**РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ**

**Факультет физико-математических и естественных наук**

**Кафедра прикладной информатики и теории вероятностей**

**ОТЧЕТ**

**по лабораторной работе № 2**

*дисциплина: Архитектура компьютера*

Студент: Кузнецова А. С.

Группа: НБИбд-01-23

**МОСКВА**

2023 г.

**Содержание**

1. Цель работы
2. Теоретическое введение
3. Выполнение лабораторной работы
   1. Базовая настройка GetHub
   2. Создание SSH ключа
   3. Создание рабочего пространства и репозитория курса на основе шаблона
4. Вывод
5. **Цель работы**

Целью работы является изучить идеологию и применение

средств контроля версий. Приобрести практические навыки по работе с системой̆ git.

**2. 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, произведенные разными участниками проекта, производить откат к любой̆ более ранней̆ 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 любой̆ 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 (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. 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.

**3. Выполнение лабораторной̆ работы**

**3.1. Настройка GitHub**

Создаю учётную запись на сайте https://github.com/ и заполните основные данные.

Изображение выглядит как текст, диаграмма, снимок экрана, дизайн

Автоматически созданное описание

рис. 1

Создаем аккаунт в Github (рис. 1).

Делаем предварительную конфигурацию git. Открываем терминал и вводим следующие команды, указав свое имя и email (рис. 2).



рис. 2

Настроим utf-8 в выводе сообщений git (рис. 3)



рис. 3

Зададим имя начальной ветки (будем называть её master) (рис. 4)



рис. 4

Параметр autocrlf (рис. 5)



рис. 5

Параметр safecrlf (рис. 6)



рис. 6

**3.2. Создание SSH ключа**

Сгенерируем пару ключей (рис. 7)

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

рис. 7

Далее необходимо загрузить сгенерированный открытый̆ ключ. Зайдём на сайт http://github.org. Скопировав из локальной̆ консоли ключ в буфер обмена (рис. 8)

Изображение выглядит как текст, снимок экрана, Шрифт, информация

Автоматически созданное описание

рис. 8

Вставляем ключ в появившееся на сайте поле и указываем для ключа имя (рис. 9)

Изображение выглядит как текст, Шрифт, снимок экрана, белый

Автоматически созданное описание

рис. 9

* 1. **Создание рабочего пространства и репозитория курса на основе шаблона**

Создадим каталог для предмета «Архитектура компьютера» (рис. 10)



Рис. 10

Задав имя новому репозиторию, переходим в каталог курса (рис. 11)



Рис. 11

Клонируем созданный репозиторий (рис. 12)

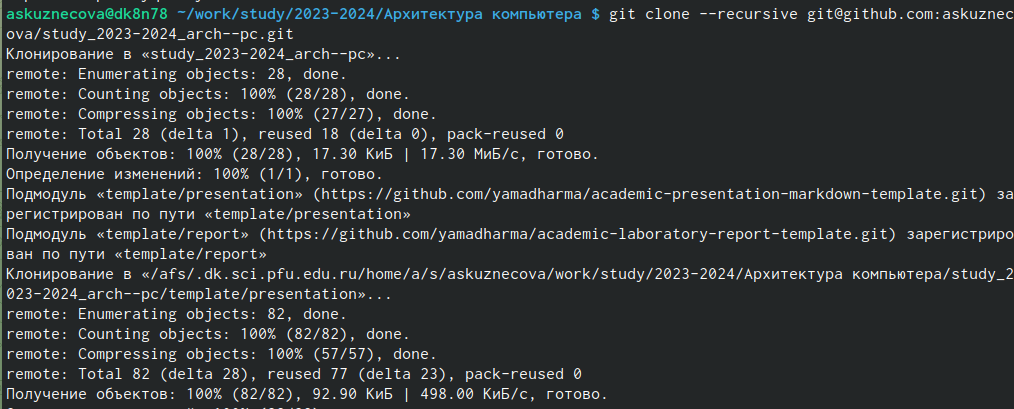


Рис. 12

Перейдём в каталог курса (рис. 13)



Рис. 13

Удалим лишние файлы (рис. 14)



Рис. 14

Создадим необходимые каталоги (рис. 15)



Рис. 15

Запускаем команду «make» (рис. 16)



Рис. 16

Отправляем файлы на сервер (рис. 17)

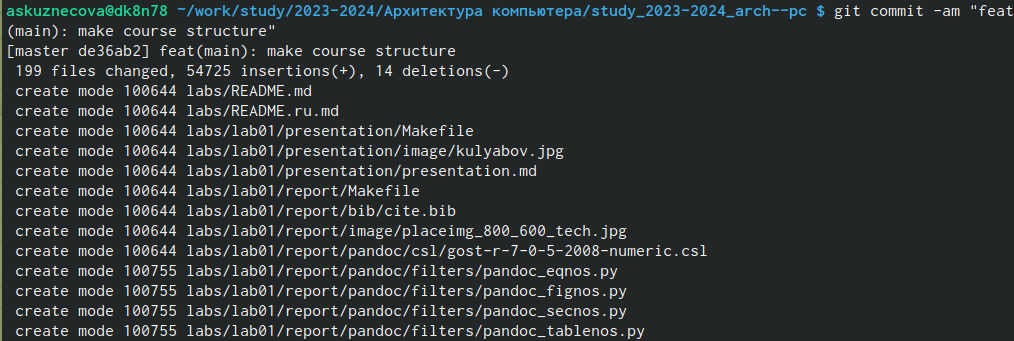


Рис. 17

Проверяем правильность создания иерархии рабочего пространства в локальном репозитории и на странице github (рис. 18)

Изображение выглядит как снимок экрана, текст

Автоматически созданное описание

Рис. 18

1. **Вывод**

Я изучила идеологию и применение средств контроля версий, а также приобрела практические навыки по работе с системой git.