Отчёт по лабораторной работе 2

Дисциплина Архитектура Компьютеров и Операционные Системы

Гомес Лопес Теофания

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Изучение идеалогии, применение средств контроля версий и освоение 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 Выполнение лабораторной работы

3.1 Создание базовой конфигурации для работы с git.

Установливаю git используя “dnf install git”:

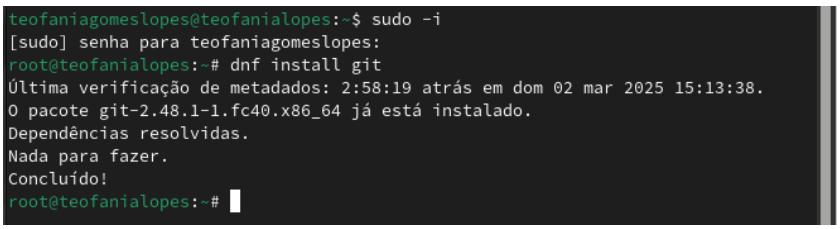

Рис. 3.1: Установление git

С помощью dnf install gh, установливаю gh:

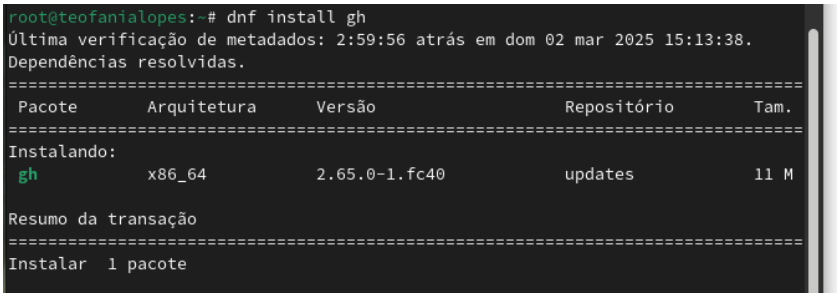

Рис. 3.2: Установление gh

В качестве имя и email владельца репозитории задаю свои имя и email и настраиваю utf-8:

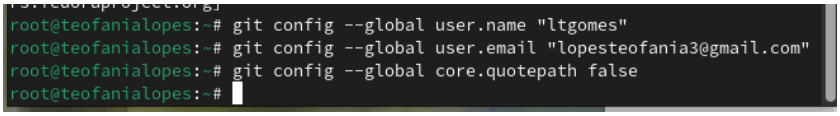

Рис. 3.3: Имя и email владельца

Задаю имя начальной ветки и паррамеры autocrlf и safecrlf:

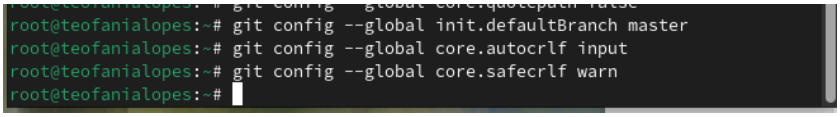

Рис. 3.4: Имя начальной ветки и паррамеры

3.2 Создание ключ SSH:

Создаю ключи ssh по алгоритму rsa с размером 4096 бит:

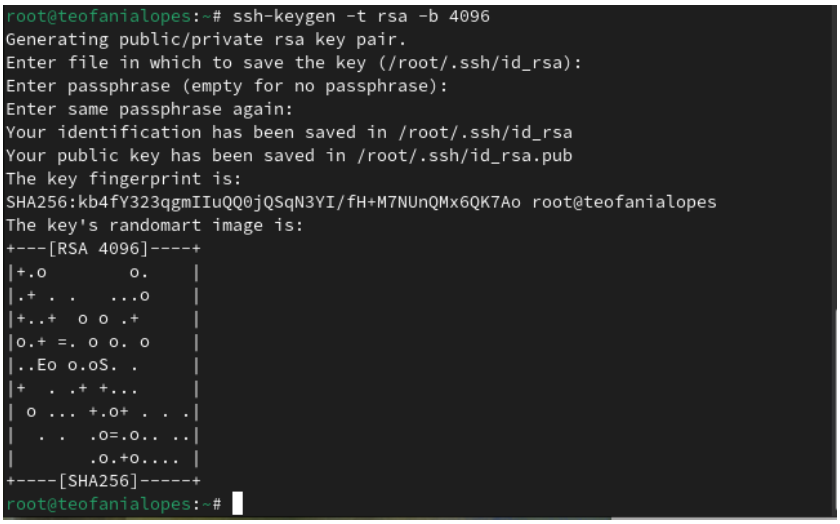

Рис. 3.5: Создание ключ ssh

3.3 Создание ключ gpg

Генерирую ключ gpg –full-generate-key:

