Лабораторная работа №2

Операционные системы

Клюкин Михаил Александрович

Содержание

# 1 Цель работы

* Изучить идеалогию и пременение средств контроля версий
* Научиться работать с git

# 2 Задание

1. Установить программное обеспечение
2. Провести базовую настройку git
3. Создать ключ SSH
4. Создать ключи pgp
5. Провести настройку github
6. Добавить ключ PGP на Github
7. Настроить автоматическое подписываение коммитов git
8. Настроить gh
9. Создать шаблон рабочего пространства

# 3 Теоретическое введение

Системы контроля версий используются при работе нескольких человек над одним проектом.  
Системы контроля версий хранят дерево проекта в локальном или удаленном репозитории, к которому имеют доступ все участники проекта.  
При внесении измений в проект система контроля версий позволяет фиксировать эти изменения, совмещать их, возвращаться к более ранним версиям проекта.  
Бывают распределенные и централизованные VCS. Мы будем работать с распределенной системой контроля версий – Git. Также к распределенным системам контроля версий относятся такие VCS как Mercurial, Bazaar, Darcs.  
В распределенных системах контроля версий в локальных репозиториях пользователей хранятся полные копии всего репозитория проекта.

# 4 Выполнение лабораторной работы

## 4.1 Установка и настройка программного обеспечения

Установили git, установили gh (Рис. [1](#fig:001)).

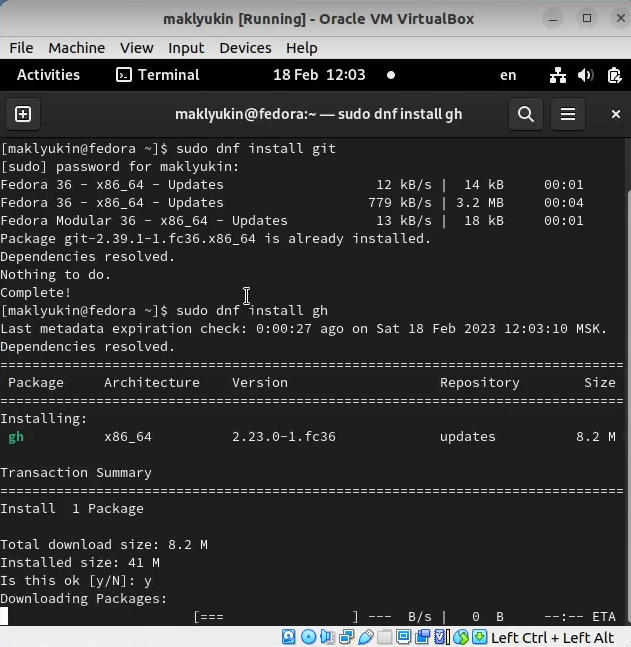


Figure 1: Установка git и gh

## 4.2 Базовая настройка git

Задали имя и email владельца репозитория (Рис. [1](#fig:001)):

‘git config –global user.name “Name Surname”’ git config –global user.email “user@email”

Настроили utf-8 в выводе сообщений git (Рис. [1](#fig:001)):

git config –global core.quotepath false

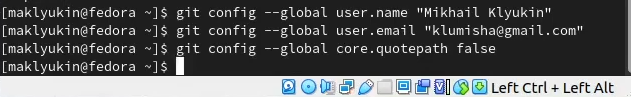


Figure 2: Настойка git

Настроили верификацию и подписывание коммитов git:

* Сгенерировали ключ (Рис. [1](#fig:001))  
  gpg –full-generate-key
* Выбрали следующие опции – RSA and RSA, 4096, 0 (Рис. [3](#fig:003))

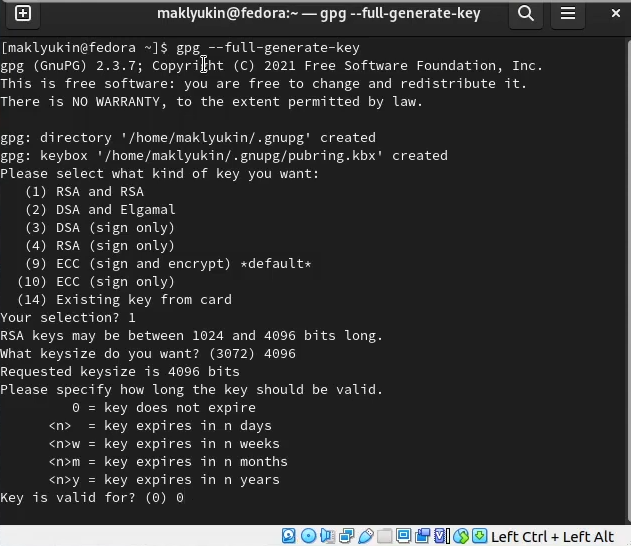


Figure 3: Генерация ключа и выбор опций

* Указали личную информацию – имя, адрес электронной почты (Рис. [4](#fig:004))

