**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ**

**Ордена Трудового Красного Знамени**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский технический университет связи и информатики»**

Отчет по лабораторной работе №1

По дисциплине

«Информационные технологии и программирование»

Основы работы с системой контроля версий

Выполнил: студент группы

БВТ2203

Бородин Кирилл

Москва

2023

Оглавление

[ВВЕДЕНИЕ 3](#_Toc144757651)

[ХОД РАБОТЫ 4](#_Toc144757652)

[ВЫВОД 12](#_Toc144757653)

[ЛИТЕРАТУРА. 12](#_Toc144757654)

# ВВЕДЕНИЕ

Цель работы: изучение базовых функций и подходов к командной разработке в системе контроля версий Git.

**Задание:** определить проект (состоящий из нескольких файлов), который будет добавлен под версионный контроль, и произвести ряд операций в СКВ, включающих:

1. Создание репозитория или клонирование существующего
2. Внесение ряда изменений в файл(ы) и выполнение коммитов (кол-во не меньше 2)
3. Откат изменений до одного из предыдущих коммитов двумя разными способами
4. Создание, переименование и удаление ветки
5. Слияние ветки с веткой мастер с решением конфликта содержания файлов
6. Настройка игнорирования git’ом файлов определенного расширения с использованием glob-шаблонов
7. Вывод лога коммитов в измененном пользовательском формате
8. Настройка удалённого репозитория: получение и отправка изменений в репозиторий
9. Организация совместной работы над проектом: клонирование чужого репозитория, выполнение коммитов, отправка изменений

# ХОД РАБОТЫ

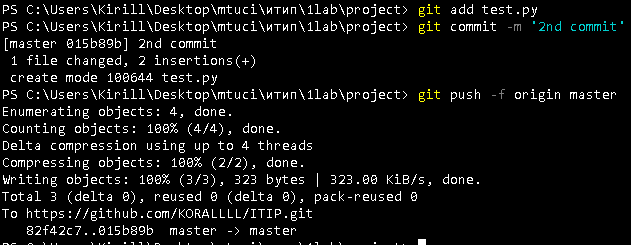
С помощью команды «git init» инициализируем текущий репозиторий, рисунок 1.

  
Рисунок 1 – инициализация репозитория.

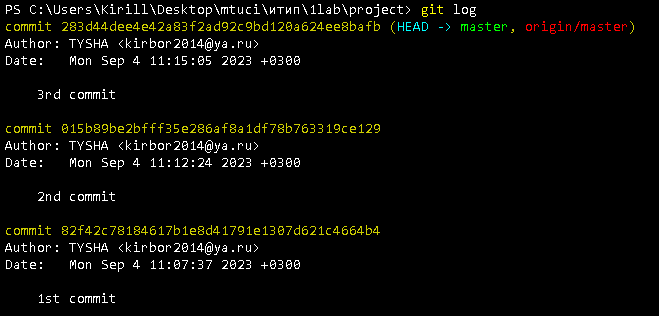
С помощью команды «git remote add <name\_of\_local\_repo> <link\_of\_repository>» связываем локальный и удалённый репозитории, рисунок 2.

  
Рисунок 2 – связывание репозиториев

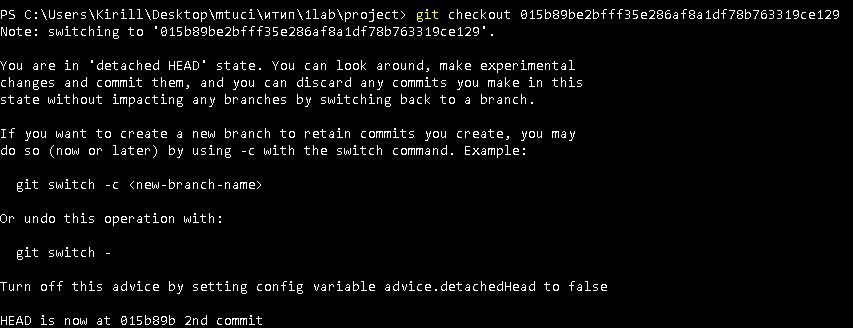
С помощью команды «git add <files>» добавляем файлы в индексирование. С помощью команды «git commit –m ‘name of commit’» создаём коммит. С помощью команды «git push» отправляем изменения в удалённый репозиторий. Флаг «-f» используем, поскольку у репозиторием разные истории коммитов. Далее указываем название локального репозитория и ветку, на которую отправляем изменения, рисунок 3.

