Отчет по лабораторной работе № 2

Операционные системы

Чувакина Мария Владимировна

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Цель данной лабораторной работы – изучение идеологии и применение средств контроля, 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 Базовая настройка git.

Задаю в качестве имени и email владельца репозитория свои имя, фамилию и электронную почту (рис.1).

```
mvchuvakina@dk2n24 ~ $ git config --global user.name "Maria Chuvakina"
mvchuvakina@dk2n24 ~ $ git config --global user.email "maria.chuvakina@mail.ru"
```

Рис. 3.1: Задаю имя и email владельца репозитория

Настраиваю utf-8 в выводе сообщений git для их корректного изображения (рис.2).

```
mvchuvakina@dk2n24 ~ $ git config --global core.quotepath false
```

Рис. 3.2: Настройка utf-8 в выводе сообщений git

Начальной ветке задаю имя master (рис.3).

```
mvchuvakina@dk2n24 ~ $ git config --global init.defaultBranch master
```

Рис. 3.3: Задаю имя начальной ветки

Задаю параметры autocrlf и safecrlf для корректного отображения конца строки (рис.4).

```
mvchuvakina@dk2n24 ~ $ git config --global core.autocrlf input
mvchuvakina@dk2n24 ~ $ git config --global core.safecrlf warn
```

Рис. 3.4: Задаю параметры autocrlf и safecrlf

3.2 Создание ключа SSH.

Создаю ключ ssh размером 4096 бит по алгоритму rsa (рис.5).

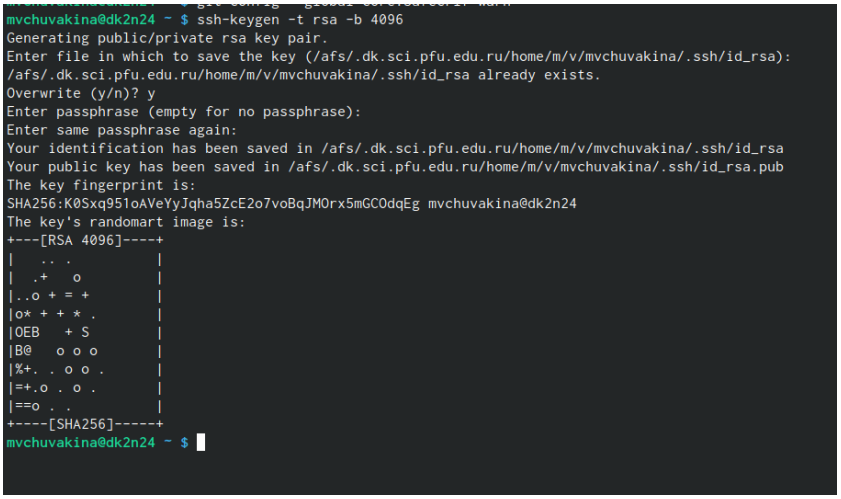

Рис. 3.5: Генерация ssh ключа по алгоритму rsa

Создаю ключ ssh по алгоритму ed25519 (рис.6).

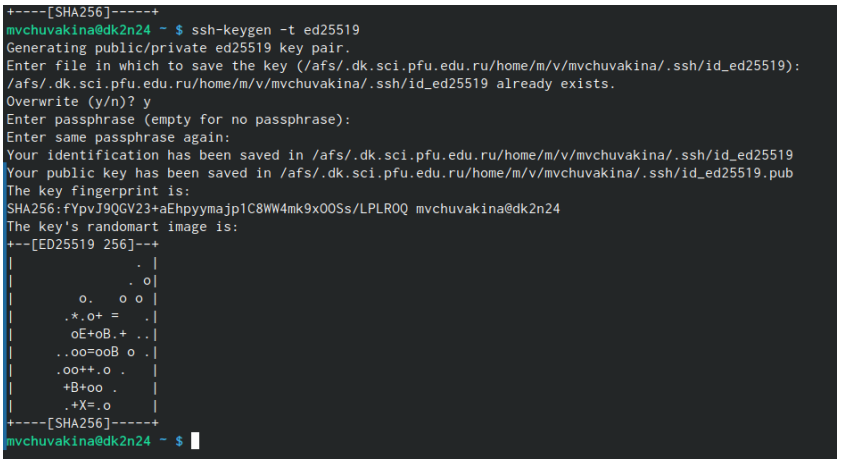

Рис. 3.6: Генерация ssh ключа по алгоритму ed25519

3.3 Создание ключа GPG.

