# Лабороторная работа № 2

**Операционные системы**

Четвергова Мария Викторовна

# Содержание

# Список иллюстраций

# Список таблиц

# 1 Цель работы

Целью данной работы является изучение идеологии и применение средств контроля версий, освоение умения по работе с git

# 2 Задание

В ходе выполнения лпбораторной работы №2 необходимо выполнить следующие задания: - создать базовую конфигурацию для работы с git - создать ключ SSH - Создать ключ PGP - Настроить подписи git - Зарегистрироваться на GitHub - Создать локальный каталог для выполнения заданий по предмету

[1] # Выполнение лабораторной работы ## Установка программного обеспечения

1. Установка Git Установим git с помощью команды, вбиваемой в терминал:

*git install git*

2. Установка gh Установим gh с помощью вбиваемов в терминал команды:

*git install gh*

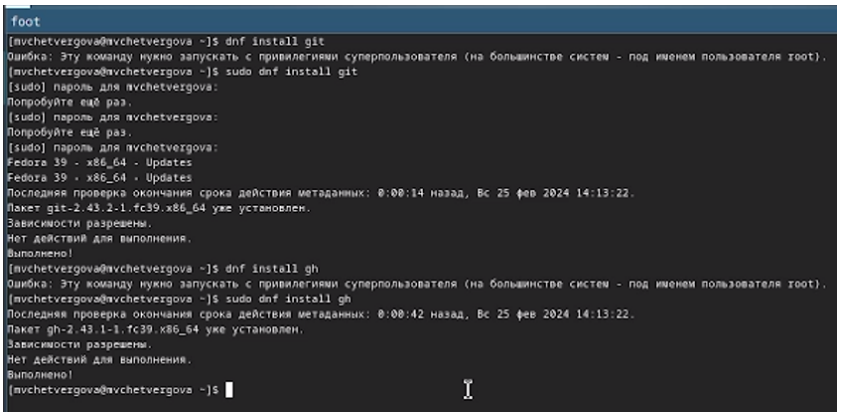


Рис. 2.1: Установка программного обеспечения

## 2.1 Базовая настройка Git

Проведём базовую настройку git: для этого необходимо задать имя владельца, емейл владельца, а также настроить utf-8 в выводе сообщений git. При работе с этими задачами поспользуемся командами типа *git config –global …*

*git config –global user.name “Maria02-23 git config –global user.email”my_email.com*

Необходимо и настроить верификацию и подписание коммитов git, а также задать имя начальной ветки(master). Настроим параметры autocrlf и safecrlf. осуществить эти действия возможно с помощью команд командной строки:

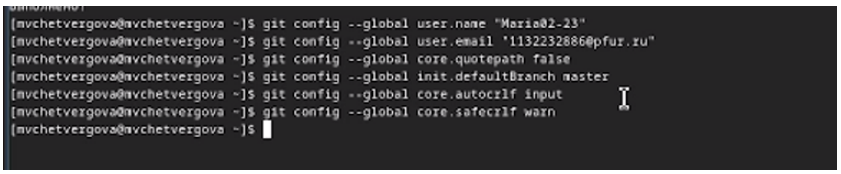


Рис. 2.2: Базовая настройка git

## 2.2 Создание ключа SSH

В инструкции к лабороторной работе указаны два способа создания SSH. Воспользуемся обоими.

1. по алгоритму rsa с ключём размером 4096 бит:

*ssh-keygen -t rsa -b 4096*

2. По алгоритму ed25519:

*ssh-keygen -t ed25519*


Рис. 2.3: Создание ключа SSH

## 2.3 Создание ключа PGP

1. Генерируем ключ с помощью команды в терминале:

*gpg –full-generate-key*

2. На мониторе появится несколько вариантов ответов. Для того, чтобы успешно создать ключ, необходимо выбрать следующие варианты:

- тип RSA
- размер 4096
- срок действия - бессрочный (0) (далее gpg запрашивает личную информацию: сперва вводим своё полное имя, а затем - емейл, привязанный к

аккаунту на GitHub. В конце можно ввести комментарий, но это уже неважно - можно вводить что угодно)

После заполнения информации на экране всплывает окошко с требованием заполнить пароль-фразу. Пропускаем этот этап, выбрав “защита не нужна”


Рис. 2.4: Создание PGP ключа

## 2.4 Настройка GitHub

В инструкции к лабороторной работе сказано, что необходимо создать учётную запись на сайте GitHub и заполнит основные данные. В первом семестре мы уже имели дело с системой git, поэтому аккаунт на GitHub у меня уже есть:

