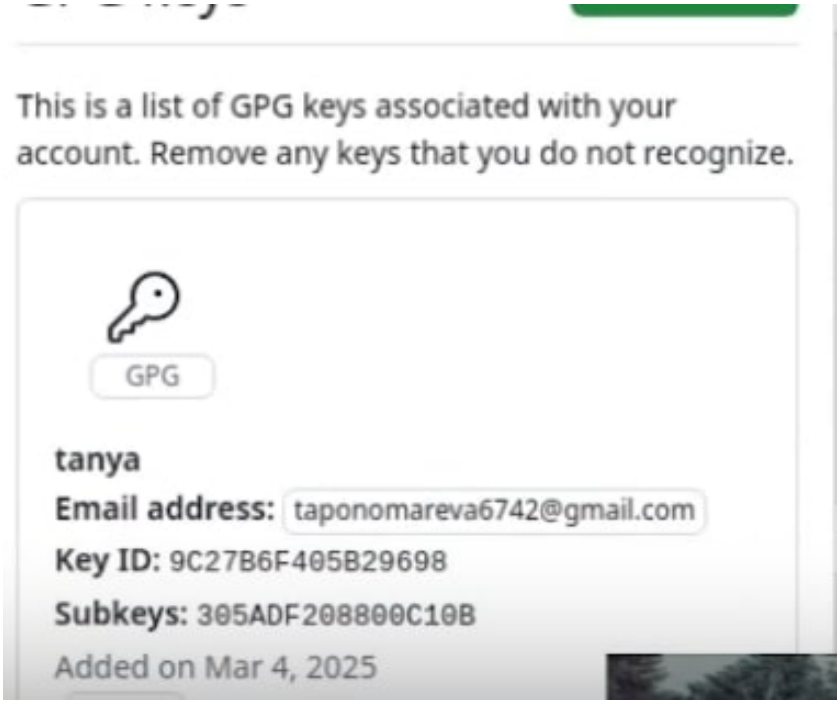

Рис. 4.6: Добавление ключа в GitHub

4.4 Базовая настройка git

Задаю имя начальной ветки (master) (рис. 4.7).

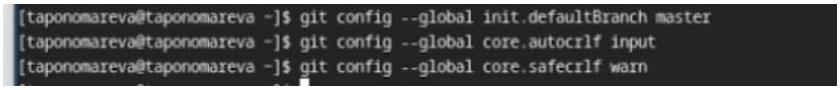

Рис. 4.7: Настройка git

4.5 Создание ключа ssh

Создаю ключ ssh по алгоритму rsa с ключем размером 4096 бит(рис. 4.8).

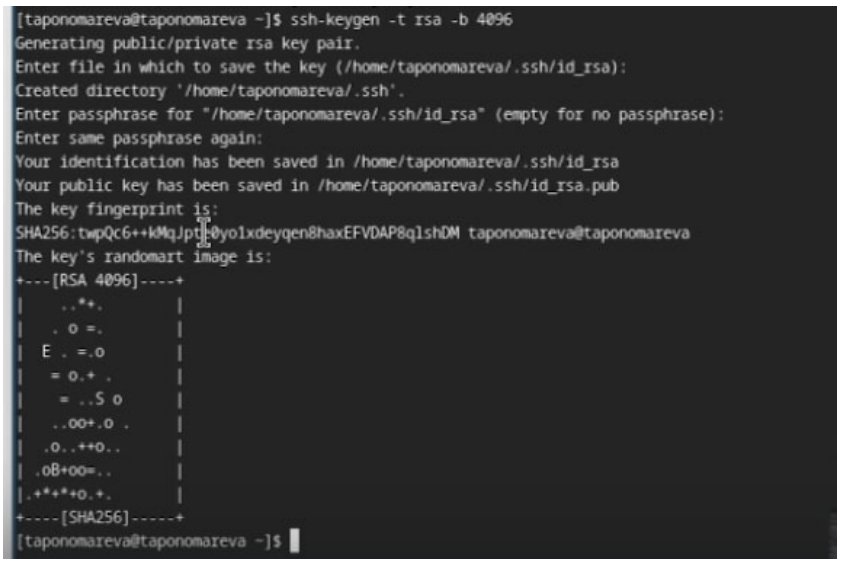

Рис. 4.8: Ключ ssh

Создаю ключ ssh по алгоритму ed25519(рис. 4.9).