  
Рисунок 3 – отправка изменений в удалённый репозиторий.

С помощью команды git log получаем историю коммитов, рисунок 4.

  
Рисунок 4 – история коммитов.

С помощью команды «git checkout» указываем хеш коммита и возвращаемся к нему, рисунок 5.

  
Рисунок 5 – возврат к коммиту.

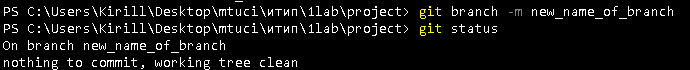
С помощью команды «git reset –hard» можно вернуться к коммиту, при этом все несохранённые данные теряются, изменения после этого коммита игнорируются, рисунок 6.

  
Рисунок 6 – переход к предыдущему коммиту.

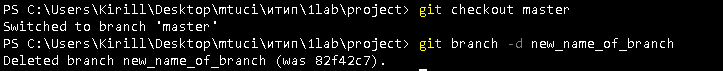
С помощью команды «git checkout» можно также переключаться между ветками, флаг «-b» используется одновременно для создания и переключения на ветку, рисунок 7.

  
Рисунок 7 – создание и переключение на ветку.

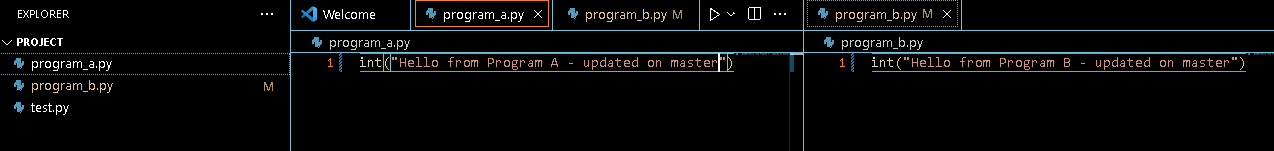
С помощью флага «-m» переименовываю ветку, рисунок 8.

  
Рисунок 8 – переименование ветки.

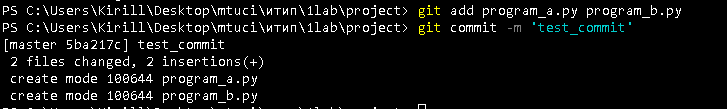
С помощью флага «-d» удаляю ветку, рисунок 9.

  
Рисунок 9 – удаление ветки.

Создаю два различающихся файла, рисунок 10.

  
Рисунок 10 – создание двух различных файлов.

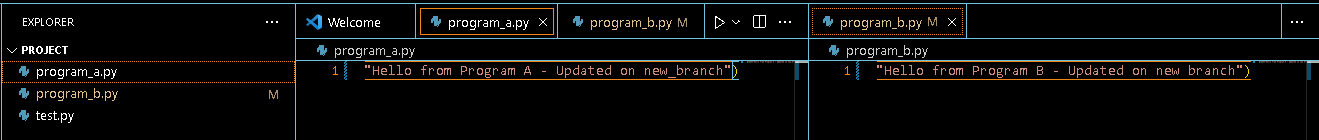
Делаю коммит на текущую ветку, рисунок 11.

  
Рисунок 11 – коммит файлов на текущую ветку.

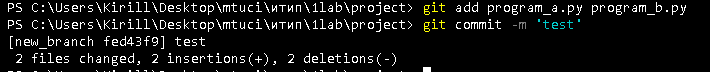
Переключаюсь на другую ветку, рисунок 12.

  
Рисунок 12 – переключение на другую ветку.

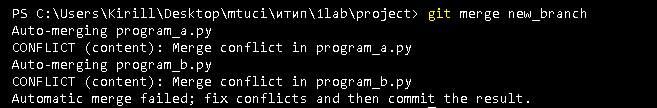
Изменяю файлы, чтобы в дальнейшем был конфликт, рисунок 13.

  
Рисунок 13 – изменение файлов.

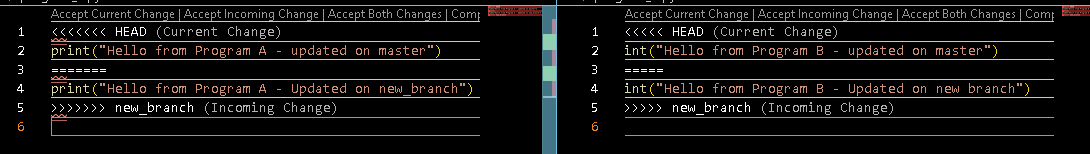
Делаю коммит на этой ветке, рисунок 14.