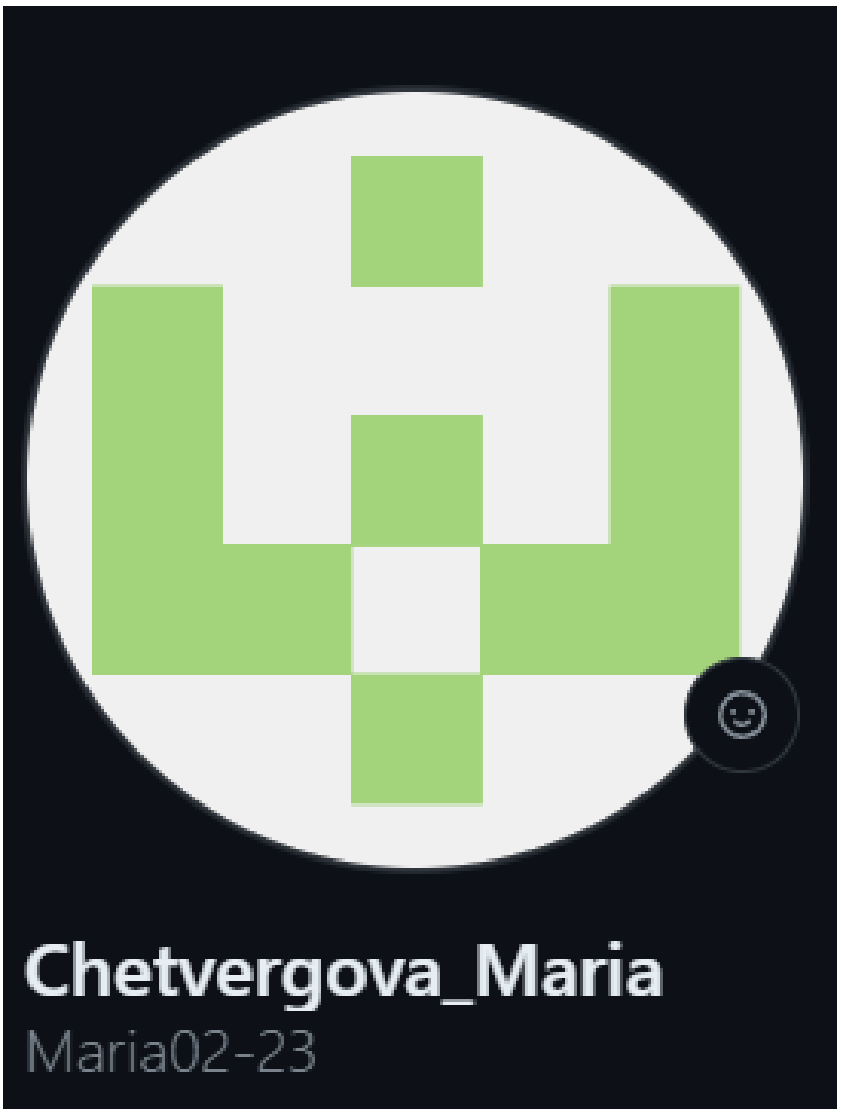


Рис. 2.5: Аккаунт на сайте GitHub

## 2.5 Добавление PGP ключа в GitHub

Для добавления PGP ключа в GitHub необходимо выполнить следующие действия:

- выведем список ключей и скопируем отпечаток приватного ключа с помощью команды в терминале:

