Отчёт по лабораторной работе №2

Операционные системы

Андреева Софья Владимировна

Содержание

# 1 Цель работы

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

# 2 Выполнение работы

Установим git (рис. fig. 1).

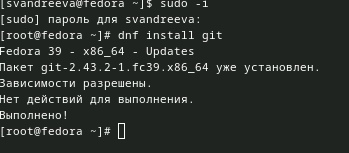


Рис. 1: Установим git:

Установим gh (рис. fig. 2).

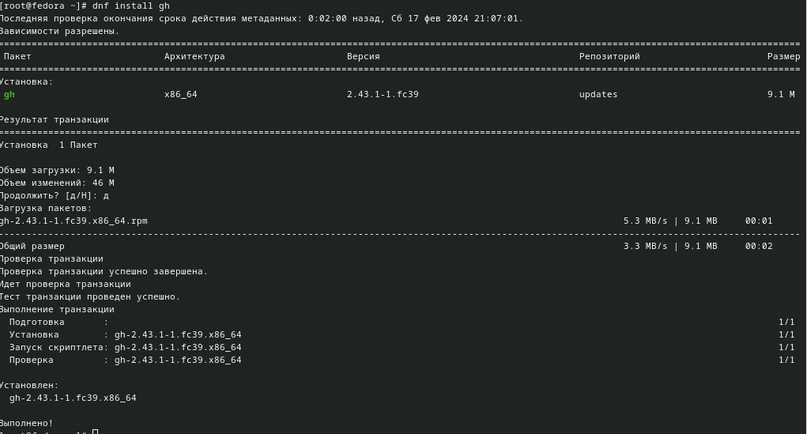


Рис. 2: Установим gh

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. 3).

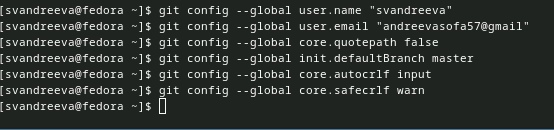


Рис. 3: Проведем базовую настройку git.

Создадим ключи ssh: по алгоритму rsa с ключом размером 4096 бит (рис. fig. 4).

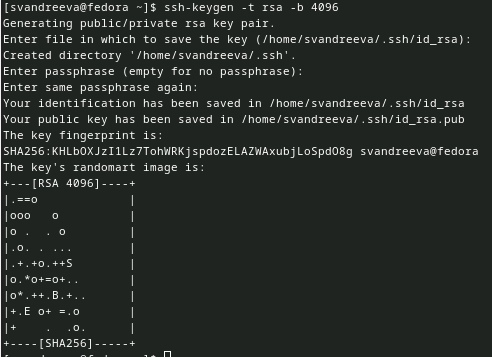


Рис. 4: ключ ssh по алгоритму rsa

И по алгоритму ed25519 (рис. fig. 5).

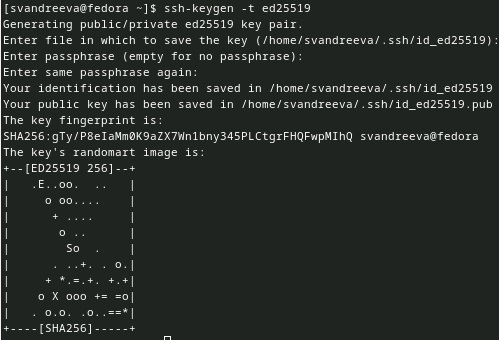


Рис. 5: ключ ssh по алгоритму ed25519

Создадим ключи pgp.Генерируем ключ, указав его тип, размер, срок действия (рис. fig. 6).