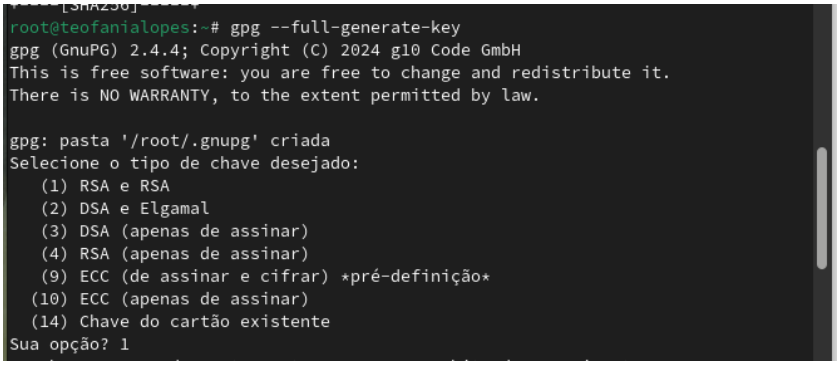

Рис. 3.6: Создание ключ gpg

Из предложенных опций выбираю тип RSA and RSA; размер 4096; срок действия 0:

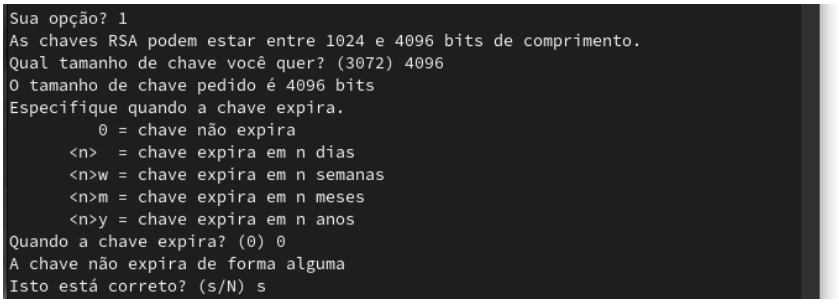

Рис. 3.7: Настройки ключ gpg

GPG запросил личную информацию, которая сохранится в ключе Имя и адрес электронной почты:

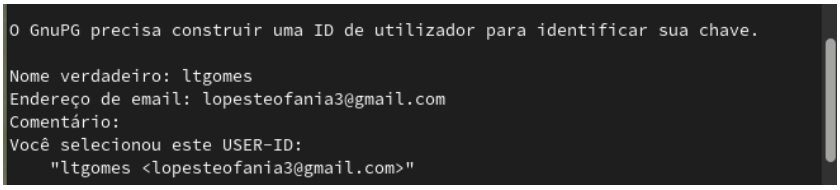

Рис. 3.8: личная информация

У меня уже есть аккаунт на github, поэтому я вхожу в систему:

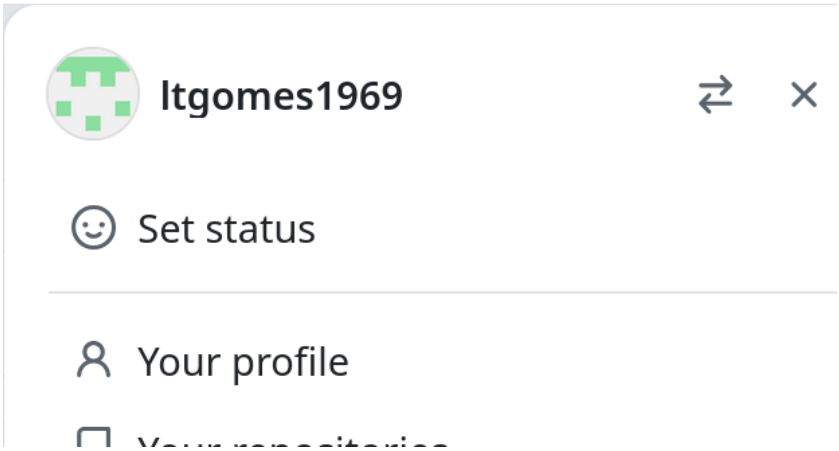

Рис. 3.9: аккаунт на git

Вывожу список ключей:

