Отчёт по лабораторной работе №2

Операционные системы

Андреева Софья Владимировна

Содержание

Список иллюстраций

1 Цель работы

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

2 Выполнение работы

Установим git (рис. fig. 2.1).

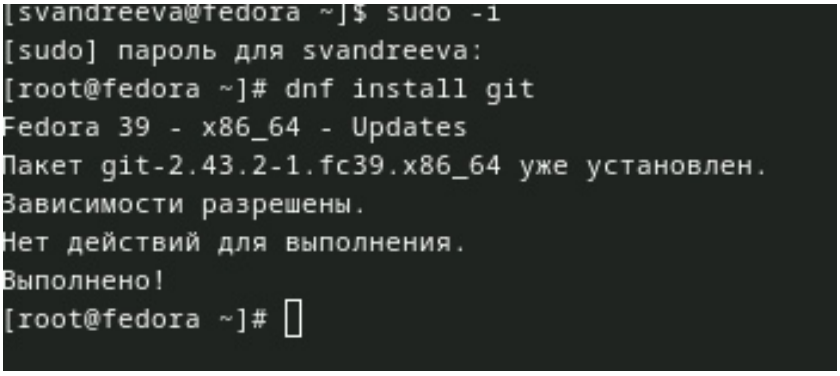

Рис. 2.1: Установим git:

Установим gh (рис. fig. 2.2).

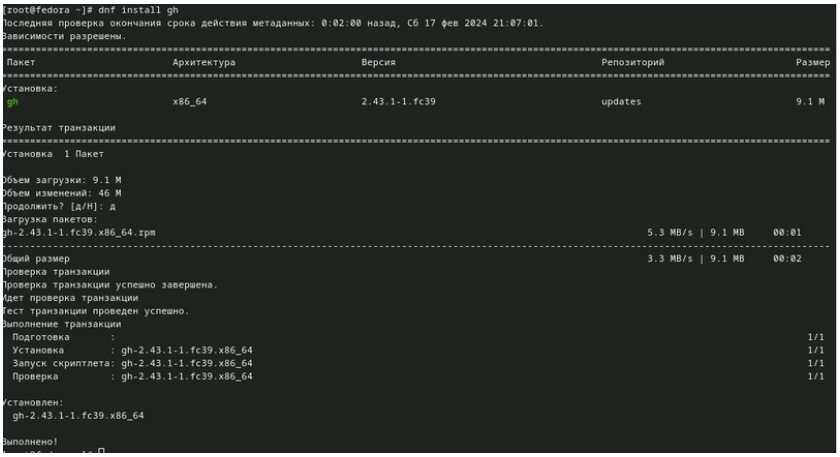

Рис. 2.2: Установим gh

Проведем базовую настройку git.Зададим имя и email владельца репозитория,

настроим utf-8 в выводе сообщений git, зададим имя начальной ветки, параметр autocrlf и safecrlf (рис. fig. 2.3).

```
[svandreeva@fedora ~]$ git config --global user.name "svandreeva"
[svandreeva@fedora ~]$ git config --global user.email "andreevasofa57@gmail"
[svandreeva@fedora ~]$ git config --global core.quotepath false
[svandreeva@fedora ~]$ git config --global init.defaultBranch master
[svandreeva@fedora ~]$ git config --global core.autocrlf input
[svandreeva@fedora ~]$ git config --global core.safecrlf warn
[svandreeva@fedora ~]$
```

Рис. 2.3: Проведем базовую настройку git.

Создадим ключи ssh: по алгоритму rsa с ключом размером 4096 бит (рис. fig. 2.4).

```
[svandreeva@fedora ~]$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/svandreeva/.ssh/id_rsa):
Created directory '/home/svandreeva/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/svandreeva/.ssh/id_rsa
Your public key has been saved in /home/svandreeva/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:KHLbOXJzI1Lz7TohWRKjspdozELAZWAxubjLoSpdO8g svandreeva@fedora
The key's randomart image is:
+---[RSA 4096]----+
|.==o             |
|ooo   o          |
|o .  . o         |
|.o. . ...        |
|.+.+o.++S        |
|o.*o+=o+..       |
|o*.++.B.+..      |
|+.E o+ =.o       |
|+    .  .o.      |
+----[SHA256]-----+
```

Рис. 2.4: ключ ssh по алгоритму rsa

И по алгоритму ed25519 (рис. fig. 2.5).

