Лабораторная работа номер 2

Отчёт

Виноградова Мария Андреевна

Содержание

Список иллюстраций

Список таблиц

# Цель работы

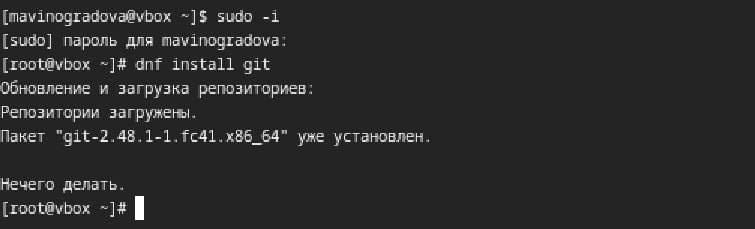
Изучить идеологию и применение средств контроля версий. Освоить умения по работе с git.

# Выполнение лабораторной работы

## Порядок выполнения лабораторной работы

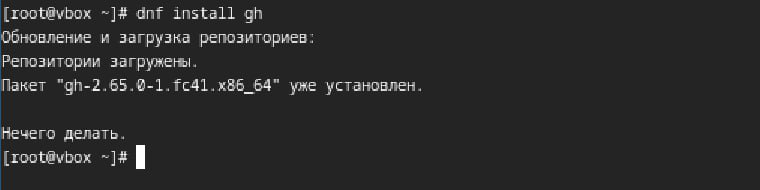
### Установка git и gh

Установка git (рис. [-@fig:001]).



устанавливаю git

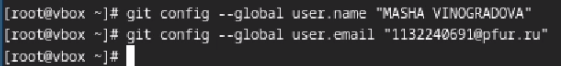
Установка gh (рис. [-@fig:002]).



устанавливаю gh

### Базовая настройка git

Зададим имя и email владельца репозитория (рис. [-@fig:003]).



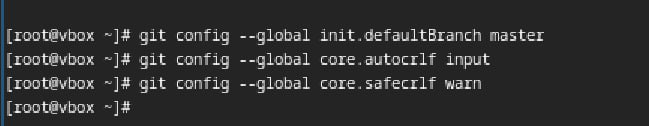
задаю имя и email владельца

Настроим utf-8 в выводе сообщений git: (рис. [-@fig:004]).

настраиваю utf-8 в выводе сообщений

настраиваю utf-8 в выводе сообщений

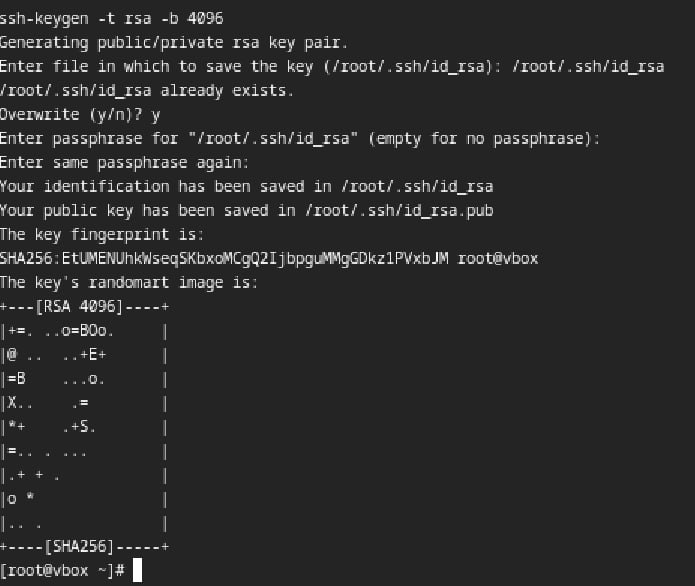
Настраиваю верификацию и подписание коммитов git. И зададаю имя начальной ветки (буду называть её master) (рис. [-@fig:005]).



