Отчет по лабораторной работе №2

Операционные системы

Серебрякова Дарья Ильинична

Содержание

# 1 Цель работы

Цель данной работы - изучение идеологии и применения средств контроля версий, освоение умения работы с git

# 2 Задание

1. Создать базовую конфигурацию для работы с git
2. создать ключи SSH и GPG
3. Создать локальный каталог для выполнения заданий по предмету

# 3 Выполнение лабораторной работы

## 3.1 Настройка git

Устанавливаю необходиммое программное обеспечение и перехожу к настройке git.

Задаю в качестве имени и адреса почты владельца репозитория свои данные (рис. 1).

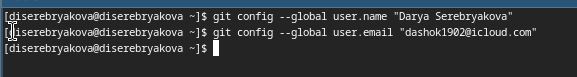


Рис. 1: 1

Настраиваю utf-8 в выводе сообщений гит для их корректного отображения (рис. 2).

Рис. 2: 2

Рис. 2: 2

## 3.2 Создание ключа SSH

Создаю ключ ssh размером 4096 бит по алгоритму rsa (рис. 3).

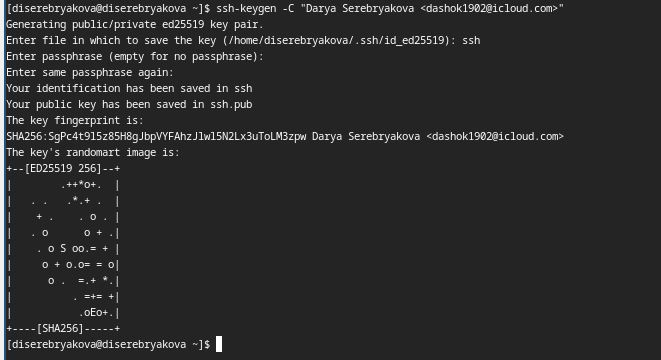


Рис. 3: 5

Ключ создан (рис. 4).

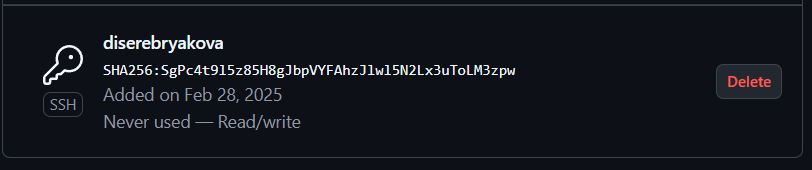


Рис. 4: 6

## 3.3 Создание ключа GPG

Генерирую ключ gpg, затем выбираю тип ключа, задаю максимальную его длину и ставлю неограниченный срок действия ключа (рис. 5).

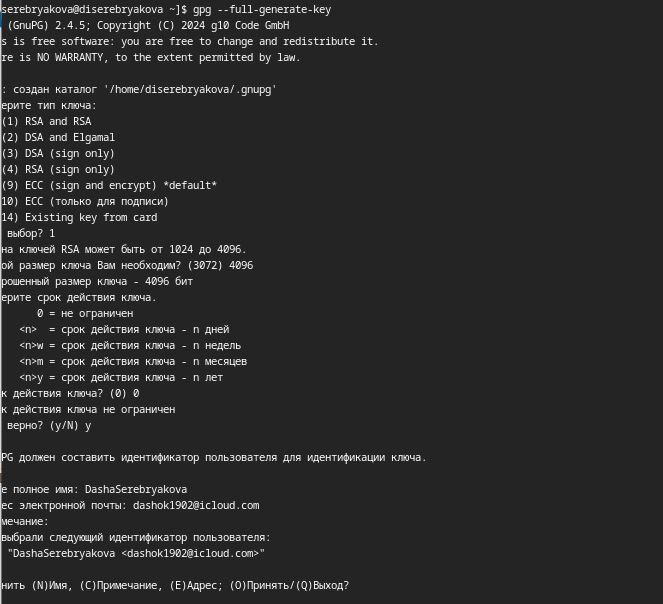


Рис. 5: 7

## 3.4 Добавление ключа GPG в гитхаб

Выыожу список созданных ключей в терминал, копирую в буфер обмена отпечаток ключа (рис. 6).

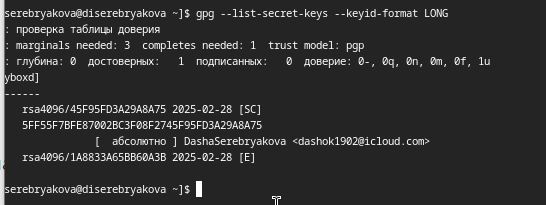


Рис. 6: 8

Ввожу в терминале команду, с помощью которой копирую сам ключ в буфер обмена (рис. 7).

Рис. 7: 9

Рис. 7: 9

Ключ создан (рис. 8).

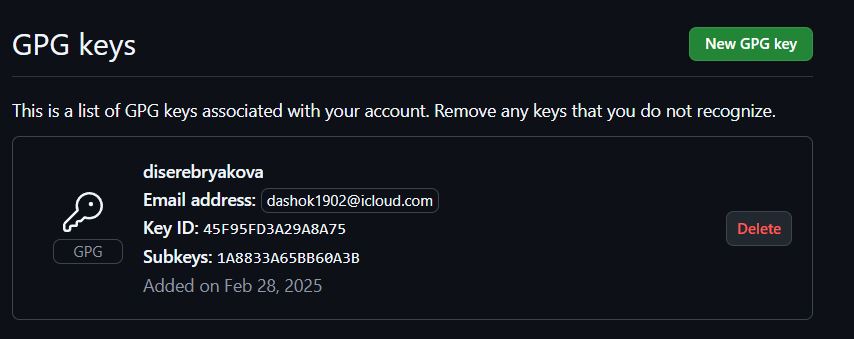


Рис. 8: 10

## 3.5 Настройка подписи git

Настраиваю автоматические подписи коммитов гит (рис. 9).

