Отчёт по лабораторной работе №2

Операционные системы

Рогожина Надежда Александровна, НКАбд-02-22

Содержание

# 1 Цель работы

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

# 2 Задание

1. Установка программного обеспечения

* Установка git
* Установка gh

1. Базовая настройка git
2. Создайте ключи ssh
3. Создайте ключи pgp
4. Настройка github
5. Добавление PGP ключа в GitHub
6. Настройка автоматических подписей коммитов git
7. Настройка gh
8. Шаблон для рабочего пространства

* Сознание репозитория курса на основе шаблона
* Настройка каталога курса

# 3 Теоретическое введение

## 3.1 Примеры использования git:

* Система контроля версий Git представляет собой набор программ командной строки. Доступ к ним можно получить из терминала посредством ввода команды git с различными опциями.
* Благодаря тому, что Git является распределённой системой контроля версий, резервную копию локального хранилища можно сделать простым копированием или архивацией.

## 3.2 Основные команды git:

Например, в табл. [1](#tbl:std-dir) приведено краткое описание основных команд Git.

Table 1: Описание некоторых команд системы контроля версий Git

| Команда | Описание команды |
| --- | --- |
| git init | Создание основного дерева репозитория |
| 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 diff | Просмотр текущих изменений |
| git add . | Добавление все изменённые и/или созданные файлы и/или каталоги |
| git rm имена\_файлов | Удаление файлов и/или каталогов из индекса репозитория |
| git commit -am ‘Описание коммита’ | Сохранение всех добавленных изменений и всех изменённых файлов |
| git commit | Сохранение добавленный изменений с внесением комментария через встроенный редактор |
| git checkout -b имя\_ветки | Создание новой ветки, базирующейся на текущей |
| git branch -d имя\_ветки | Удаление локальной уже слитой с основным деревом ветки |
| git branch -D имя\_ветки | Принудительное удаление локальной ветки |

Полный список команд можно посмотреть на официальном сайте: [Github.com](https://docs.github.com/en/get-started/using-github/github-command-palette)

# 4 Выполнение лабораторной работы

Первоначально установим гит(рис. [1](#fig:001)):

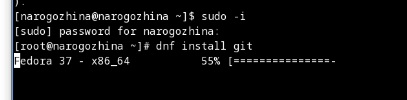


Figure 1: Установка системы контроля версий GIT

Производим первичную настройку параметров git, а именно:

1. Зададим имя и email владельца репозитория(рис. [2](#fig:002)):

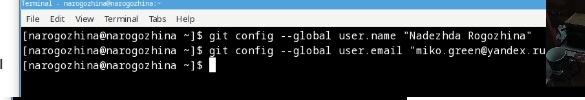


Figure 2: Первичная настройки git

1. Настроим utf-8 в выводе сообщений, а также зададим имя начальной ветки, установим параметры **autocrlf** и **safecrlf**(рис. [3](#fig:003)):

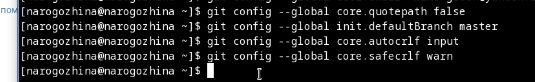


Figure 3: Базовая настройка git

1. Создадим ключ *ssh* по алгоритму *rsa* с ключём размером 4096 бит(рис. [4](#fig:005)):

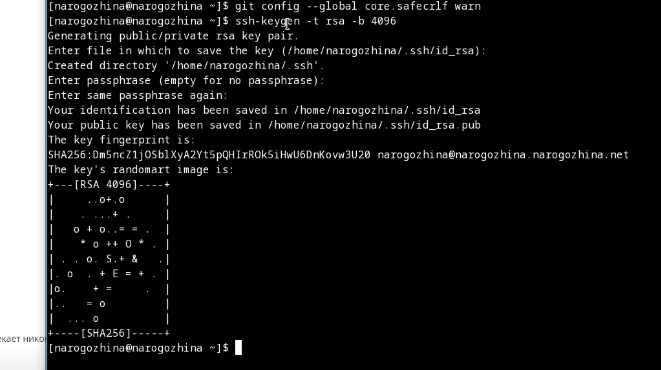


Figure 4: Создание ssh-ключа

1. Создадим gpg-ключ(рис. [5](#fig:006))(рис. [6](#fig:007))(рис. [7](#fig:008))(рис. [8](#fig:009)):