*gpg –list-secret-keys –keyid-format LONG*

- Далее нужно скопировать сгенерированный PGP ключ в буфер обмена командой

*gpg –armor –export | xclip -sel clip*

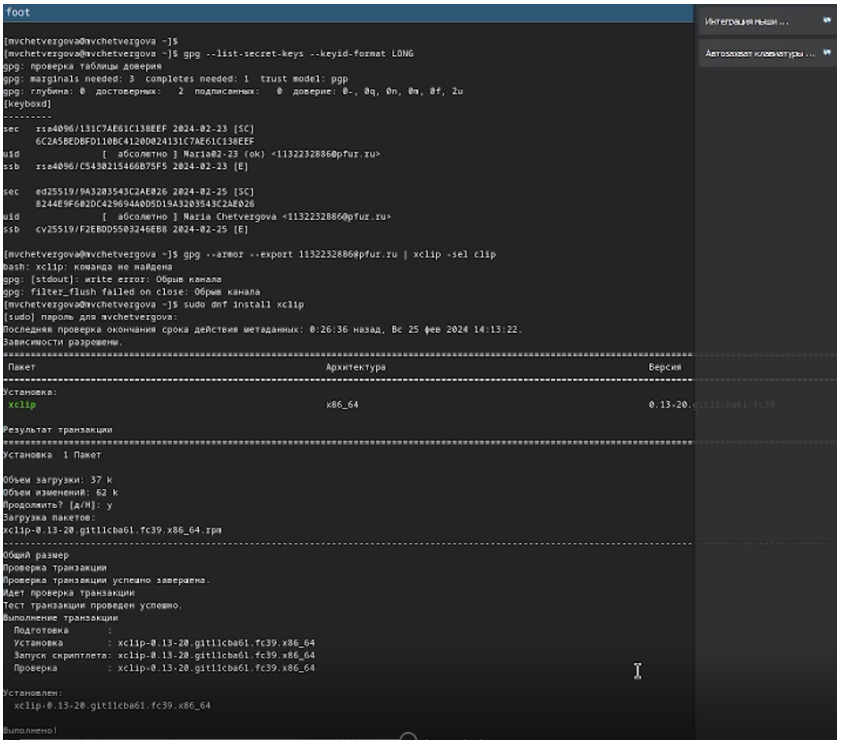


Рис. 2.6: Вывод и копирование ключа в буфер обмена

В случае, если предыдущая команда не сработала, можно вывести содержимое ключа командой

*gpg –armor –export | cat*

и скопировать вручную :-)

- переходим в настройки GitHub и вставляем ключ в нужное поле

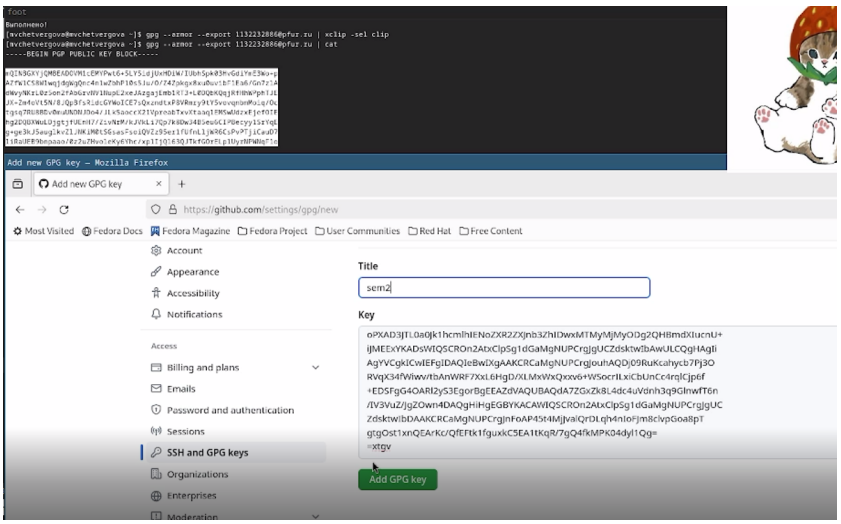


Рис. 2.7: Добавление ключа в GitHub

## 2.6 Настройка автоматических подписей коммитов

Используя введёный емейл, укажем Git применять его при подписи коммитов. Это можно сделать при помощи следующих команд:

*git config –global user.sighingkey  git config –global commit.gpgsign true git config –global gpg.program $(which gpg2)*


Рис. 2.8: Настройка автоматических подписей коммитов

## 2.7 Настройка gh

- Для начала необходимо авторизоваться, ответив на несколько вопросов после вбивания этой программы:

*gh auth login*

Авторизация происходит через браузер. Важно выбрать SSH в одном из вопросов :-)

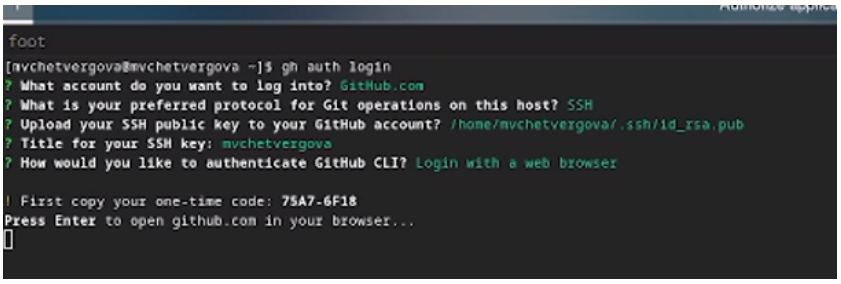


Рис. 2.9: Настройка gh

- Улита задаст несколько наводящих вопросов, после чего можно авторизоваться через браузер