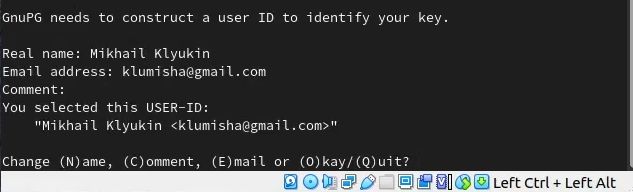


Figure 4: Ввод личной информации

* Произвели экспорт ключа и добавили его на Github (Рис. [5](#fig:005), [6](#fig:006)) gpg –list-secret-keys –keyid-format LONG gpg –armor –export

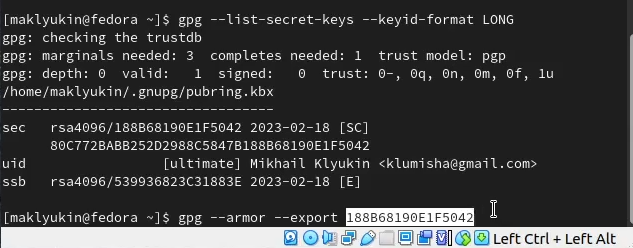


Figure 5: Экспорт ключа

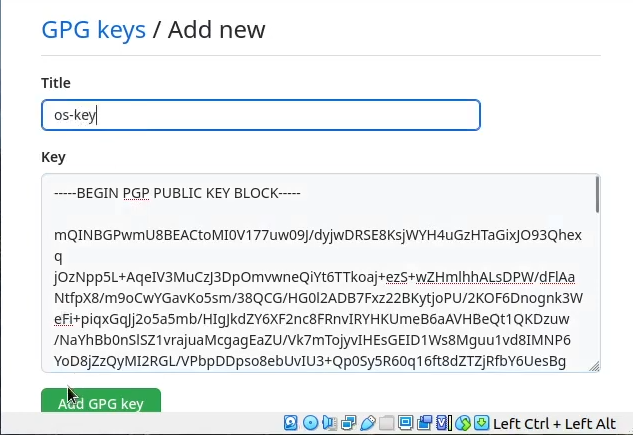


Figure 6: Экспорт ключа

* Произвели нвстройку автоматических подписей коммитов git (Рис. [7](#fig:007))

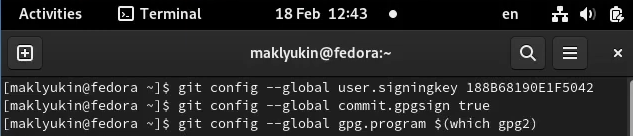


Figure 7: Настройка автоматических подписей коммиитов

Задали имя начальной ветки, параметр autocrlf, параметр safecrlf (Рис. [8](#fig:008))

git config – global init.defaultBranch master  
git config – global core.autocrlf input  
git config – global core.safecrlf warn

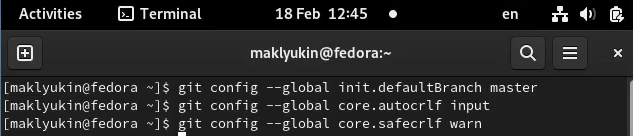


Figure 8: Задание параметров и имени

Создали ключ ssh по алгоритму rsa с размером 4096 бит, по алгоритму ed25519 (Рис. [9](#fig:009))

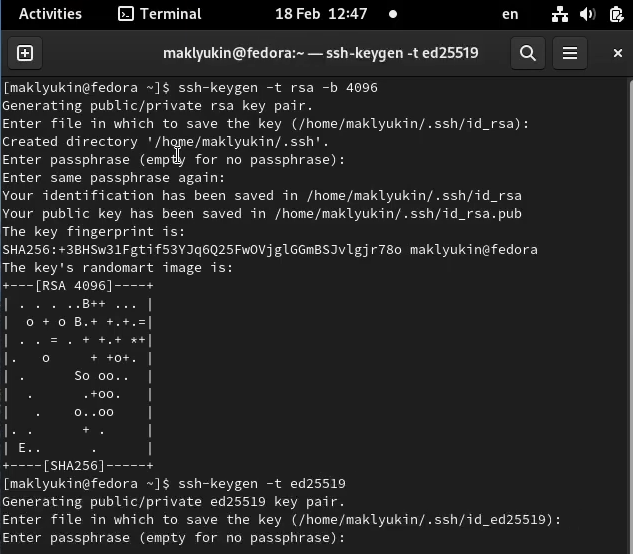


Figure 9: Создание ssh ключа

Создали учетную запись на github и провели базовую настройку профиля.