  
Рисунок 14 – создание коммита на текущей ветке.

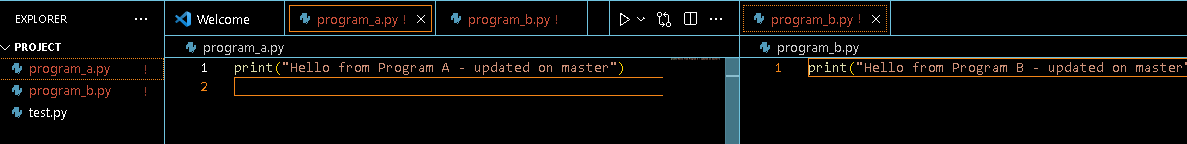
Произвожу слияние двух веток, рисунок 15.

  
Рисунок 15 – слияние двух веток.

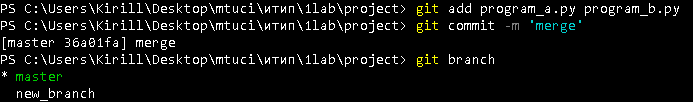
Показаны различия между коммитами, рисунок 16.

  
Рисунок 16 – изменение между коммитами.

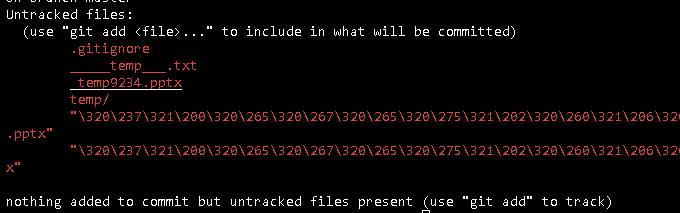
Оставляю только те изменения, что хочу сохранить, рисунок 17.

  
Рисунок 17 – изменение файлов.

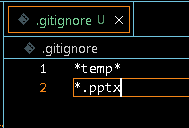
Переключаюсь на основную ветку, с которой буду производить слияние и делаю коммит, рисунок 18.

  
Рисунок 18 – коммит для слияния.

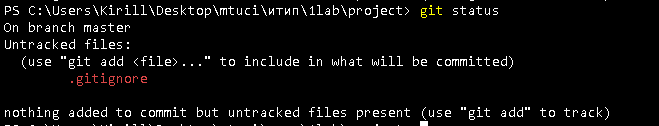
Добавляю бесполезные файлы для работы с файлом «.gitignore», рисунок 19.

  
Рисунок 19 – добавление бесполезных файлов.

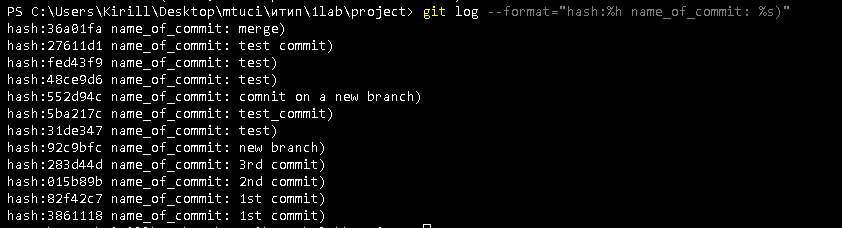
Создаю файл «.gitignore2». С помощью glob-шаблона указываю файлы с игнорируемой частью в названии с игнорируемым расширением, рисунок 20.

  
Рисунок 20 – создание «.gitignore».

После создания файла остальные, не нужные нам компоненты, не отслеживаются, рисунок 21.

  
Рисунок 21 – результат создания «.gitignore».

С помощью команды «git log» вывожу историю коммитов. Флаг «--format» используется для форматирования ввода. «%h» - для вывода обрезанного хеша коммита, «%s» - для отображения названия коммита. Рисунок 21.

  
Рисунок 21 – пользовательский вывод истории коммита.

Добавим на удалённый репозиторий файл для имитации совместной работы, рисунок 22.