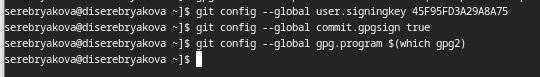


Рис. 9: 11

## 3.6 Настройка gh

Начинаю авторизацию в gh, авторизируюсь через браузер (рис. 10).

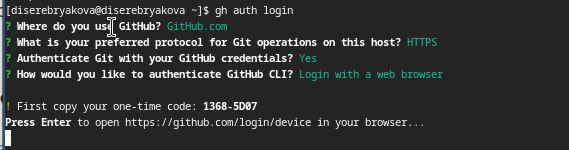


Рис. 10: 12

Авторизация успешно выполнена (рис. 11).

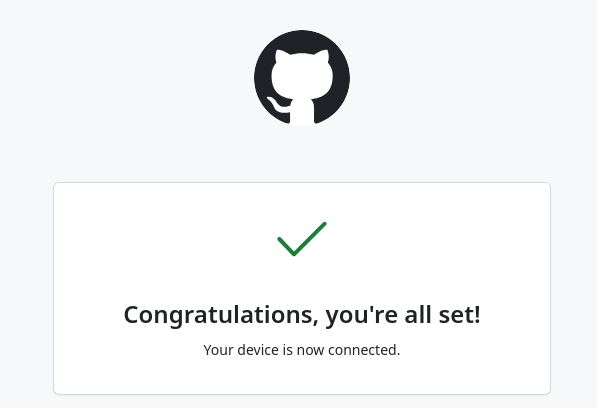


Рис. 11: 13

Вижу сообщение об окончании авторизации под именем dserebryakova (рис. 12).

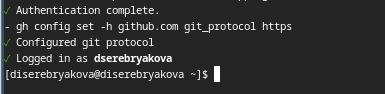


Рис. 12: 14

## 3.7 Создание репозитория курса на основе шаблона

Создаю директорию, перехожу в нее, создаю репозиторий на основе шаблона репозитория и клонирую его к себе в директорию (рис. 13).

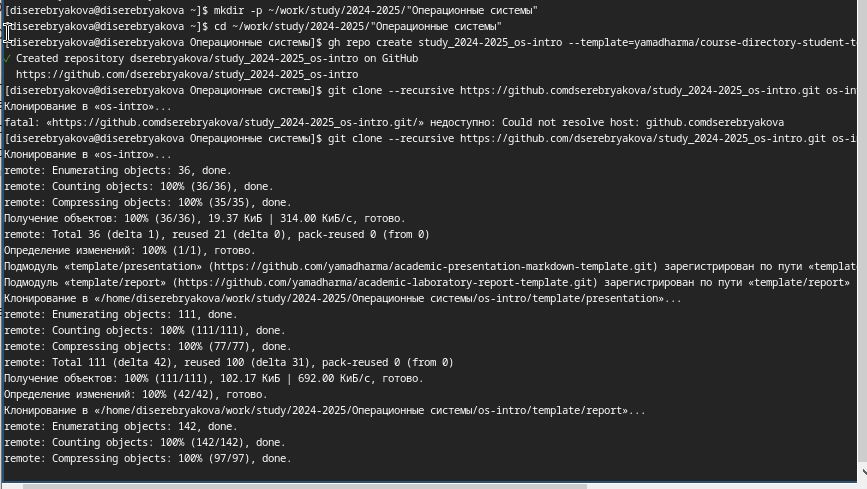


Рис. 13: 15

Перехожу в каталог курса, удаляю лишние файлы и создаю необходимые каталоги (рис. 14).

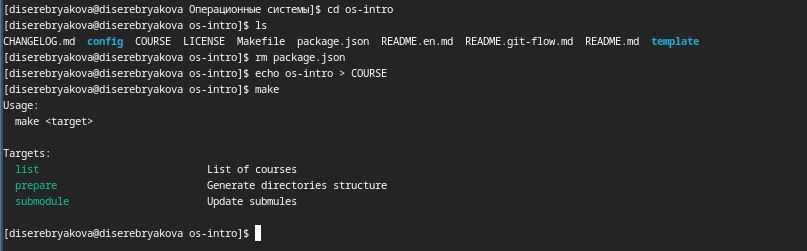


Рис. 14: 16

Добавляю все новые файлы и отправляю их на сервер (рис. 15).

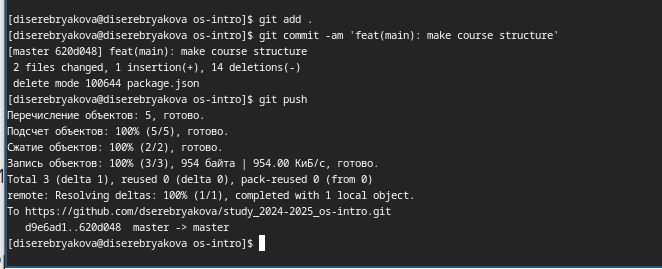


Рис. 15: 17

# 4 Выводы

При выполнении данной работы я изучила идеологию и применение средств контроля версий, освоила работу с git

# 5 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

1. git push -all отправляем из локального репозитория все сохраненные изменения в центральный репозиторий, предварительно создав локальный репозиторий и сделав предварительную конфигурацию.
2. 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.
3. 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.

# 6 Список литературы

1. Лабораторная работа №2