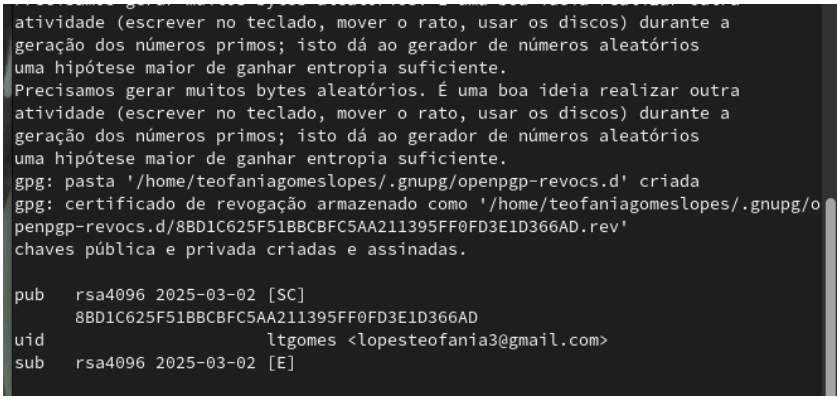

Рис. 3.10: список ключей

Установливаю xclip:

Рис. 3.11: Установление xclip

Скопирую сгенерированный gpg ключ в буфер обмена:

Рис. 3.12: Копирование ключ gpg

Далее перехожу в настройки GitHub, нажимаю на кнопку New GPG key и вставляю полученный ключ:

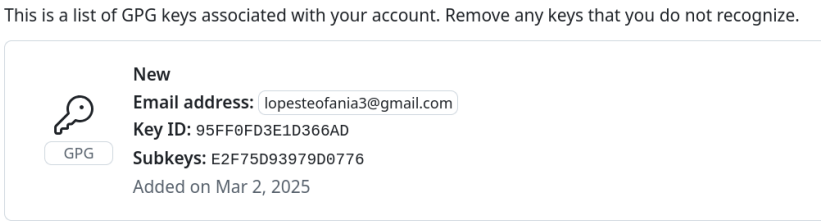

Рис. 3.13: Добавлен ключ gpg

Используя введёный email, указываю Git применять его при подписи коммитов:

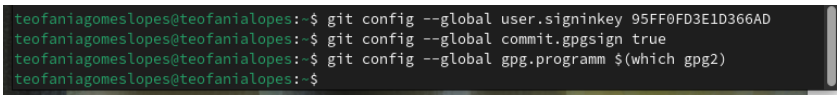

Рис. 3.14: указиваю Git

Начинаю авторизацию в gh используя gh auth login:

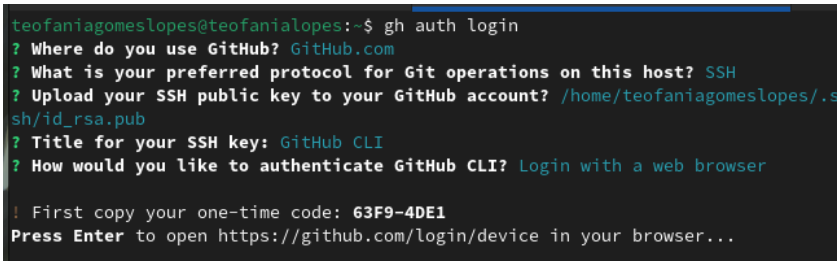

Рис. 3.15: авторизацию в gh

Завершаю авторизацию на броузер:

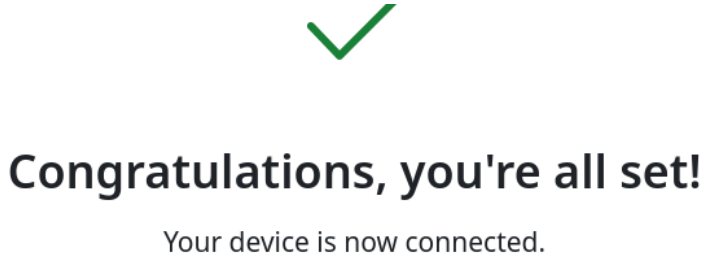

Рис. 3.16: Авторизоваться через броузер

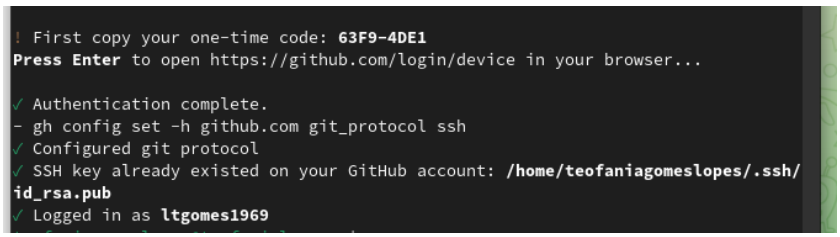

Рис. 3.17: Завершена авторизация

3.4 Создание локального каталога для выполнения заданий.