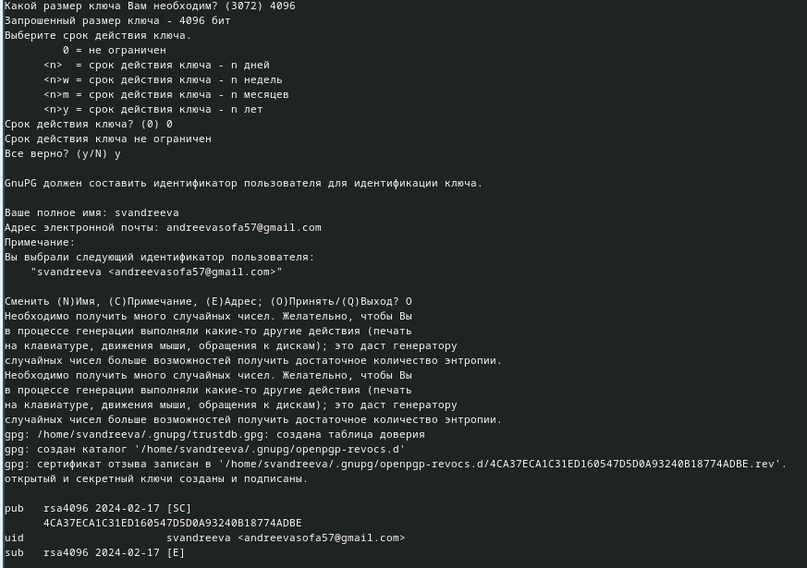


Рис. 6: Создадим ключи pgp

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. 7).

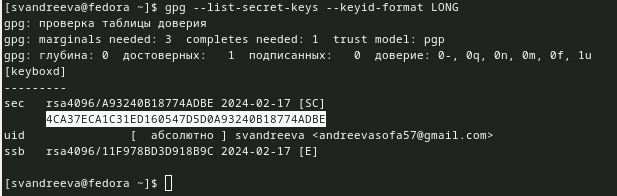


Рис. 7: список ключей

Затем копируем наш сгенерированный PGP ключ в буфер обмена и вставляем его при создании New GPG key в GitHub. (рис. fig. 8).

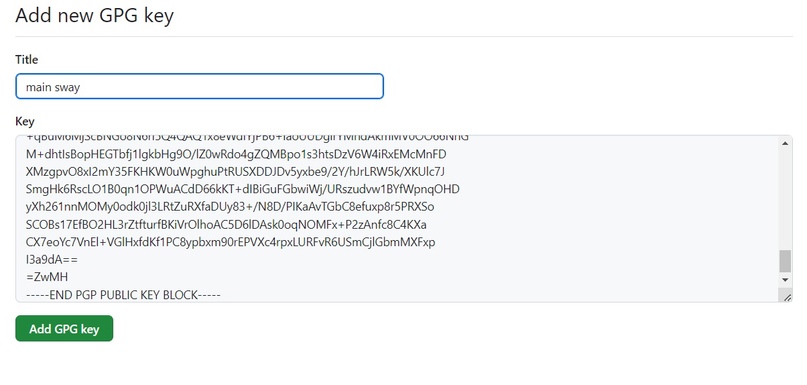


Рис. 8: New GPG key в GitHub

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

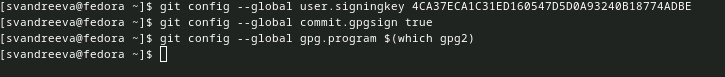


Рис. 9: email при подписи коммитов

Настройка gh.Для начала авторизуемся, ответив на несколько наводящих вопросов. (рис. fig. 10).