Провели настройку gh (Рис. [10](#fig:010))

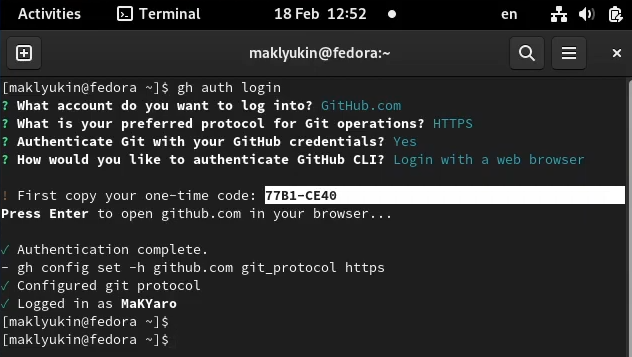


Figure 10: Настройка gh

## 4.3 Настройка рабочего пространства

Создали шаблон рабочего пространства (Рис. [11](#fig:011), [12](#fig:012), [13](#fig:013))

mkdir -p ~/work/study/2022-2023/“Операционные системы”  
cd ~/work/study/2022-2023/“Операционные системы”  
gh repo create study\_2022-2023\_os-intro –template=yamadharma/course-directory-student-template –public  
git clone –recursive

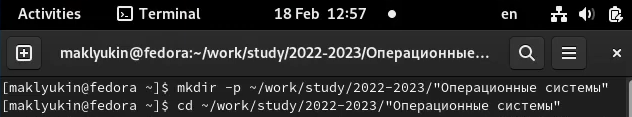


Figure 11: Создание каталога

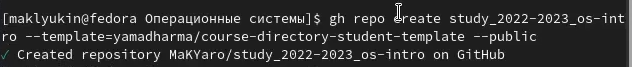


Figure 12: Создание репозитория

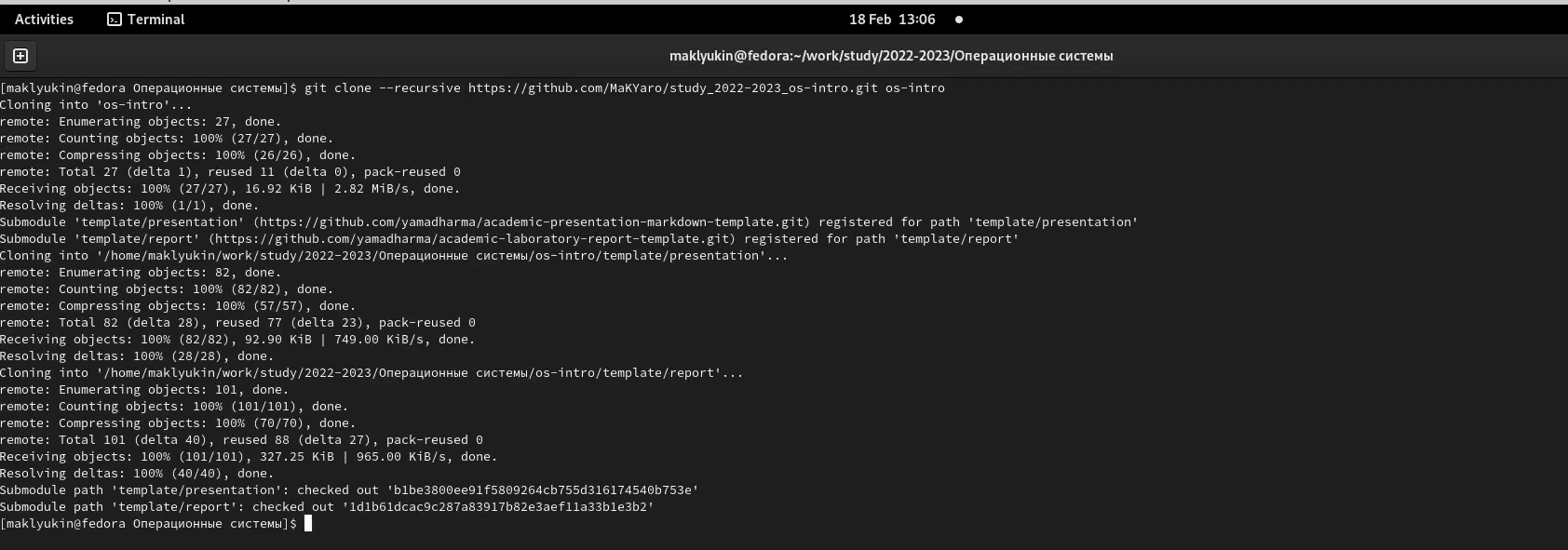


Figure 13: Клонирование репозитория

Произвели настройку каталога курса. Для этого перешли в каталог курса, удалили лишние файлы, создали необходимые каталоги, отправили файлы на сервер (Рис. [14](#fig:014), [15](#fig:015))

cd ~/work/study/2022-2023/“Операционные системы”/os-intro  
rm package.json  
echo os-intro > COURSE  
make  
git add .  
git commit -am ‘feat(main): make course structure’  
git push