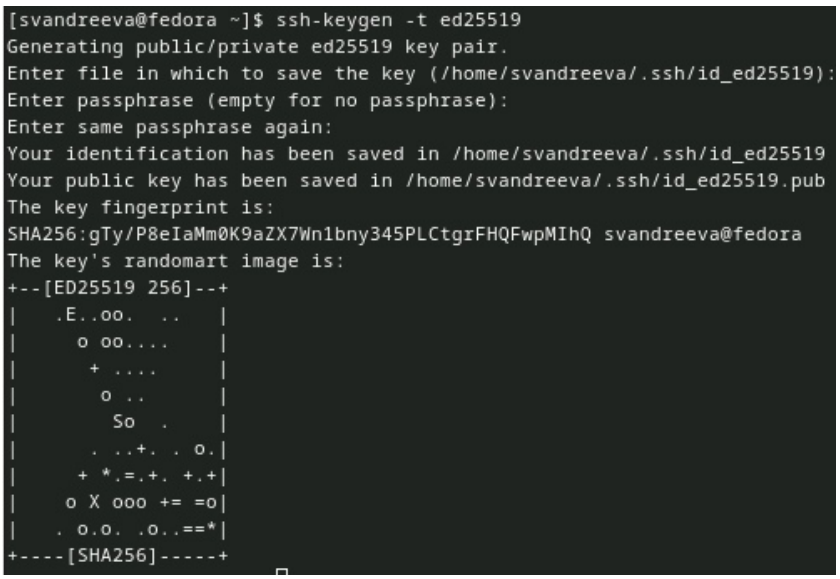

Рис. 2.5: ключ ssh по алгоритму ed25519

Создадим ключи pgp.Генерируем ключ, указав его тип, размер, срок действия (рис. fig. 2.6).

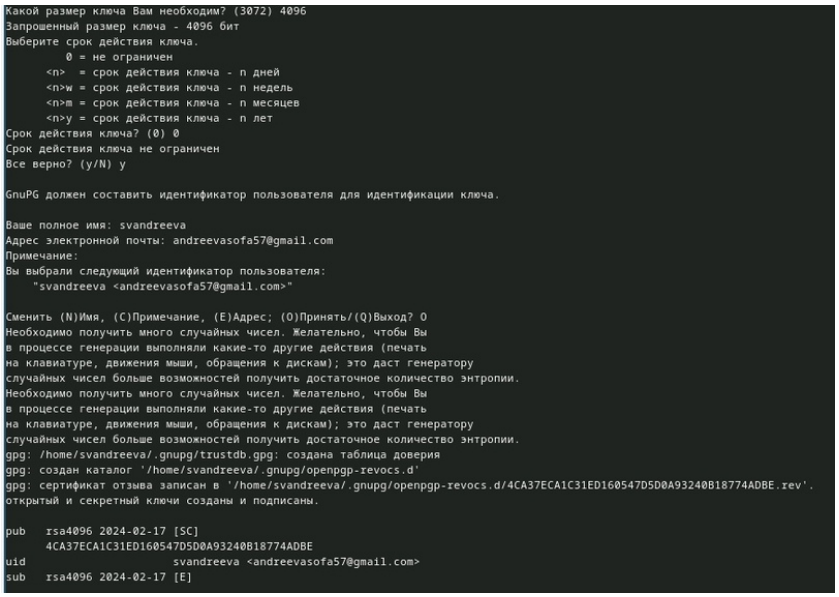

Рис. 2.6: Создадим ключи pgp

У меня уже есть учетная запись в github, поэтому следующим шагом мы добавляем PGP ключ в GitHub, для этого выводим список ключей и копируем отпечаток приватного ключа. (рис. fig. 2.7).

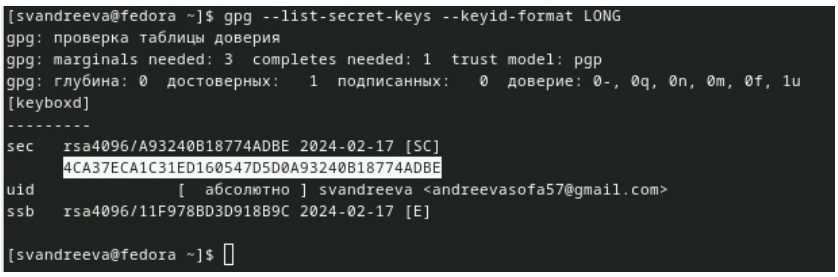

Рис. 2.7: список ключей

Затем копируем наш сгенерированный PGP ключ в буфер обмена и вставляем его при создании New GPG key в GitHub. (рис. fig. 2.8).

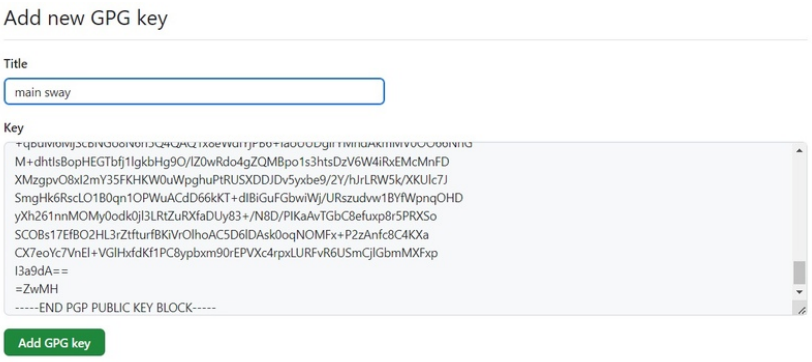

Рис. 2.8: New GPG key в GitHub

Используя введёный email, укажем Git применять его при подписи коммитов(рис.@fig:009).