Генерирую ключ GPG, затем выбираю тип ключа RSA and RSA, задаю максимальную длину ключа: 4096, оставляю неограниченный срок действия ключа.

Далее отвечаю на вопросы программы о личной информации. (рис.7).

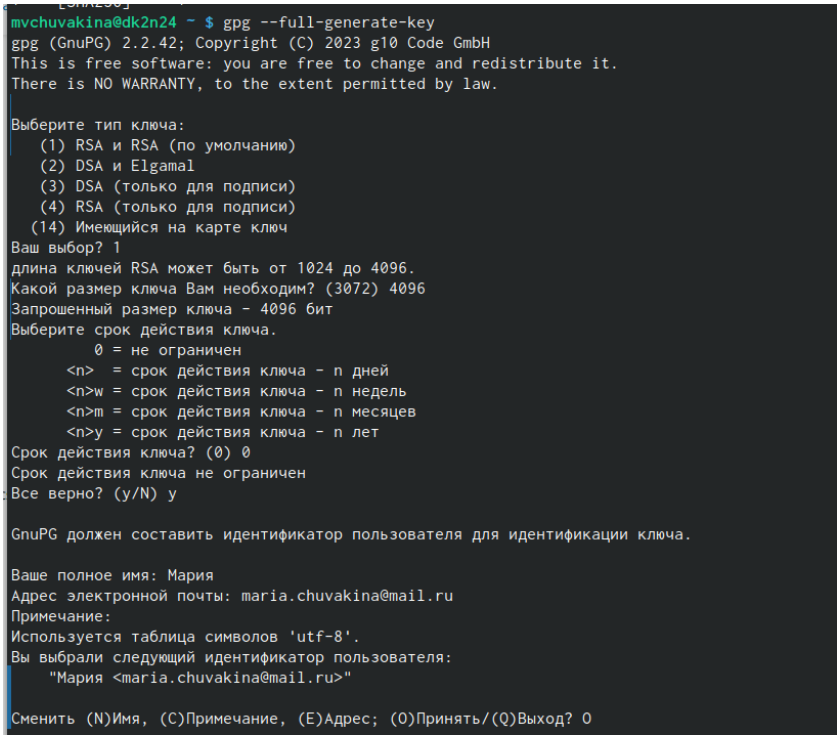

Рис. 3.7: Генерация ключа

Ввожу фразу-пароль для защиты нового ключа (рис.8).

