Шаблон отчёта по лабораторной работе № 2

Markdown

Скобеева Алиса Алексеевна

Содержание

# 1 Цель работы

Научиться работать с Markdown и составлять с помощью него отчеты к лабораторным работам.

# 2 Задание

Сделать отчет к лабораторной работе № 2.

# 3 Выполнение лабораторной работы

1. Установка программного обеспечения

Устанавливаем git.

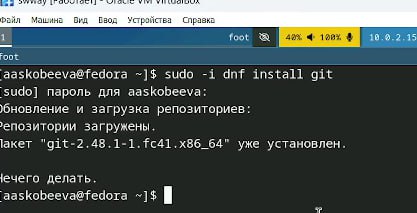


Рис. 1: Вводим необходимую команду

Устанавливаем gh.

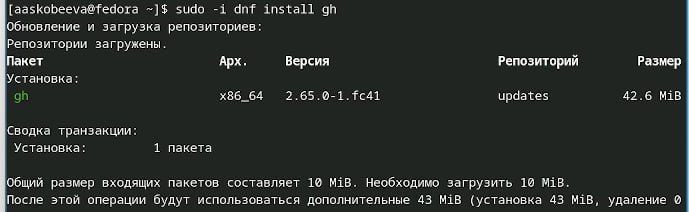


Рис. 2: Вводим необходимую команду

1. Базовая настройка git. Задаем имя и email владельца репозитория, настраиваем utf-8 в выводе сообщений git, настраиваем верификацию и подписание коммитов, задаем имя начальной ветки, устанавливаем параметр autocrlf, а также параметр safecrlf.

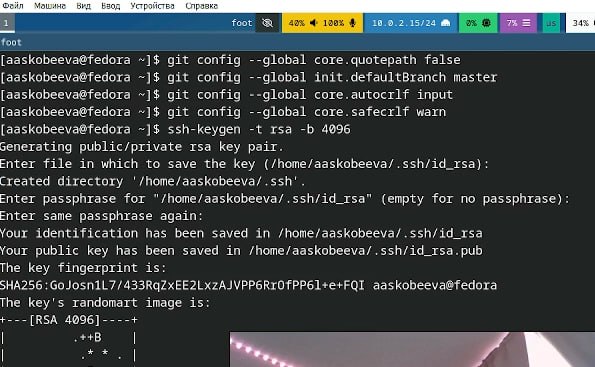


Рис. 3: Вводим необходимые команды для установки

1. Создаем ключи ssh

По алгоритму rsa с ключем размером 4096 бит(см. изображение 3 выше)

По алгоритму ed25519:

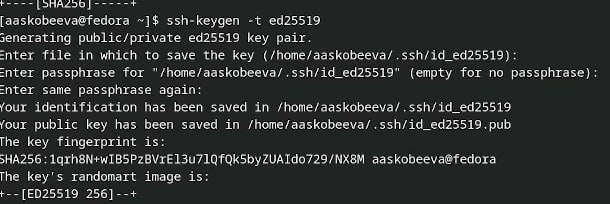


Рис. 4: Вводим ssh-keygen -t ed25519

1. Создание ключей pgp

Генерируем ключ с помощью команды gpg –full-generate-key

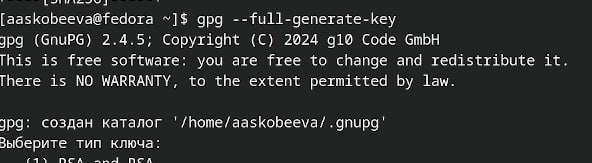


Рис. 5: Ввод и выполнение команды

Иллюстрация выбора из предложенных опций:

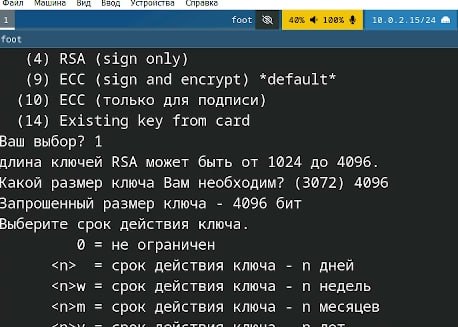


Рис. 6: Выбираем необходимые параметры

1. Настройка github

Поскольку учетная запись у меня была создана и заполнена ранее, данный пункт пропускаем.

1. Добавление PGP ключа в GitHub

Выводим список ключей, копируем отпечаток приватного ключа - в нашем случае это адрес электронной почты.

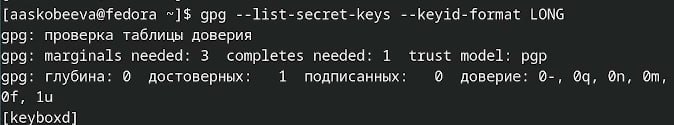


Рис. 7: Ввод команды

Копируем сгенерированный PGP ключ в буфер обмена.