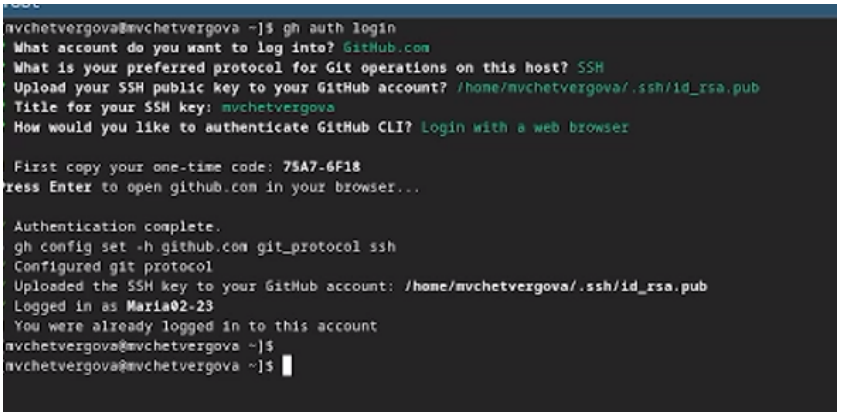


Рис. 2.10: Настройка gh

## 2.8 Шаблон для рабочего пространства

1. Создание репозитория курса на основе шаблона Создадим шаблон рабочего пространства для 2023-2024 годов обучения. Для этого введём в терминал следующие команды:

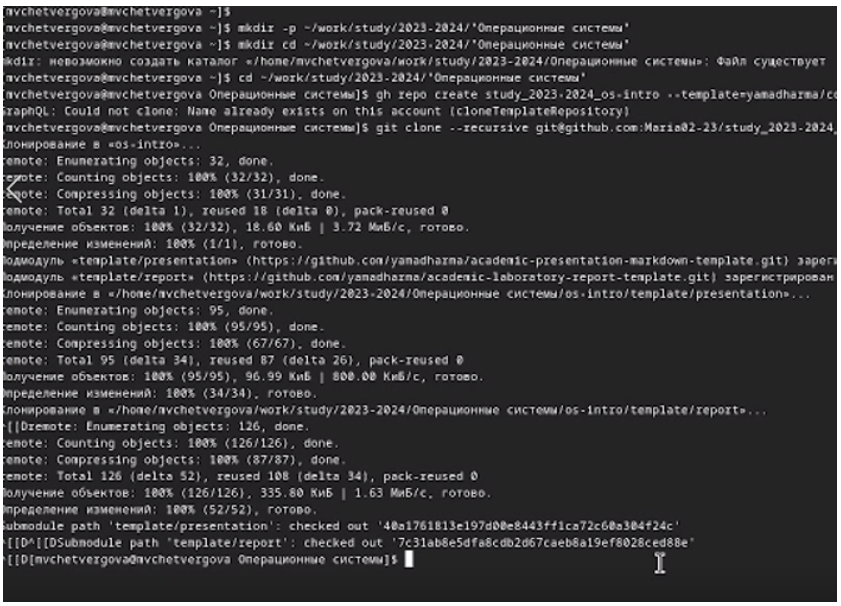


Рис. 2.11: Создание шаблона рабочего пространства. Репозиторий на основе шаблона

2. Настройка каталога курса

Перейдём в каталог курса и удалим лишние файлы с помощью команды *rm package.json*. Затем создадим необходимые файлы, которые помогут с работой. В конце необходимо отправить файлы на сервер

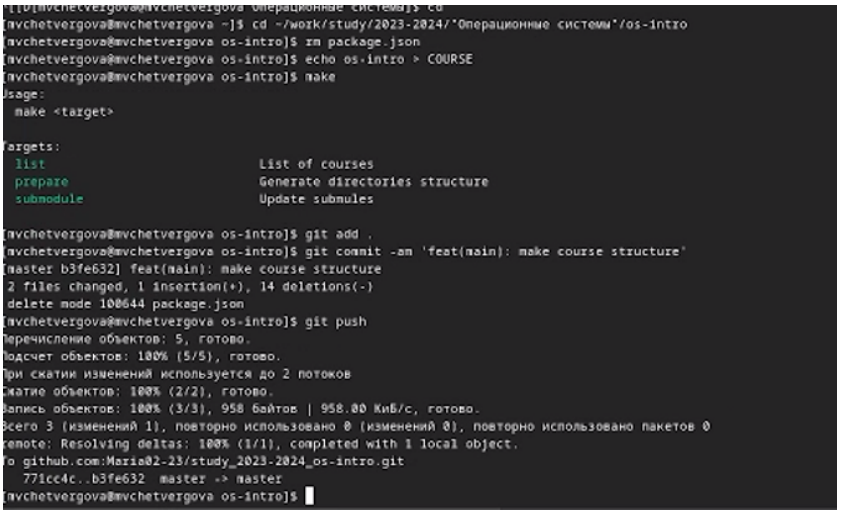


Рис. 2.12: Настройка gh

## 2.9 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. Можно прописать шаблоны игнорируемых при добавлении в репозиторий типов файлов в файл .gitignore с помощью сервисов.

# 3 Выводы

в ходе выполнения лабораторной работы мы изучили идеологию и применение средств контроля версий и освоили умения по работе с системой git.

# Список литературы

1. U. L.E. studing. 2018.