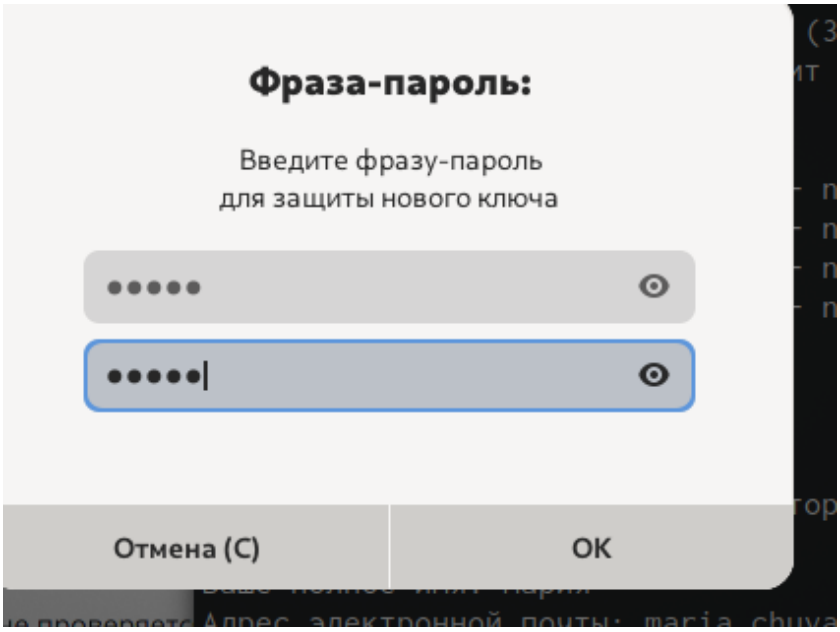

Рис. 3.8: Защита ключа GPG

3.4 Регистрация на Github.

У меня уже был создан аккаунт на Github, соответсвенно, основные данные аккаунта я так же заполняла и проводила его настройку, поэтому просто вхожу в свой аккаунт.

3.5 Добавление ключа GPG в Github.

Вывожу список созданных ключей в терминал, ищу в результате запроса отпечаток ключа (последовательность байтов для идентификации более длинного, по сравнению с самим отпечатком,ключа), он стоит после знака слеша, копирую его в буфер обмена. (рис.9).

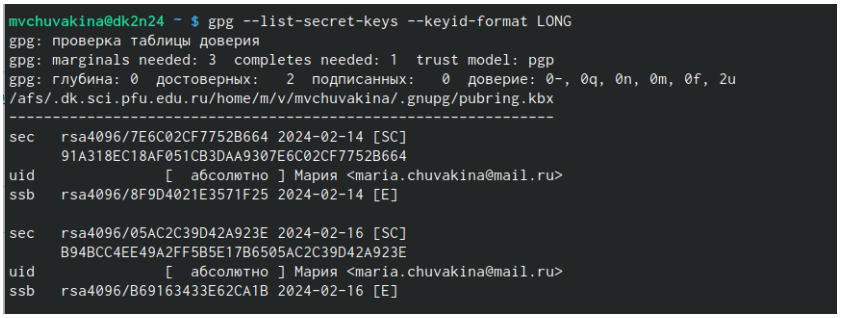

Рис. 3.9: Вывод списка ключей

Ввожу в терминале команду, с помощью которой копирую сам ключ GPG в буфер обмена, за это отвечает утилита xclip. (рис.10).

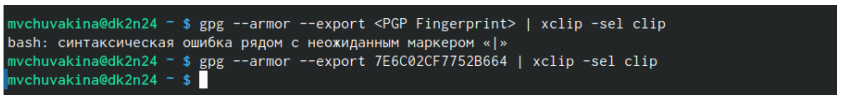

Рис. 3.10: Копирование ключа в буфер обмена

Открываю настройки Github, ищу среди них добавление GPG ключа (рис.11).

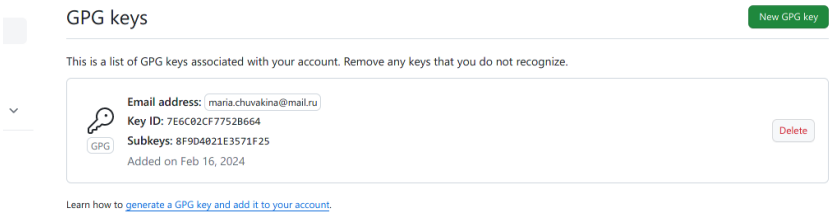

Рис. 3.11: Настройки Github

Нажимаю на “New GPG key” и вставляю в поле ключ из буфера обмена (рис.12).