Рис. 8: Пробуем это сделать

Поскольку с помощью команды скопировать ключ не вышло, скопируем его вручную, воспользовавшись командой cat:

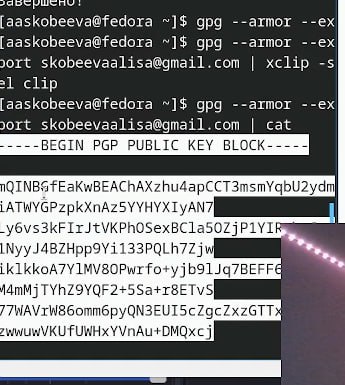


Рис. 9: Копируем ключ вручную

Вставляем ключ в настройках GitHub:

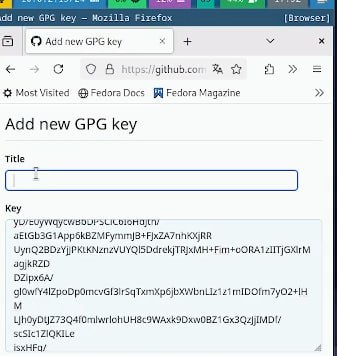


Рис. 10: Вставляем в необходимое окно ключ

1. Настройка автоматических подписей коммитов git

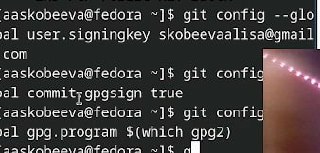


Рис. 11: Используя введенный email, указываем Git применять его при подписи коммитов

1. Создание репозитория курса

Вводим необходимые команды для создания каталогов и подключения рабочего пространства:

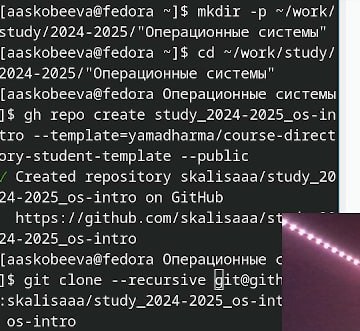


Рис. 12: Вводим необходимую команду

1. Настройка каталога курса

Переходим в каталог курса, удаляем лишние файлы и создаем необходимые каталоги

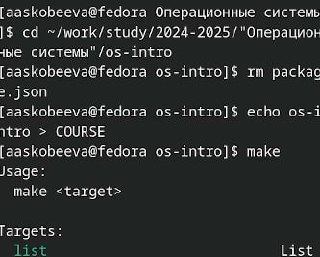


Рис. 13: Вводим необходимые команды

В конце отправляем файлы на сервер с помощью команд:

git add . git commit -am ‘feat(main): make course structure’ git push

# 4 Контрольные вопросы

1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются?

* Системы контроля версий (VCS) - это системы для отслеживания изменений файлов проекта. Предназначаются для следующих задач: совместная разработка, версионность, откат к предыдущим версиям, управление ветвями.

1. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия.
   * Хранилище: База данных со всеми версиями проекта.
   * Commit: Запись изменений с комментарием.
   * История: Последовательность коммитов.
   * Рабочая копия: Локальная копия файлов для работы. (Отношения: Рабочая копия -> Commit -> История в Хранилище).
2. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида.
   * Централизованные VCS: Одно центральное хранилище. Пример: SVN.
   * Децентрализованные VCS: Полная локальная копия хранилища у каждого. Пример: Git.
3. Опишите действия с VCS при единоличной работе с хранилищем.

* Единоличная работа: git init, git add, git commit, git log, git checkout.

1. Опишите порядок работы с общим хранилищем VCS.

* Общее хранилище: git clone, git branch, git checkout, git add, git commit, git push, pull request, code review, merge.

1. Каковы основные задачи, решаемые инструментальным средством git?

* Git: Версионность, совместная разработка, управление ветвями, удаленное хранение, разрешение конфликтов.

1. Назовите и дайте краткую характеристику командам git.

* git init (создание), git clone (копирование), git add (добавление), git commit (фиксация), git push (отправка), git pull (получение), git branch (ветки), git merge (слияние).

1. Приведите примеры использования при работе с локальным и удалённым репозиториями.
   * Локальный: git init, git add ., git commit -m “Initial commit”.
   * Удаленный: git clone , git push origin .
2. Что такое и зачем могут быть нужны ветви (branches)?

* Ветви: Параллельные линии разработки. Зачем: разработка новых функций, эксперименты, управление релизами.

1. Как и зачем можно игнорировать некоторые файлы при commit?

* Игнорирование файлов: Файл .gitignore. Зачем: избегать захламления хранилища, безопасность, производительность.

# 5 Выводы

Мы научились работать с Markdown и составлять отчеты с его помощью.