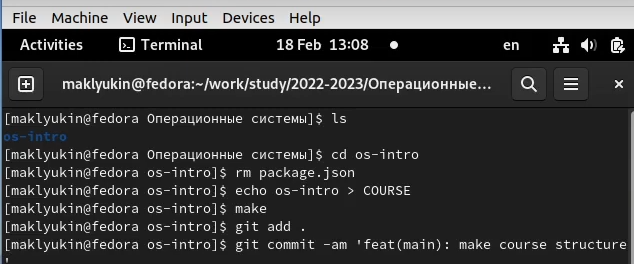


Figure 14: Настройка репозитория

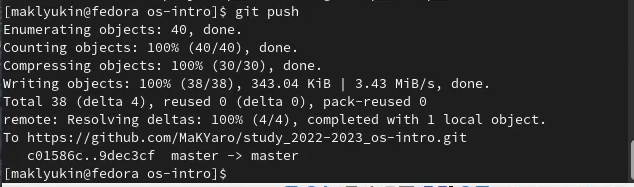


Figure 15: Настройка репозитория

# 5 Выводы

Изучили идеалогию и научились применять системы контроля версий. Научились работать с git

# 6 Ответы на контольные вопросы

1. Что такое системы контроля версий? Для решения каких задач они используются?

Системы контроля версий – это системы, записывающая изменения в файл или набор файлов втечение времени и позволяющая вернуться к определенной версии.  
VCS используются для работы нескольких человек над одним проектом.

1. Что такое хранилище, коммит, история, рабочая копия? Как они они связаны между собой?

Хранилище – это место, в котором хранятся и поддерживаются файлы.  
Коммит – это сохранение изменений.  
История – все измененения в проекте.  
Рабочая копия – текущее состояние файлов проекта

1. Что представляют собой централизованные и распределенные системы контроля версий?

Централизованные системы имеют одно хранилище всего проекта, каждый пользователь копирует необходимые файлы, изменяет их и добавляет изменения обратно (Subversion, CVS, TFS, AccuRev).  
В распределенных системах контроля версий у каждого пользователя свой вариант репозитория. В них присутствует возвможность добавлять изменения из любого репозитория (Git, Mercurial, Bazaar).

1. Опишите действия с VCS при единоличной работе с хранилищем.

Создаем и подключаем рабочий репозиторий. В ходе работы над проектом отправляем все изменения на сервер.

1. Опишите порядок работы с общим хранилищем.

Получаем нужную версию фалов, вносим изменения, размещаем новую версию в хранилище. Предыдущие версии не удаляются, к ним можно всегда вернуться.

1. Каковы основные задачи, решаемые инструментальным средством git?

Хранит информацию о всех изменениях, обеспечивает комфортную командную работу над проектом.

1. Назовите и дайте краткую характеристику командам git/

* git init – создает основное дерево репозитория
* git pull – вытягивает обновления текущего дерева из центрального репозитория
* git push – отправляет все произведенные изменения локального дерева в центральный репозиторий
* git status – позволяет просмотреть список измененных файлов в текущей директории
* git diff – позволяет посмотреть текущие изменения
* git add – добавляет все измененные или (и) созданные каталоги (файлы)
* git rm – удаляеи файл и (или) каталог из индекса репозитория
* git commit – сохраняет добавленные изменения
* git checkout – изпользуется для переключения на ветку

1. Приведите примеры использования при работе с локальным и удаленным репозиториями.

git push -all  
git push origin main

1. Что такое и зачем нужны ветки?

Ветки – параллельные участки истории изменений  
Ветки используются при разработке новых функций

1. Как и зачем можно игнорировать некоторые файлы?

Для того, чтобы игнорировать файлы, их названия нужно добавить в файл .gitignore.  
Некоторые файлы не требуется добавлять в репозиторий. Именно такие файлы и игнорируются.

# 7 Список литературы

1. Демидова А. В. Лабораторная работа №3. Система контроля версий Git – Методическое пособие
2. Система контроля версий git [электронный ресурс]: https://git-scm.com/book/en/v2