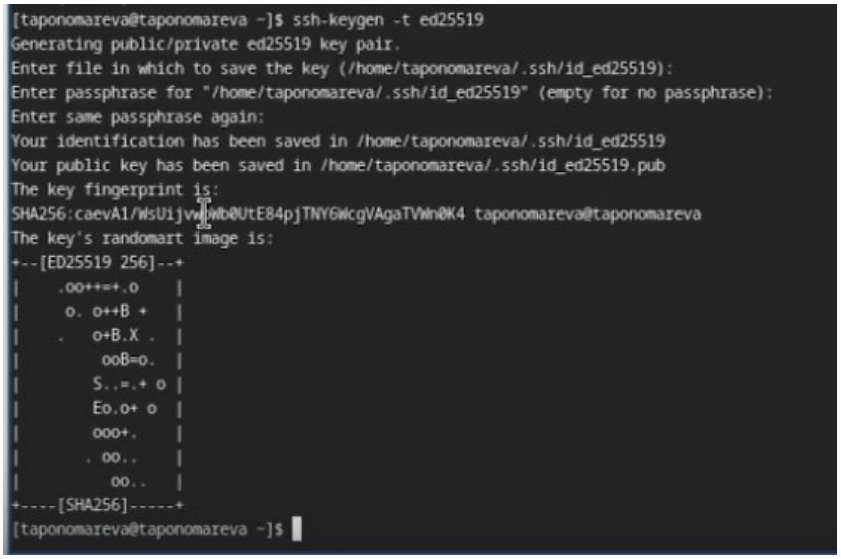

Рис. 4.9: Ключ ssh

Ключ был добавлен в GitHub (рис. 4.10).

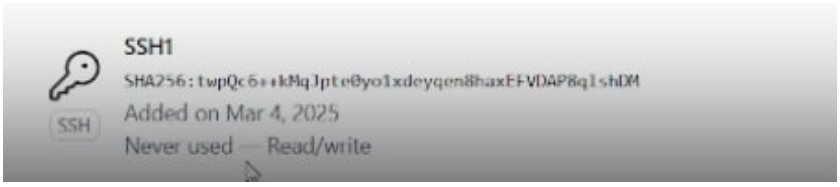

Рис. 4.10: Добавление ключа в GitHub

4.6 Настройка gh

Авторизовываюсь через команду gh auth login (рис. 4.11).

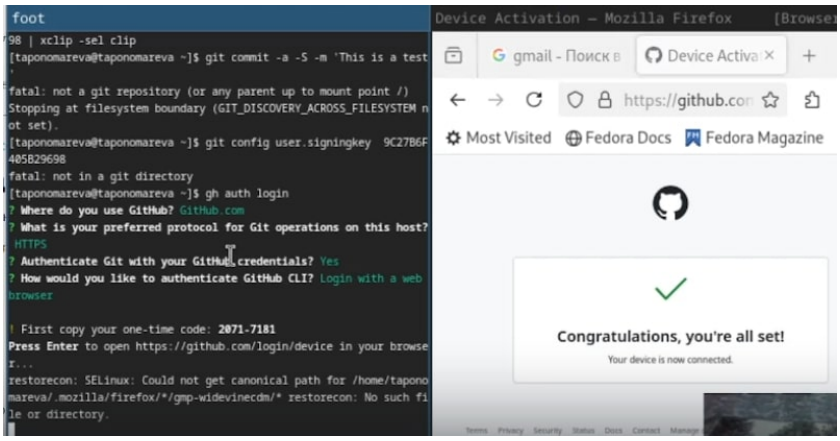

Рис. 4.11: Настройка git

4.7 Настройка автоматических подписей коммитов git

Используя введенный email, указываю Git применять его при подписи коммитов (рис. 4.12).

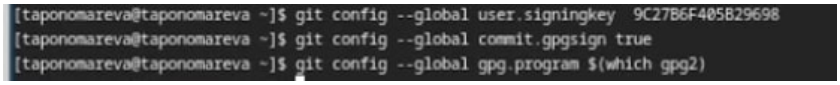

Рис. 4.12: Настройка подписей коммитов git

4.8 Создание репозитория курса на основе шаблона

Создаю репозиторий курса на основе шаблона (рис. 4.13).

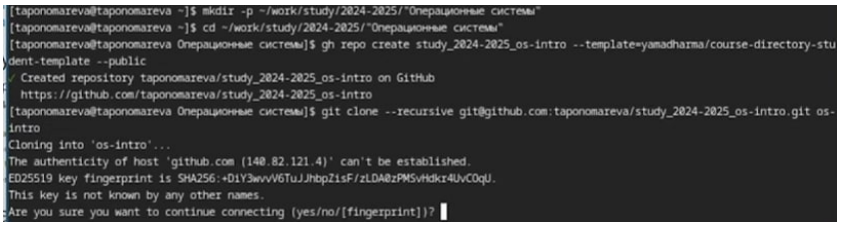

Рис. 4.13: Создание репозитория

4.9 Настройка каталога курса

Перехожу в каталог и удаляю лишние файлы, затем создаю необходимые каталоги, далее отправляю файлы на сервер (рис. 4.14).

