Отчёт по лабораторной работе №2

Дисциплина: Архитектура компьютеров и операционные системы

Гусев Степан Андреевич

Содержание

[1. Цель работы 1](#__RefHeading___Toc571_4185069229)

[2. Задание 2](#__RefHeading___Toc573_4185069229)

[3. Теоретическое введение 2](#__RefHeading___Toc575_4185069229)

[4. Выполнение лабораторной работы 3](#__RefHeading___Toc577_4185069229)

[4.1 Настройка GitHub 3](#__RefHeading___Toc579_4185069229)

[4.2 Базовая настройка Git 4](#__RefHeading___Toc581_4185069229)

[4.3 Создание SSH-ключа 4](#__RefHeading___Toc583_4185069229)

[4.4 Создание рабочего пространства и репозитория курса на основе шаблона 5](#__RefHeading___Toc585_4185069229)

[4.5 Создание репозитория курса на основе шаблона 5](#__RefHeading___Toc587_4185069229)

[4.6 Настройка каталога курса 7](#__RefHeading___Toc589_4185069229)

[4.7 Выполнение заданий для самостоятельной работы. 8](#__RefHeading___Toc591_4185069229)

[5. Выводы 11](#__RefHeading___Toc593_4185069229)

[6. Список литературы 11](#__RefHeading___Toc595_4185069229)

# 1. Цель работы

Изучить идеологию и применение средств контроля версий и научиться работать с системой контроля версий git.

# 2. Задание

1. Настройка GitHub.
2. Базовая настройка Git.
3. Создание SSH-ключа.
4. Создание рабочего пространства и репозитория курса на основе шаблона.
5. Создание репозитория курса на основе шаблона.
6. Настройка каталога курса.
7. Выполнение заданий для самостоятельной работы.

# 3. Теоретическое введение

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. В классических системах контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов. Выполнение большинства функций по управлению версиями осуществляется специальным сервером. 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. Системы контроля версий поддерживают возможность отслеживания и разрешения конфликтов, которые могут возникнуть при работе нескольких человек над одним файлом. Можно объединить изменения, сделанные разными участниками, 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. 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. В отличие от классических, в распределённых системах контроля версий центральный репозиторий не является обязательным. Среди классических VCS наиболее известны CVS, Subversion, а среди распределённых — Git, Bazaar, Mercurial. Принципы их работы схожи, отличаются они в основном синтаксисом используемых в работе команд. 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. Работа пользователя со своей веткой начинается с проверки и получения изменений из центрального репозитория (при этом в локальное дерево до начала этой процедуры не должно было вноситься изменений). Затем можно вносить изменения в локальном дереве и/или ветке. После завершения внесения какого-то изменения в файлы и/или каталоги проекта необходимо разместить их в центральном репозитории.

# 4. Выполнение лабораторной работы

## 4.1 Настройка GitHub

Создал учётную запись на сайте GitHub ([рис. 1](#fig-001)).

|  |
| --- |
| Рисунок 1: Аккаунт GitHub |

## 4.2 Базовая настройка Git

Сделал предварительную конфигурацию git. В терминале ввёл следующие команды, указав свои имя и email. Сделал предварительную конфигурацию git. В терминале ввёл следующие команды, указав свои имя и email. Задал имя master для начальной ветки. Задал параметры autocrlf и safecrlf ([рис. 2](#fig-002)).

|  |
| --- |
| Рисунок 2: Предварительная конфигурация git |

## 4.3 Создание SSH-ключа

Сгенерировал пару ключей: приватный и открытый ([рис. 3](#fig-003)).

|  |
| --- |
| Рисунок 3: Генерация ключей |

Скопировал ключ в буфер обмена ([рис. 4](#fig-004)).

|  |
| --- |
| Рисунок 4: Копирование ключа |

На сайте GitHub в меню settings в разделе SSH and GPG keys добавил только что созданный SSH-ключ ([рис. 5](#fig-005)).

|  |
| --- |
| Рисунок 5: Добавление ключа |

## 4.4 Создание рабочего пространства и репозитория курса на основе шаблона

Создал каталог для предмета для предмета “Архитектура компьютера” ([рис. 6](#fig-006)).