настраиваю верификацию и задаю имя ветки

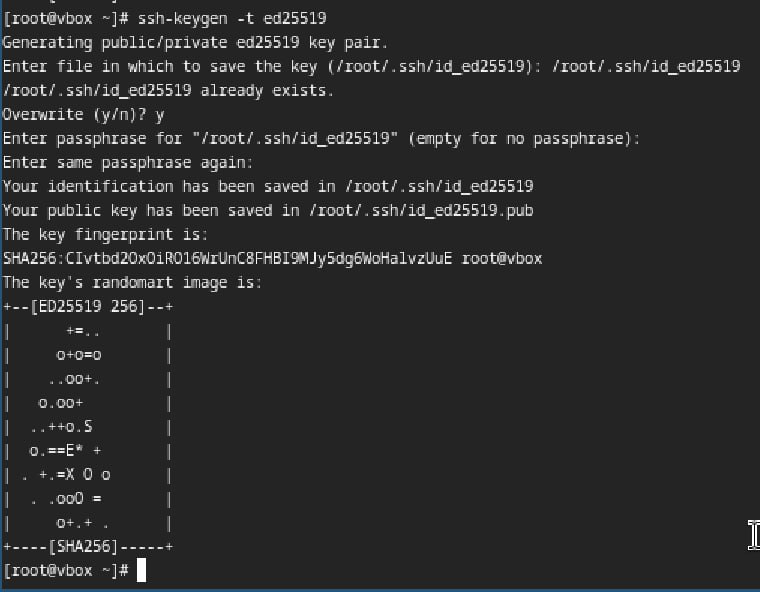
### Создание SSH ключей

По алгоритму rsa создаю ключ размером 4096 бит: (рис. [-@fig:006]).



создаю ключ

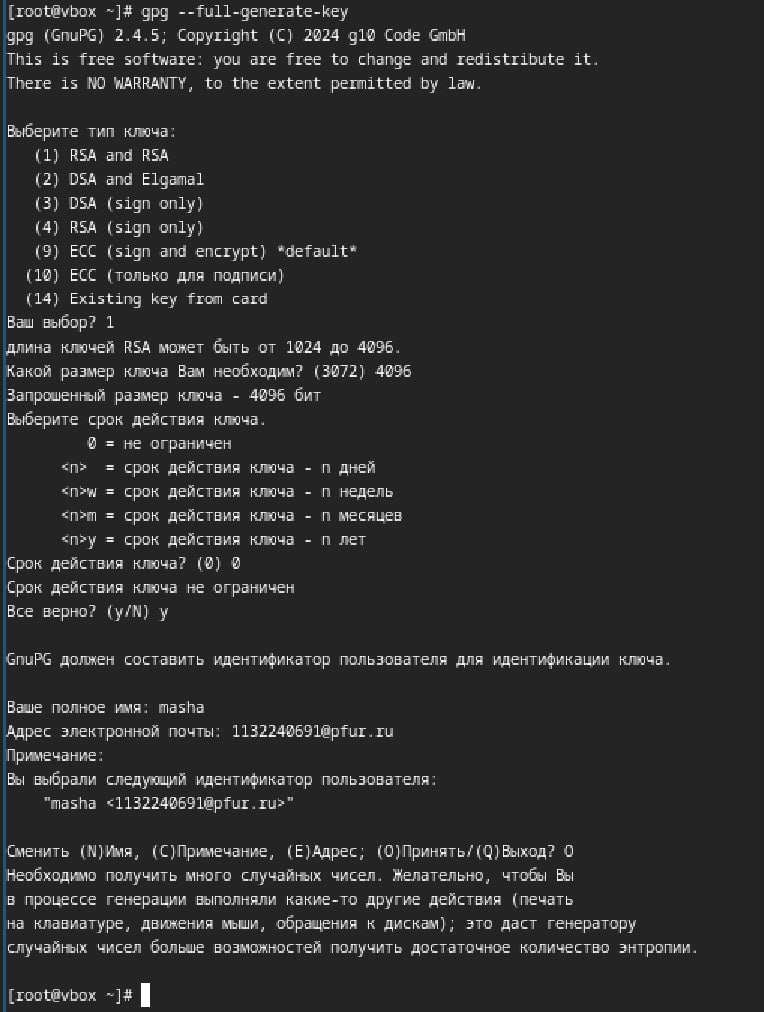
создаю ключ (рис. [-@fig:007]).