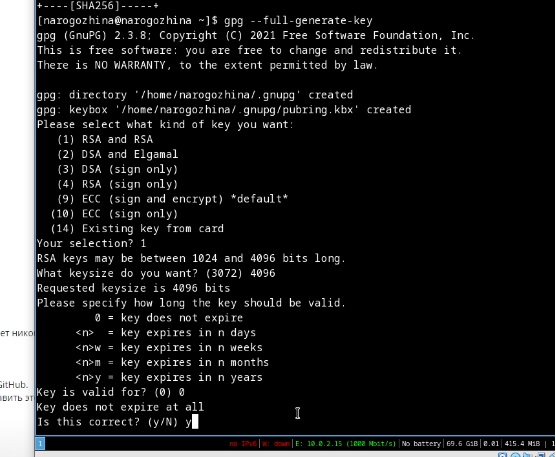


Figure 5: Создание gpg-ключа



Figure 6: Создание gpg-ключа (прод.)

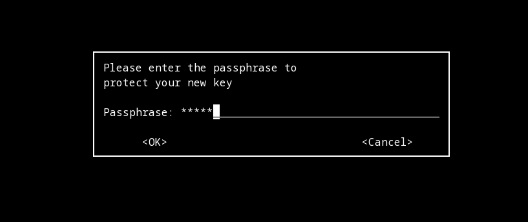


Figure 7: Ввод ключевой фразы

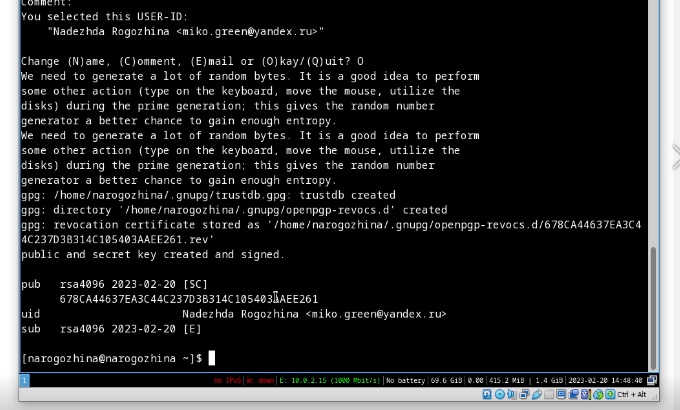


Figure 8: Создание gpg-ключа

1. Настройка github(рис. [9](#fig:010)):