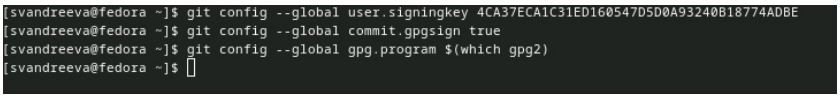

Рис. 2.9: email при подписи коммитов

Настройка gh.Для начала авторизуемся, ответив на несколько наводящих вопросов. (рис. fig. 2.10).

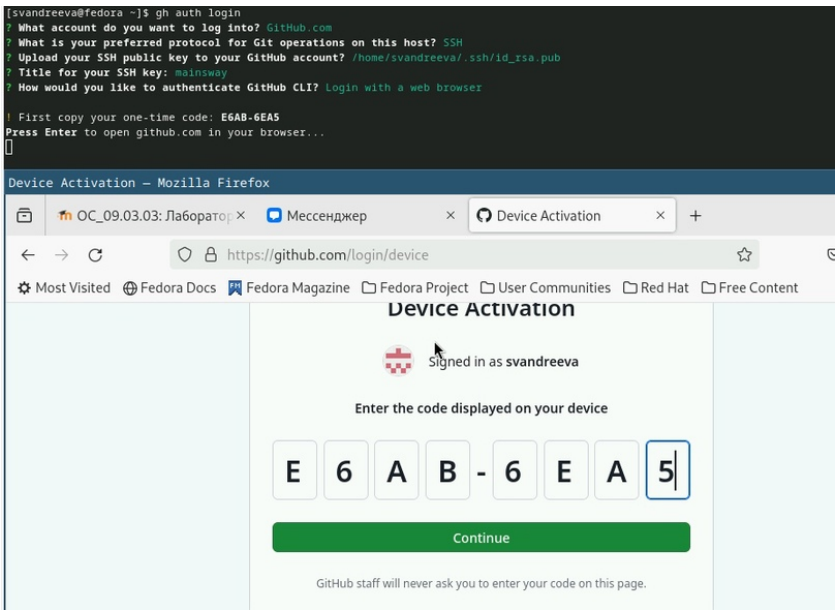

Рис. 2.10: Настройка gh

Создадим репозиторий, предварительно создав рабочее пространство (рис. fig. 2.11).

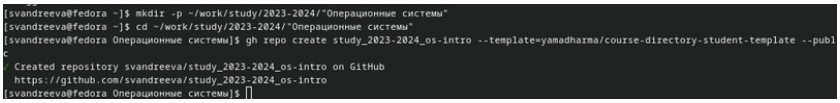

Рис. 2.11: Создадим репозиторий

Клонируем репозиторий (рис. fig. 2.12).

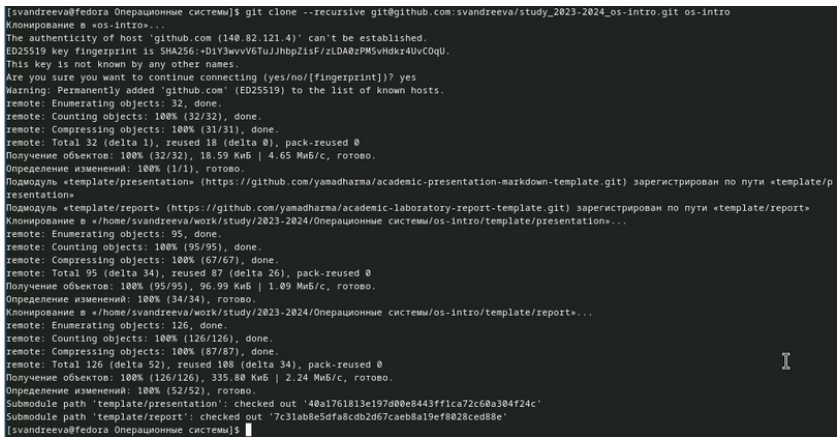

Рис. 2.12: Клонируем репозиторий

Перейдем в каталог курса, удалим лишние файлы и создадим необходимые каталоги(рис. fig. 2.13).