создаю ключ

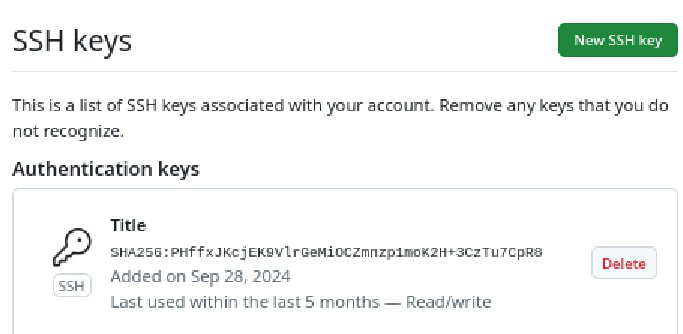
### Создание ключей pgp

Генерирую ключ (рис. [-@fig:008]).



генерация ключа

Добавляю SSH ключ на github (рис. [-@fig:009]).



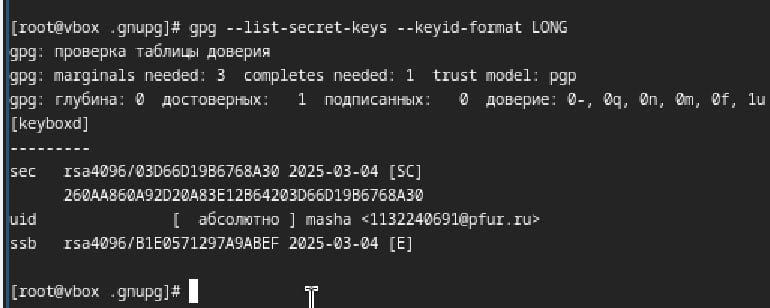
добавляю ключ на github

### Настройка github

Учетная запись была создана мною ранее для выполнения предыдущих лабораторных работ

### Добавление PGP ключа в GitHub

Вывожу список ключей и копирую отпечаток приватного ключа (рис. [-@fig:010]).



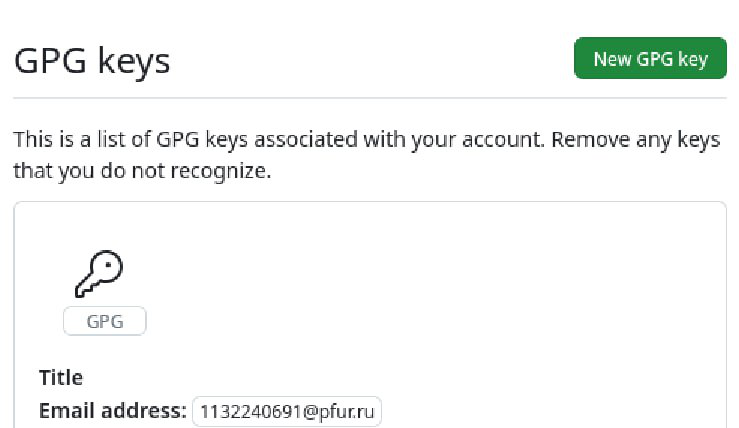
вывожу список ключей

Копирую свой сгенерированный PGP ключ в буфер обмена (рис. [-@fig:011]).



копирую ключ

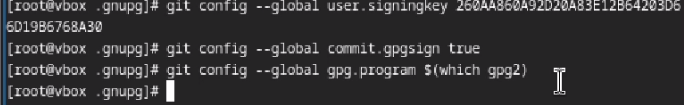
Перехожу в настройки GitHub, нажимаю на кнопку New GPG key и вставляю полученный ключ в поле ввода. (рис. [-@fig:012]).