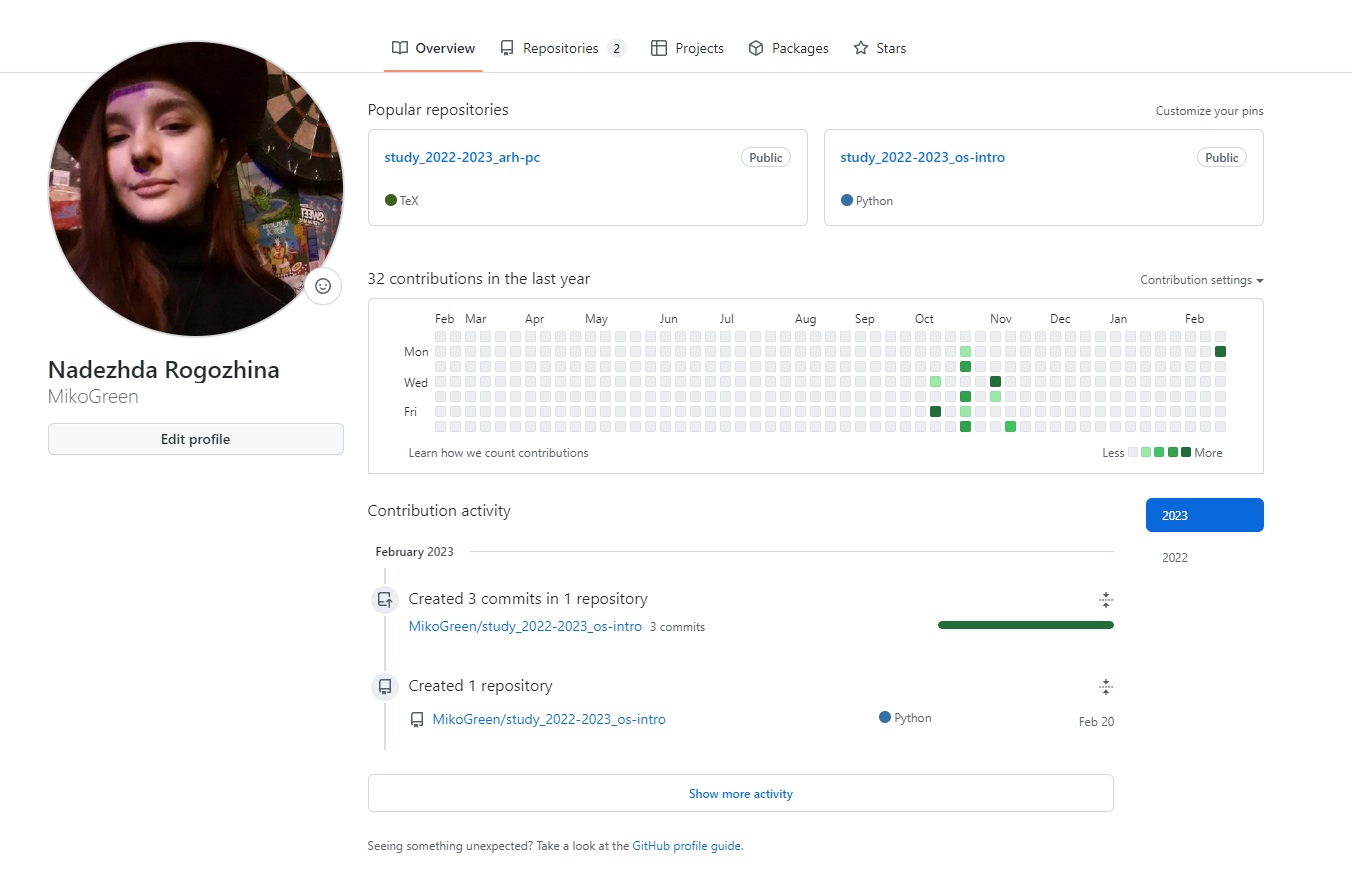


Figure 9: Аккаунт github

1. Добавим PGP ключ в GitHub(рис. [10](#fig:011))(рис. [11](#fig:012))(рис. [12](#fig:013)):

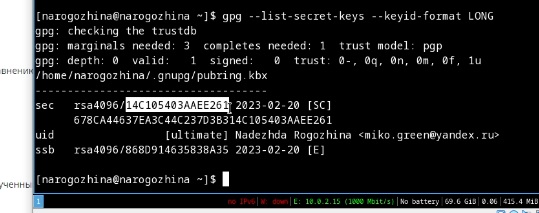


Figure 10: Вывод списка ключей

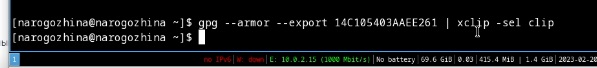


Figure 11: Копирование отпечатка ключа

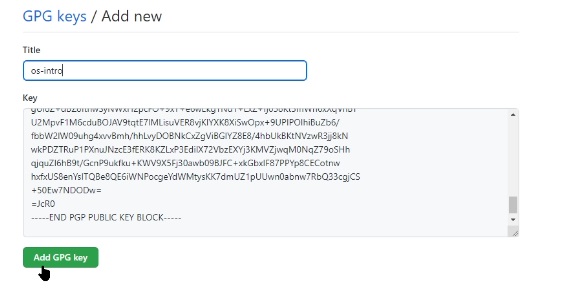


Figure 12: Добавление ключа на github.com

1. Настройка автоматических подписей коммитов git(рис. [13](#fig:014)):

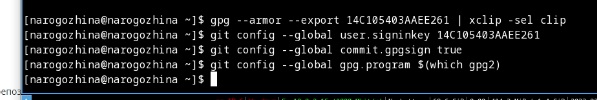


Figure 13: Настройка коммитов

1. Настройка и авторизация в gh(рис. [14](#fig:015))(рис. [15](#fig:016))(рис. [16](#fig:017)):

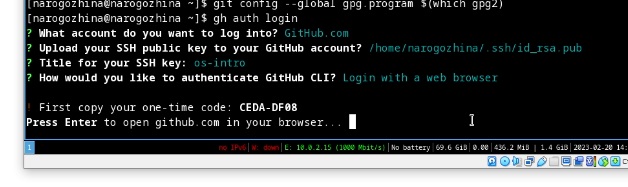


Figure 14: Авторизация в gh

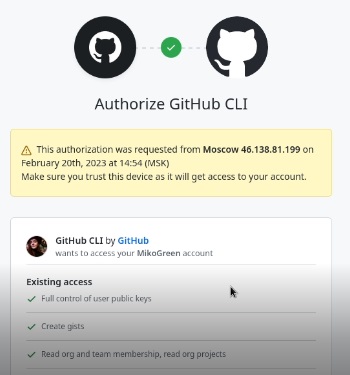


Figure 15: Подтверждение авторизации через браузер

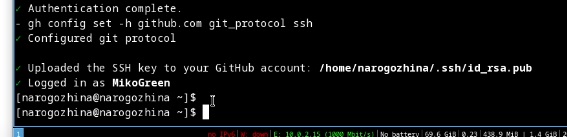


Figure 16: Подтверждение авторизации

1. Создание репозитория курса на основе шаблона(рис. [17](#fig:018))(рис. [18](#fig:019)):