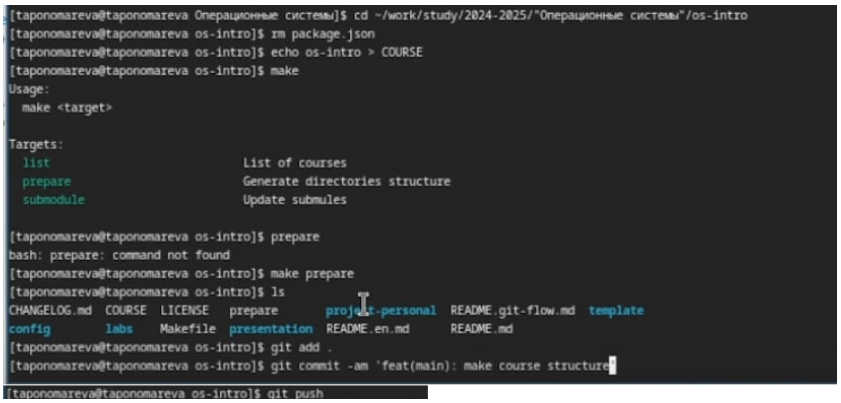

Рис. 4.14: Настройка каталога курса

5 Контрольные вопросы

1. Системы контроля версий (VCS) — это инструменты, предназначенные для отслеживания изменений в файлах или наборе файлов. Они позволяют сохранять версии данных и управлять ими, что важно при разработке программного обеспечения, написании текстов или любых других длительных проектных работах, требующих отслеживания изменений. VCS помогают работать с историей изменений, сотрудничать с другими людьми и восстанавливать предыдущие версии файлов.

2. Пояснение понятий VCS:

- Хранилище (repository) — это место, где хранится вся информация о проекте, включая файлы и историю изменений.
- Commit — это операция, при которой изменения в рабочих файлах фиксируются в хранилище с добавлением сообщения, объясняющего изменения.
- История (history) — это последовательность коммитов, представляющая изменения, которые были сделаны в проекте за время его существования.
- Рабочая копия (working copy) — это локальная версия файлов проекта, с которыми пользователь работает. Изменения сначала происходят в рабочей копии и затем фиксируются в хранилище с помощью коммита.

3. Централизованные и децентрализованные VCS:

- Централизованные VCS предполагают наличие одного центрального хранилища, к которому подключаются все пользователи. Все изменения сохраняются в этом едином месте. Пример: Subversion (SVN).

- Децентрализованные VCS позволяют каждому пользователю иметь локальное хранилище, синхронизируя изменения между репозиториями. Каждый участник может работать автономно, а затем обмениваться данными с другими. Пример: Git.

4. Примеры VCS:

- Централизованные: Subversion (SVN), CVS.
- Децентрализованные: Git, Mercurial.

5. Действия с VCS при единоличной работе:

Инициализация репозитория (создание нового хранилища). Изменение файлов в рабочей копии. Добавление новых файлов в индекс с помощью команды git add. Фиксация изменений в хранилище с помощью git commit. Просмотр истории изменений с помощью git log.

6. Порядок работы с общим хранилищем VCS:

Клонирование репозитория с удалённого сервера с помощью git clone. Синхронизация с удалённым хранилищем с помощью git fetch или git pull. Редактирование локальных файлов и фиксация изменений через git commit. Отправка изменений в удалённое хранилище через git push. Разрешение конфликтов при слиянии изменений с помощью git merge.

7. Основные задачи, решаемые инструментальным средством Git:

Отслеживание изменений в проекте. Совместная работа с другими пользователями. Ветвление и слияние изменений (работа с ветками). Управление удалёнными репозиториями и синхронизация изменений. Восстановление предыдущих версий файлов и проекта.

8. Команды Git:

git init — инициализация нового репозитория. git clone — клонирование удалённого репозитория. git add — добавление изменений в индекс (подготовка файлов к коммиту). git commit -m “message” — фиксация изменений в репозитории с сообщением. git status — просмотр текущего состояния рабочей копии. git log — просмотр истории коммитов. git push — отправка изменений в удалённый репозиторий. git pull — получение изменений с удалённого репозитория. git branch — создание и управление ветками.

9. Примеры использования Git с локальными и удалёнными репозиториями:

Локальный репозиторий: git init — инициализация репозитория, git commit — фиксация изменений, git status — проверка состояния. Удалённый репозиторий: git clone — клонирование удалённого репозитория, git push — отправка изменений на сервер, git pull — получение обновлений с сервера.

10. Ветви в Git позволяют изолировать изменения в проекте, не влияя на основную рабочую версию. Это полезно для параллельной разработки новых функций, исправлений ошибок или экспериментов, которые затем могут быть объединены в основную ветку (например, main или master). Ветки позволяют легко работать с несколькими направлениями разработки и избежать конфликтов между различными задачами.

6 Выводы

Была изучена идеология и применение средств контроля версий и освоены умения по работе с git

Список литературы

1. Курс на ТУИС
2. Лабораторная работа №2