добавляю ключ на github

### Настройка автоматических подписей коммитов git

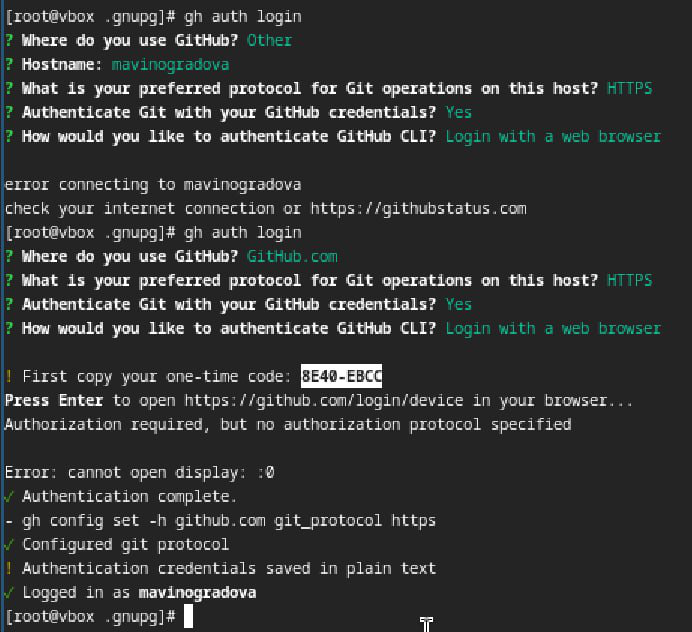
Используя введёный email, указываю Git применять его при подписи коммитов (рис. [-@fig:013]).



настраиваю подписи коммитов

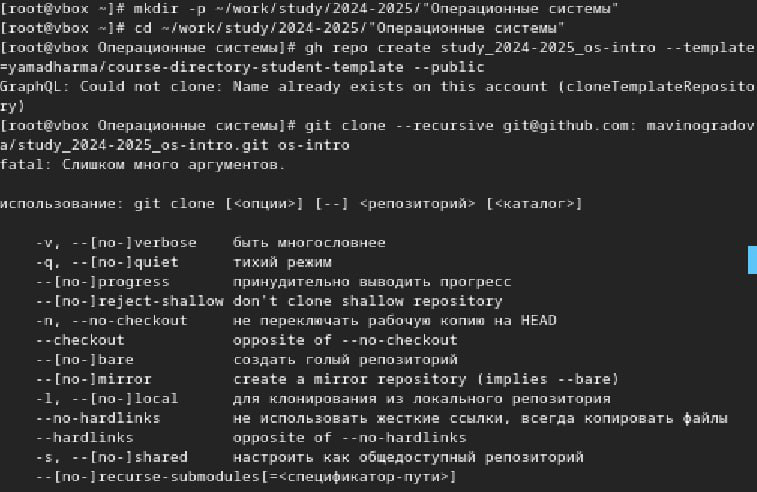
### Настройка gh

Для начала авторизуюсь (рис. [-@fig:014]).

 После этого авторизуюсь через браузер (ввожу код в предложенное окно)

### Сознание репозитория курса на основе шаблона

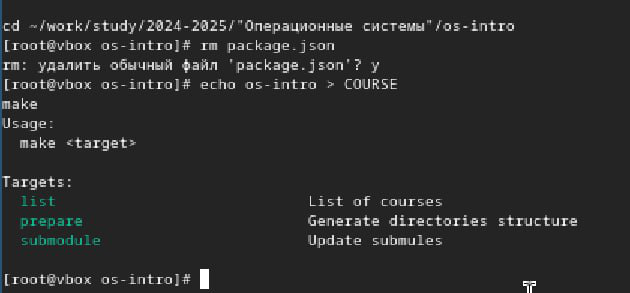
Создаю шаблон рабочего пространства (рис. [-@fig:015]).



создаю рабочее пространство

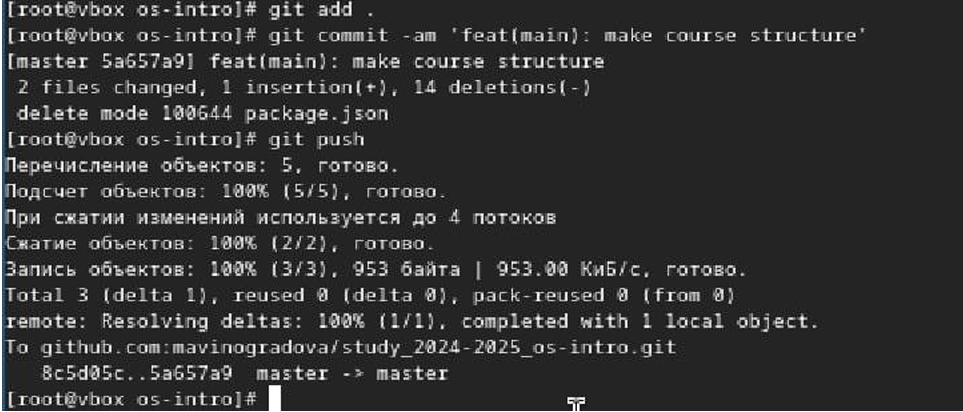
### Настройка каталога курса

Перехожу в каталог курса, удаляю лишние файлы и создаю необходимые каталоги (рис. [-@fig:016]).