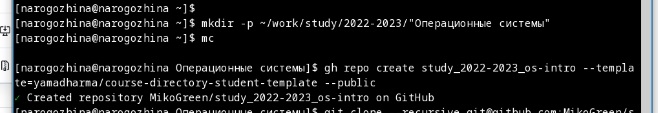


Figure 17: Создание репозитория и копирование шаблона

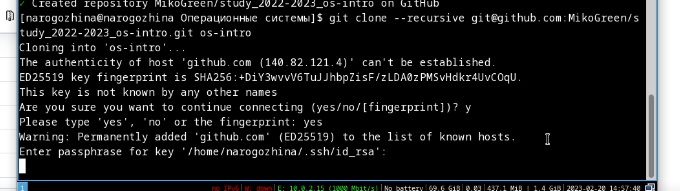


Figure 18: Подтвержение

1. Настройка репозитория и каталогов(рис. [19](#fig:020)):

Удаление лишний файлов, а также создание необходимых каталогов:

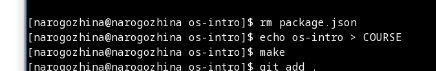


Figure 19: Настройка каталогов

1. Отправка файлов на сервер(рис. [20](#fig:021)):

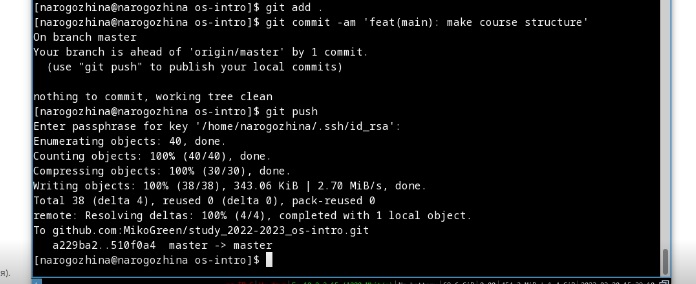


Figure 20: Отправка файлов на сервер

# 5 Выводы

В ходе данной лабораторной работы мы приобрели навыки работы с системой контроля версий git.

# 6 Контрольные вопросы

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

Системы контроля версий (Version Control System, VCS) применяются при работе нескольких человек над одним проектом. Обычно основное дерево проекта хранится в локальном или удалённом репозитории, к которому настроен доступ для участников проекта. При внесении изменений в содержание проекта система контроля версий позволяет их фиксировать, совмещать изменения, произведённые разными участниками проекта, производить откат к любой более ранней версии проекта, если это требуется.

1. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия.

**Хранилище** - место хранения всех версий и служебной информации.

**Commit** - (==версия) создание новой версии.

**История** - журнал изменений.

**Рабочая копия** - текущее состояние файлов проекта, основанное на версии из хранилища.

1. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида.

**Централированные** VCS имеют одно основное хранилище всего проекта, достаточно просты в использовании. Участник проекта (пользователь) перед началом работы посредством определённых команд получает нужную ему версию файлов. После внесения изменений, пользователь размещает новую версию в хранилище. При этом предыдущие версии не удаляются из центрального хранилища и к ним можно вернуться в любой момент. Сервер может сохранять не полную версию изменённых файлов, а производить так называемую дельта-компрессию — сохранять только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных. Необходимо подключение к сети. Например:

* Subversion
* CVS
* TFS, VAULT
* AccuRev

**Децентрализованные** VCS характеризуются тем, что у каждого пользователя свой вариант репозитория. Присутствует возможность добавлять и забирать изменения из любого репозитория. Подключение к сети не нужно. Например:

* Git
* Mercurial
* Bazaar

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

Системы контроля версий поддерживают возможность отслеживания и разрешения конфликтов, которые могут возникнуть при работе нескольких человек над одним файлом. Можно объединить (слить) изменения, сделанные разными участниками (автоматически или вручную), вручную выбрать нужную версию, отменить изменения вовсе или заблокировать файлы для изменения. В зависимости от настроек блокировка не позволяет другим пользователям получить рабочую копию или препятствует изменению рабочей копии файла средствами файловой системы ОС, обеспечивая таким образом, привилегированный доступ только одному пользователю, работающему с файлом.