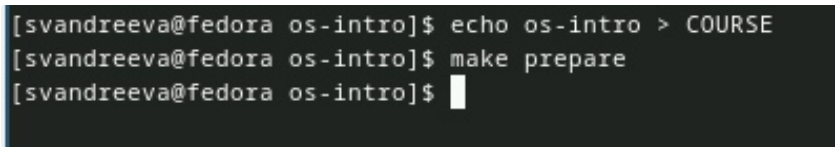

Рис. 2.13: создадим необходимые каталоги

Отправим файлы на сервер (рис. fig. 2.14).

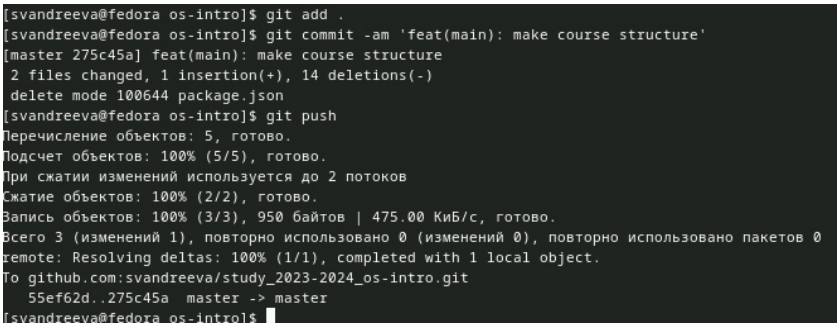

Рис. 2.14: Отправим файлы на сервер

3 Контрольные вопросы.

1. Системы контроля версий (VCS) разработаны специально для того, чтобы максимально упростить и упорядочить работу над проектом (вне зависимости от того, сколько человек в этом участвуют). СКВ дает возможность видеть, кто, когда и какие изменения вносил; позволяет формировать новые ветви проекта, объединять уже имеющиеся; настраивать контроль доступа к проекту; осуществлять откат до предыдущих версий.

2. Основные понятия:

- Хранилище (repository, сокр. repo), или репозитарий, — место хранения всех версий и служебной информации;
- Коммит (commit) — 1) синоним версии; 2) создание новой версии («сделать коммит», «закоммитить»);
- История разработки — совокупность всех версий файлов, над которыми ведется работа. Историей разработки в данном случае будет список изменений: создание файла, добавление изначального текста, исправление опечатки, добавление нового текста, объединение двух версий файла (при выполнении слияния);
- Рабочая копия (working copy или working tree) — текущее состояние файлов проекта, основанное на версии из хранилища (обычно на последней).

3. Централизованные и децентрализованные VCS:

- Централизованные VCS - одно основное хранилище всего проекта, где каждый пользователь копирует себе необходимые ему файлы из этого репози-

тория, изменяет и, затем, добавляет свои изменения обратно. Например Subversion, CVS, TFS, VAULT, AccuRev;

- Децентрализованные VCS - у каждого пользователя свой вариант (возможно не один) репозитория, присутствует возможность добавлять и забирать изменения из любого репозитория. Например Git, Mercurial, Bazaar.

4. Единоличная работа с хранилищем:

- работа в локальном репозитории;
- сохранение изменений и загрузка на серверов.

5. Работа с общим хранилищем VCS:

- проверка обновлений;
- загрузка обновлений (при наличии);
- работа в локальном репозитории;
- создаются ветвления, если несколько пользователей работают над одним и тем же файлом/документом;
- по результатам различных версий могут происходить слияния в одну ветвь.

6. Основные задачи, решаемые инструментальным средством git:

- хранить информацию о всех изменениях в коде;
- обеспечение удобства командной работы над кодом.

7. Примеры команд git:

- git pull - получение обновлений (изменений) текущего дерева из центрального репозитория;
- git push - отправка всех произведённых изменений локального дерева в центральный репозиторий;
- git status - просмотр списка изменённых файлов в текущей директории;

- git add - добавить все изменённые и/или созданные файлы и/или каталоги;
- git commit -am ‘Описание коммита’ - сохранить все добавленные изменения и все изменённые файлы.

8. Примеры команд для работы с локальным и удалённым репозиториями git push -all (push origin master/любой branch)

9. Ветка (англ. branch) — это последовательность коммитов, в которой ведётся параллельная разработка какого-либо функционала. Основная ветка – master. Ветки нужны, чтобы несколько программистов могли вести работу над одним и тем же проектом или даже файлом одновременно, при этом не мешая друг другу. Кроме того, ветки используются для тестирования экспериментальных функций: чтобы не повредить основному проекту, создается новая ветка специально для экспериментов.

10. Для игнорирования некоторых файлов можно создать файл .gitignore в корневом каталоге репозитория, чтобы сообщить Git, какие файлы и каталоги следует игнорировать при фиксации. Иногда имеется группа файлов, которые не нужно автоматически добавлять в репозиторий. К таким файлам обычно относятся автоматически генерируемые файлы (различные логи, результаты сборки программ и т. п.)

4 Выводы

Я изучила идеологию и применение средств контроля версий.Освоила умения по работе с git.