удаляю файлы и создаю каталоги внутри каталога курса

Отправляю файлы на сервер (рис. [-@fig:017]).



отправляю файлы на сервер

## Задание для самостоятельной работы

### Контрольные вопросы

1. Системы контроля версий (VCS) — это программные инструменты, предназначенные для управления изменениями в исходном коде или других файлах. Они позволяют: Сохранять историю изменений. Отслеживать, кто и когда внёс изменения. Возвращаться к предыдущим версиям файлов. Совместно работать над проектами без конфликтов. Создавать ветки для параллельной разработки.
2. Хранилище (репозиторий): Место, где хранится вся история изменений проекта, включая файлы, коммиты и метаданные. Commit (фиксация): Отдельная запись в истории изменений, которая содержит изменения файлов, комментарий и автора. История: Последовательность коммитов, которая показывает, как проект развивался с течением времени. Рабочая копия: Текущая версия файлов проекта, с которой работает пользователь. Она может быть изменена, а затем зафиксирована в репозитории. Отношения: Рабочая копия берётся из хранилища. Изменения в рабочей копии фиксируются в виде коммитов. Коммиты формируют историю проекта.
3. Централизованные VCS: Все изменения хранятся на центральном сервере. Пользователи работают с локальными копиями, но для фиксации изменений требуется подключение к серверу. Пример: SVN (Subversion). Децентрализованные VCS: Каждый пользователь имеет полную копию репозитория, включая всю историю изменений. Работа возможна без постоянного подключения к серверу. Пример: Git, Mercurial.
4. Инициализация репозитория; Добавление файлов в отслеживание; Фиксация изменений; Просмотр истории изменений; Возврат к предыдущей версии
5. Опишите порядок работы с общим хранилищем VCS. Клонирование удалённого репозитория; Создание новой ветки для работы; Внесение изменений и их фиксация; Отправка изменений на удалённый репозиторий; Получение изменений от других разработчиков; Слияние веток (например, после code review)
6. Управление версиями файлов. Совместная работа над проектами. Создание и управление ветками. Отслеживание изменений и их авторов. Резервное копирование и восстановление проекта.
7. git init: Инициализация нового репозитория. git clone : Клонирование удалённого репозитория. git add : Добавление файлов в индекс для последующего коммита. git commit -m “сообщение”: Фиксация изменений. git status: Просмотр состояния рабочей копии. git log: Просмотр истории коммитов. git push: Отправка изменений на удалённый репозиторий. git pull: Получение изменений из удалённого репозитория. git branch: Управление ветками. git merge: Слияние веток.
8. Локальный репозиторий:  
   Создание репозитория: git init Добавление файлов: git add . Удалённый репозиторий: Клонирование: git clone Отправка изменений: git push origin main Получение изменений: git pull origin main
9. Ветви (branches) — это отдельные линии разработки в репозитории. Они нужны для: Параллельной работы над разными функциями или исправлениями. Изоляции экспериментальных изменений. Упрощения процесса code review и слияния изменений.
10. Для игнорирования файлов используется файл .gitignore. В него добавляются шаблоны файлов или каталогов, которые не должны отслеживаться Git. Это полезно для: Исключения временных файлов (например, кэша или логов). Игнорирования конфиденциальных данных (например, паролей или ключей).

# Вывод

В ходе выполнения лабораторной работы были изучены основные принципы идеологии и применения систем контроля версий (VCS), в частности, инструмента Git. Освоены ключевые навыки работы с Git, включая создание и клонирование репозиториев, фиксацию изменений, работу с ветками, а также взаимодействие с удалёнными репозиториями. Были выполнены практические задания по управлению версиями файлов, что позволило понять важность и удобство использования VCS для совместной разработки и контроля изменений в проектах.