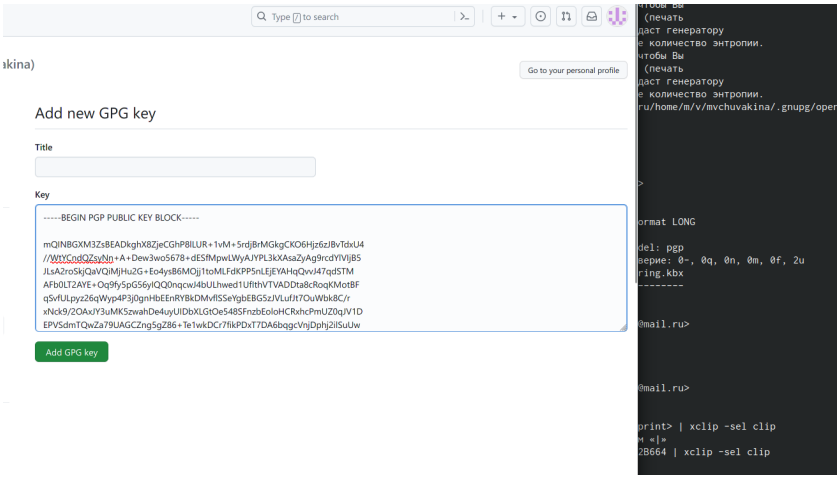

Рис. 3.12: Добавление нового PGP ключа

Я добавила ключ GPG на Github.

3.6 Настройка подписи Git.

Настраиваю автоматические подписи коммитов git: используя введенный ранее email, указываю git, использую его при создании подписей коммитов.(рис.13).

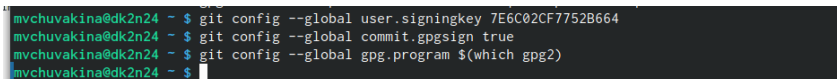

Рис. 3.13: Настройка подписей Git

3.7 Настройка gh.

Начинаю авторизацию в gh, отвечаю на наводящие вопросы от утилиты, в конце выбираю авторизоваться через браузер. Далее вижу сообщение о завершении авторизации под именем mvchuvakina. (рис.14).

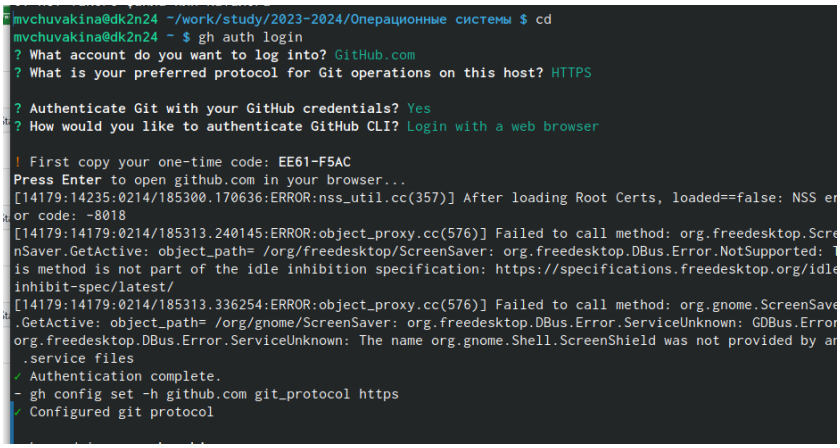

Рис. 3.14: Авторизация в gh

3.8 Создание репозитория курса на основе шаблона.

Сначала создаю директорию с помощью утилиты mkdir и флага -p, который позволяет установить каталоги на всем указанном пути. После этого с помощью утилиты cd перехожу в созданную директорию. После этого клонирую репозиторий к себе в директорию (рис.15).