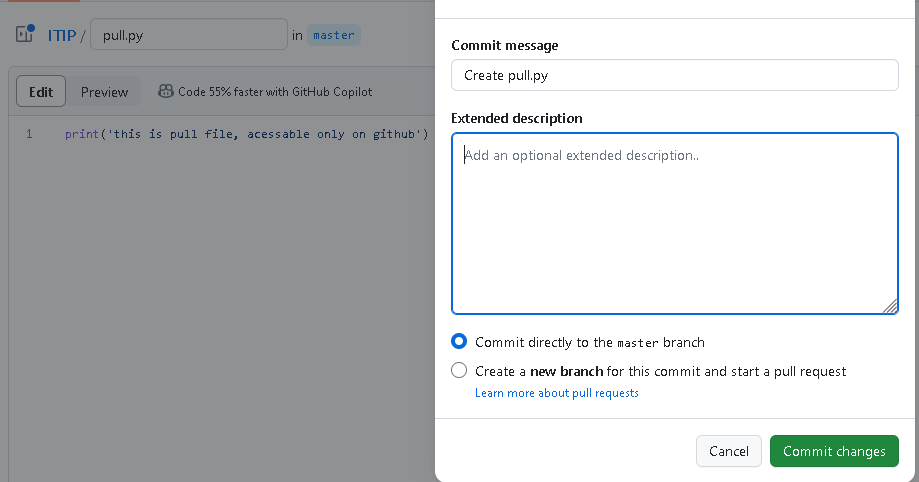
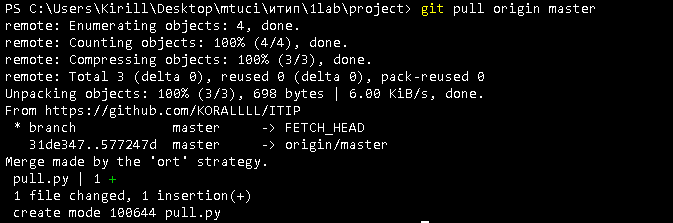
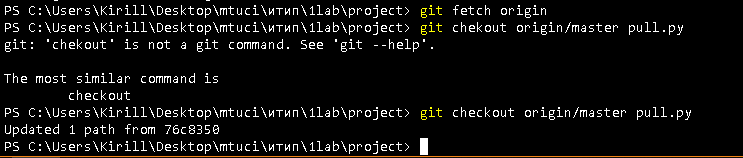


Рисунок 22 – добавление файла в удалённый репозиторий.

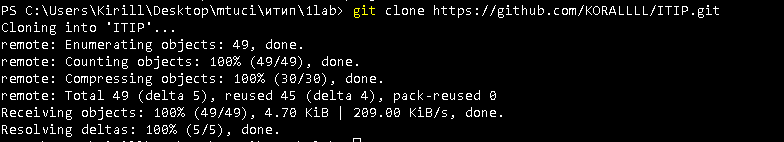
С помощью команды «git pull» получаю изменения из удалённого репозитория, далее указываю название локального репозитория, далее указываю название ветки, с которой я хочу получить изменения, рисунок 23.

  
Рисунок 23 – получение изменений из удалённого репозитория.

С помощью команды «git fetch» получаю изменения из удалённого репозитория, его название на локальном компьютере указываю далее. С помощью команды «git checkout <name\_of\_local\_repository/branch> <name\_of\_file>» получаю тот файл, которого нет у меня на локальном репозитории, рисунок 24.

  
Рисунок 24 – получения файлов из удалённого репозитория.

С помощью команды «git clone <link\_on\_repository» клонирую репозиторий, рисунок 25.

  
Рисунок 25 – клонирование репозитория.

# ВЫВОД

Изучены базовые функции и подходы к командной разработке в системе контроля версий Git. Изучены возможности для: создания репозитория, клонирования репозитория; индексирования файлов; создания коммитов; отката изменений; создания, переименования и удаления веток; слияния веток и разрешения конфилкта содержания фалов; настройка .gitignore файла с использованием glob-шаблонов; вывода истории коммитов в пользовательском формате; получения и отправки изменений в удалёный репозиторий.

# ЛИТЕРАТУРА.

1. A3.3 Приложение C: Команды Git - Основные команды // git URL: https://git-scm.com/book/ru/v2/%D0%9F%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5-C%3A-%D0%9A%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4%D1%8B-Git-%D0%9E%D1%81%D0%BD%D0%BE%D0%B2%D0%BD%D1%8B%D0%B5-%D0%BA%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4%D1%8B
2. Шпаргалка по Git, в которой представлены основные команды // proglib URL: https://proglib.io/p/git-cheatsheet
3. 30 команд Git, необходимых для освоения интерфейса командной строки Git // habr URL: https://habr.com/ru/companies/ruvds/articles/599929/
4. Команды Git // atlassian URL: https://www.atlassian.com/ru/git/glossary