|  |
| --- |
| Рисунок 6: Создание каталога |

## 4.5 Создание репозитория курса на основе шаблона

В браузере перешёл на страницу репозитория с шаблоном курса. Далее нажал “Use this template”, чтобы использовать этот шаблон для своего репозитория ([рис. 7](#fig-007)).

|  |
| --- |
| Рисунок 7: Шаблон курса |

Задал имя репозитория и создал его ([рис. 8](#fig-008)).

|  |
| --- |
| Рисунок 8: Создание каталога |

В терминале перешёл в каталог курса ([рис. 9](#fig-009)).

|  |
| --- |
| Рисунок 9: Перемещение между директориями |

Клонировал созданный репозиторий ([рис. 10](#fig-010)).

|  |
| --- |
| Рисунок 10: Клонирование репозитория |

## 4.6 Настройка каталога курса

Перешёл в каталог курса ([рис. 11](#fig-011)).

|  |
| --- |
| Рисунок 11: Перемещение между директориями |

Создал необходимые каталоги ([рис. 12](#fig-012)).

|  |
| --- |
| Рисунок 12: Создание директорий |

Отправил файлы на сервер ([рис. 13](#fig-013))([рис. 14](#fig-014)).

|  |
| --- |
| Рисунок 13: Загрузка файлов на сервер |

|  |
| --- |
| Рисунок 14: Загрузка файлов на сервер |

Проверил правильность создания иерархии рабочего пространства в локальном репозитории ([рис. 15](#fig-015)).

|  |
| --- |
| Рисунок 15: Проверка |

Проверил правильность создания иерархии рабочего пространства на странице GitHub ([рис. 16](#fig-016)).

|  |
| --- |
| Рисунок 16: Проверка на GitHub |

## 4.7 Выполнение заданий для самостоятельной работы.

Перешёл с помощью команды cd в каталог labs/lab02/report и создал в нём файл для отчёта по второй лабораторной работе с помощью команды touch. Проверил, что файл создан, с помощью команды ls ([рис. 17](#fig-017)).

|  |
| --- |
| Рисунок 17: Создание файлов |

Оформить отчёт я смогу в текстовом редакторе LibreOffice Writer, найдя его в меню приложений ([рис. 18](#fig-018)).

|  |
| --- |
| Рисунок 18: Поиск приложения |

После открытия редактора, открыл в нем созданный файл и начал работу над отчётом ([рис. 19](#fig-019)).

|  |
| --- |
| Рисунок 19: Открытие файла |

C помощью команды cd перешёл в каталог lab01/report и переместил в него отчёт по первой лабораторной работе с помощью команды mv. Проверил, что файл успешно перемещён, с помощью ls ([рис. 20](#fig-020)).

|  |
| --- |
| Рисунок 20: Перемещение файла |

С помощью команды git add добавил в коммит отчёты по двум лабораторным работам. Сохранил изменения на сервере командой git commit -m “Add existing file”, пояснив добавление файлов ([рис. 21](#fig-021))([рис. 22](#fig-022)).

|  |
| --- |
| Рисунок 21: git add |

Отправил в центральный репозиторий изменения командой git push -f origin master ([рис. 22](#fig-022)).

|  |
| --- |
| Рисунок 22: git push |

Проверил на сайте GitHub правильность выполнения заданий. Увидел, что пояснение к совершённым действиям отображается ([рис. 23](#fig-023)).

|  |
| --- |
| Рисунок 23: Проверка |

При просмотре изменений увидел, что были добавлены файлы с отчётами по лабораторным работам ([рис. 24](#fig-024)).

|  |
| --- |
| Рисунок 24: Проверка |

Проверил, что отчёты по лабораторным работам находятся в соответствующих каталогах репозитория: отчёт по первой лабораторной в lab01/report ([рис. 25](#fig-025)), по второй – в lab02/report ([рис. 26](#fig-026)).

|  |
| --- |
| Рисунок 25: Проверка |

|  |
| --- |
| Рисунок 26: Проверка |

# 5. Выводы

В ходе работы я приобрел практические навыки работы с системой контроля версий GitHub, разобрав следующие команды: clone, add, commit, push. Также изучил идеологию и применение подобных систем.

# 6. Список литературы

1. [Архитектура ЭВМ](https://esystem.rudn.ru/pluginfile.php/1584622/mod_resource/content/1/Лабораторная%20работа%20№3.pdf)
2. [Git - gitattributes Документация](https://git-scm.com/docs/gitattributes)