Создаю каталог “mkdir -p ~/work/study/2022-2023/”Операционные системы”:

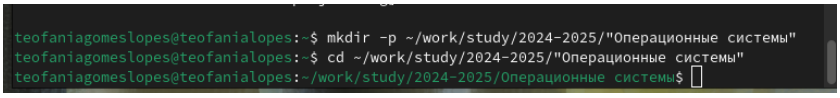

Рис. 3.18: Создание каталог

Перехожу в созданный каталог:

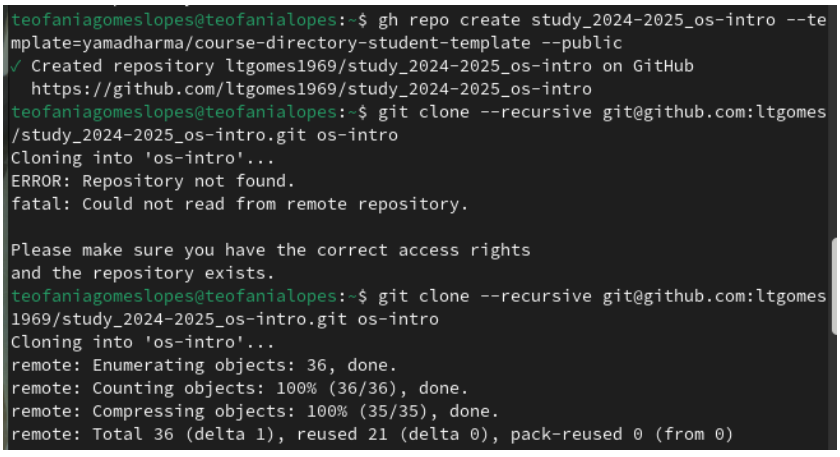

Рис. 3.19: Создание каталог

Удаляю лишные файлы:

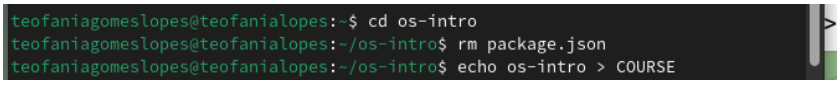

Рис. 3.20: Удаление файла

Создаю еще необходимые каталоги:

```
teofaniagomeslopes@teofanialopes:~/os-intro$ make
Usage:
  make <target>

Targets:
  list                           List of courses
  prepare                        Generate directories structure
  submodule                      Update submules
```

Рис. 3.21: Создани необходимых каталогов

Отправляю Файлы на сервер:

```
teofaniagomeslopes@teofanialopes:~/os-intro$ git add .
teofaniagomeslopes@teofanialopes:~/os-intro$ git commit -am 'feat(main): make co
urse structure'
[master 9fa3129] feat(main): make course structure
 2 files changed, 1 insertion(+), 14 deletions(-)
 delete mode 100644 package.json
teofaniagomeslopes@teofanialopes:~/os-intro$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 948 bytes | 948.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:ltgomes1969/study_2024-2025_os-intro.git
   afeca9b..9fa3129  master -> master
teofaniagomeslopes@teofanialopes:~/os-intro$
```

Рис. 3.22: Отправление файлы на сервер

4 Выводы

При выполнении лабораторной работы я изучила идеалогию, применение средств контроля версий и освоеила умение по работе с git.

5 Ответы на контрольные вопросы

1. Системы Контроля Версий - Программные инструменты, помагающие командам разработчиков управлять изменениями в исходном коде с течением времени.

2. Хранилище - в нем храняются все документы, включая историю их изменение и прочей служебной информацией.

   commit - отслеживание изменениййб сохраняет разницу в изменениях.

   история - Хранит все изменения в проекте и позволяет при необходимости обратиться к нужным данным.

   рабочая копия- копия проекта основанная на версии из хранилища.

3. В ценрализованном VCS например AccuRev, каждый пользователь копирует себе необходимые ему файлы из репзитория, изменяет их а затем добавляет измениения обратно в хранилище. В децентрализованном VCS например Git, есть возможность добавлять и забирать изменения из любого репозитория.

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.

Создание основного дерева репозитория: git init

7. 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.

Список литературы