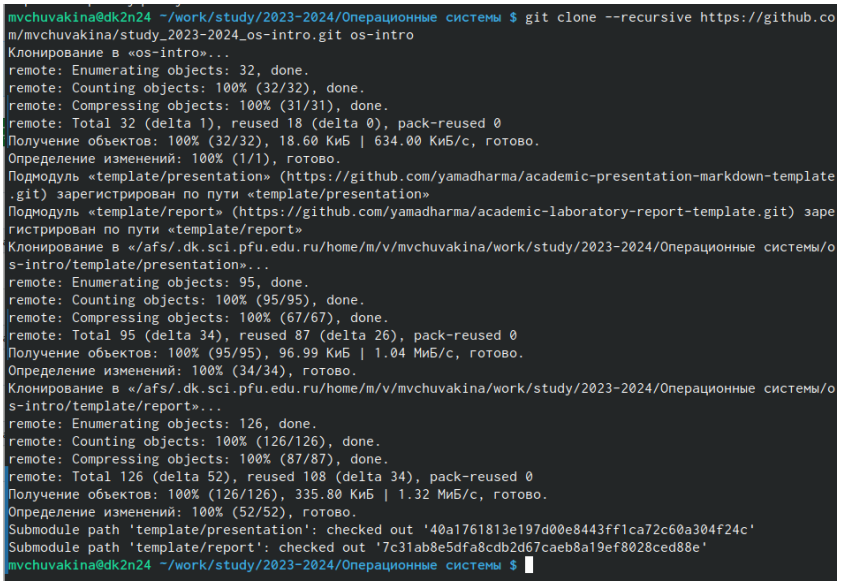

Рис. 3.15: Создание репозитория

Перехожу в каталог курса с помощью утилиты cd, проверяю содержание каталога с помощью утилиты ls (рис.16).

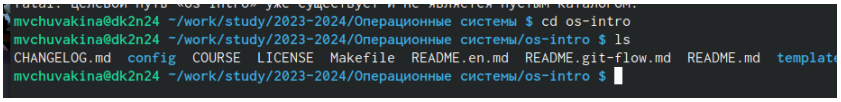

Рис. 3.16: Перемещение между директориями

Удаляю лишние файлы с помощью утилиты rm, далее создаю необходимые каталоги используя makefile (рис.17).

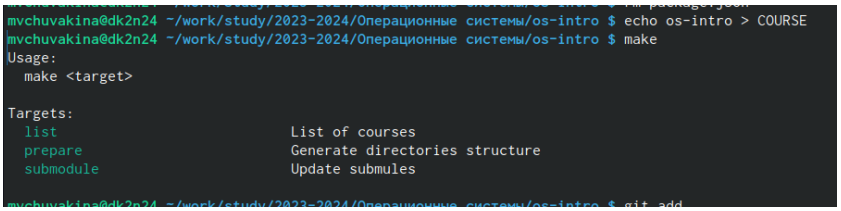

Рис. 3.17: Удаление файлов и создание каталогов

Добавляю все новые файлы для отправки на сервер с помощью команды git add и комментирую из с помощью git commit. Отправляю файлы на сервер с помощью git push (рис.18).

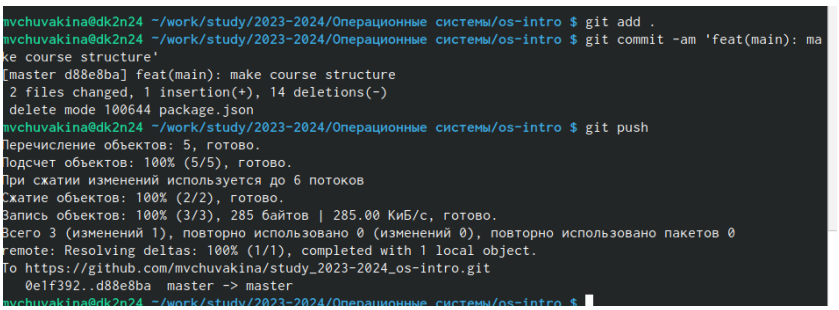

Рис. 3.18: Отправка файлов на сервер

4 Выводы

При выполнении данной лабораторной работы я изучила идеологию и применение средств контроля версий, освоила умение по работе с git.

5 Ответы на контрольные вопросы.

1. Системы контроля версий (VCS) - программное обеспечение работы с изменящейся информацией. Они позволяют хранить несколько версий изменяющейся информации, одного и того же документа, может предоставлять 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 применяются для: хранения полной истории изменений, сохранения причин всех изменений и 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 (децентрали-

зованных) 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. Используется для разработки новых функций.

10. Во время работы над проектом могут создаваться файлы, которые не следует добавлять в репозиторий. Например, временные файлы. Можно прописать шаблоны игнорируемых при добавлении в репозиторий типов файлов в файл .gitignore с помощью сервисов.