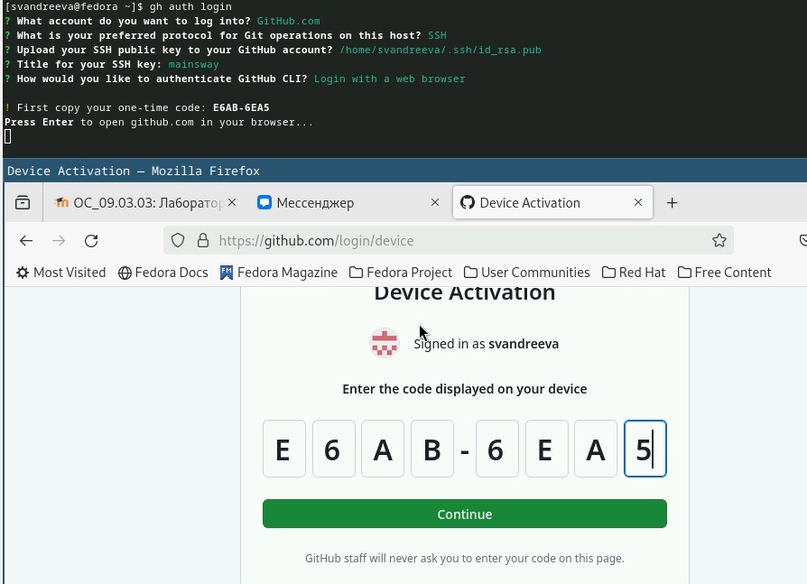


Рис. 10: Настройка gh

Создадим репозиторий, предварительно создав рабочее пространство (рис. fig. 11).

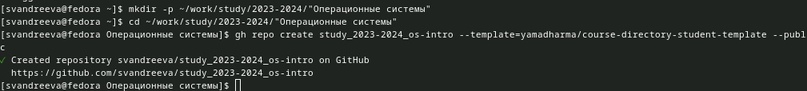


Рис. 11: Создадим репозиторий

Клонируем репозиторий (рис. fig. 12).

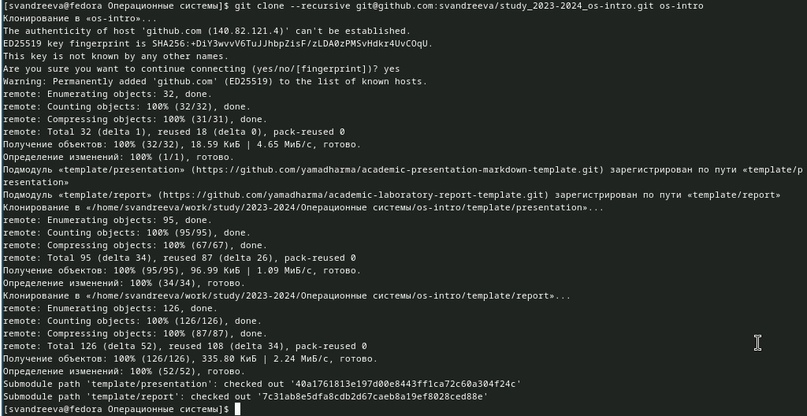


Рис. 12: Клонируем репозиторий

Перейдем в каталог курса, удалим лишние файлы и создадим необходимые каталоги(рис. fig. 13).

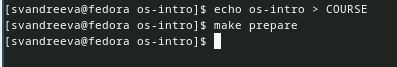


Рис. 13: создадим необходимые каталоги

Отправим файлы на сервер (рис. fig. 14).

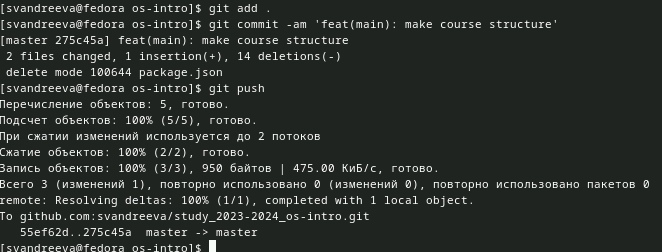


Рис. 14: Отправим файлы на сервер

# 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).

1. Централизованные и децентрализованные VCS:

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

1. Единоличная работа с хранилищем:

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

1. 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.

1. Основные задачи, решаемые инструментальным средством git:

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

1. 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.

1. Примеры команд для работы с локальным и удалённым репозиториями git push -all (push origin master/любой branch)
2. 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.
3. 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.