1. Опишите порядок работы с общим хранилищем VCS.

Системы контроля версий поддерживают возможность отслеживания и разрешения конфликтов, которые могут возникнуть при работе нескольких человек над одним файлом. Можно объединить (слить) изменения, сделанные разными участниками (автоматически или вручную), вручную выбрать нужную версию, отменить изменения вовсе или заблокировать файлы для изменения. В зависимости от настроек блокировка не позволяет другим пользователям получить рабочую копию или препятствует изменению рабочей копии файла средствами файловой системы ОС, обеспечивая таким образом, привилегированный доступ только одному пользователю, работающему с файлом. Системы контроля версий также могут обеспечивать дополнительные, более гибкие функциональные возможности. Например, они могут поддерживать работу с несколькими версиями одного файла, сохраняя общую историю изменений до точки ветвления версий и собственные истории изменений каждой ветви. Кроме того, обычно доступна информация о том, кто из участников, когда и какие изменения вносил. Обычно такого рода информация хранится в журнале изменений, доступ к которому можно ограничить.

1. Каковы основные задачи, решаемые инструментальным средством git?

У Git две основных задачи: первая — хранить информацию о всех изменениях в вашем коде, начиная с самой первой строчки, а вторая — обеспечение удобства командной работы над кодом.

Репозиторий Git — это место, где хранится ваш код и вся информация о его изменениях. Репозитории могут находиться у вас на компьютере, на компьютерах ваших коллег и на удалённом сервере.

1. Назовите и дайте краткую характеристику командам git.

Основные команды git:

| Команда | Описание команды |
| --- | --- |
| git init | Создание основного дерева репозитория |
| 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 diff | Просмотр текущих изменений |
| git add . | Добавление все изменённые и/или созданные файлы и/или каталоги |
| git rm имена\_файлов | Удаление файлов и/или каталогов из индекса репозитория |
| git commit -am ‘Описание коммита’ | Сохранение всех добавленных изменений и всех изменённых файлов |
| git commit | Сохранение добавленный изменений с внесением комментария через встроенный редактор |
| git checkout -b имя\_ветки | Создание новой ветки, базирующейся на текущей |
| git branch -d имя\_ветки | Удаление локальной уже слитой с основным деревом ветки |
| git branch -D имя\_ветки | Принудительное удаление локальной ветки |

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

Удаленный репозиторий – это версии вашего проекта, сохраненные на удаленном сервере. Доступ к репозиторию на таком сервере может осуществляться по интернету или по локальной сети. Если мы подключим удаленный репозиторий к своему локальному, то у нас появятся копии всех ссылочных объектов удаленного репозитория. То есть, например, у удаленного репозитория есть ветка main, а у нас будет копия этой ветки – origin/main. Все такие ссылочные объекты (указатели, ветки и теги) удаленного репозитория хранятся почти там же, где и у локального – в директории .git/refs/remotes/.

1. Что такое и зачем могут быть нужны ветви (branches)?

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.

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

Git рассматривает каждый файл в вашей рабочей копии как файл одного из трех нижеуказанных типов.

Отслеживаемый файл — файл, который был предварительно проиндексирован или зафиксирован в коммите. Неотслеживаемый файл — файл, который не был проиндексирован или зафиксирован в коммите. Игнорируемый файл — файл, явным образом помеченный для Git как файл, который необходимо игнорировать. Игнорируемые файлы — это, как правило, артефакты сборки и файлы, генерируемые машиной из исходных файлов в вашем репозитории, либо файлы, которые по какой-либо иной причине не должны попадать в коммиты.

Вы можете создать файл .gitignore в корневом каталоге репозитория, чтобы сообщить Git, какие файлы и каталоги следует игнорировать при фиксации. Чтобы поделиться правилами игнорирования с другими пользователями, которые клонируют репозиторий, зафиксируйте файл .gitignore в репозитории.

# Список литературы

1. [Руководство по лабораторной работе №2](https://esystem.rudn.ru/mod/page/view.php?id=970819#org3b74ce0)
2. [Основная информация LINUX](https://uii.bitbucket.io/study/courses/cs/lecture02_vcs.slides-2021.pdf)
3. [Cистема контроля версий. Горвиц Евгений](https://glebradchenko.susu.ru/courses/bachelor/engineering/2016/SUSU_SE_2016_REP_3_VCS.pdf)
4. [Руководство пользования GIT](https://docs.github.com/ru/get-started/getting-started-with-git/ignoring-files)