Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет

информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина «Избранные главы информатики»

**ОТЧЕТ**

к лабораторной работе №1

на тему:

**«Система контроля версий Git»**

БГУИР 1-40-04-01

|  |
| --- |
| Выполнил студент группы 253504  Дмитрук Богдан Ярославович |
| 22.11.2023 |
| (дата, подпись студента) |
| Проверила  Жвакина Анна Васильевна |
|  |
| (дата, подпись преподавателя) |

Минск 2023

Оглавление

[1. Цели выполнения задания 3](#_Toc159545324)

[2. Порядок выполнения задания 5](#_Toc159545325)

[3. Вывод: 30](#_Toc159545326)

1. Цели выполнения задания

**Цель**: познакомиться с возможностями и получить практические навыки работы с системой контроля версий Git.

**Задание:**

1. Подготовьте рабочее окружение в соответствии с типом вашей операционной системы.
   1. Установите Git
   2. Выполните базовую настройку
2. Создайте структуру папок (использовать командную строку) и поместите в репозиторий Git на своем локальном компьютере:

(Наименование для базовой папки: 253501\_IVANOV\_12, где 12 – порядковый номер в журнале)

GROUP\_NUMBER\_YOUR\_LAST\_NAME\_ NUMBER\_IN\_LIST

IGI

LR1

Папки и файлы в соответствии с индивидуальным заданием

Отчет по ЛР1 со скринами действий и результатов

LR2

LR3

LR4

LR5

STRWEB

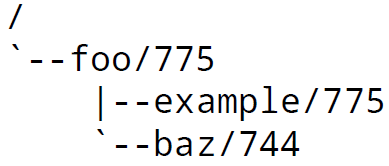
LR1

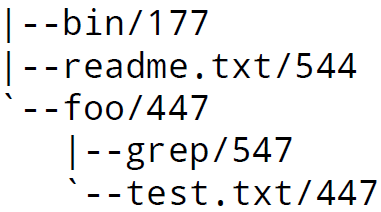
LR2

LR3

LR4

1. Создайте репозиторий внутри папки LR1 с помощью клонирования проекта с сайта github.com (https://github.com/smartiqaorg/geometric\_lib)
2. Выполните следующие действия, каждый раз проверяя состояние файлов в локальном репозитории до и после выполнения операции:
   1. сделайте индексацию нескольких файлов в вашем Git-репозитории (git add …),
   2. внесите изменения в файлы,
   3. выполните коммиты к проиндексированным файлам,
   4. добавьте и удалите файлы (один и группу файлов с определенным расширением)
   5. переименуйте файлы
   6. создайте для IGI и STRWEB, а также для каждой ЛР, вложенной в них, отдельные ветки, переместите папки и файлы на соответствующие ветки
   7. выведите историю коммитов: полностью и с ограничением вывода – отобрать только последние, до и после указанной даты, по автору файлов, коммитов (флаги n, since, after, until, before, author, committer, grep) <https://code.mu/ru/tool/git/basis/limit-output-commit-history-flags/> ,
   8. исследуйте флаги для git log: p , stat ,shortstat, name-only, name-status, relative-date, pretty, graph,
   9. найдите разницу в изменениях для каждого коммита
   10. создайте новый файл test.txt и вставьте в него строчку 'номер варианта'. затем сделайте коммит и отмените его
   11. проиндексируйте файл в вашем Git-репозитории, затем отмените его индексацию
   12. внесите изменения в файл в вашем Git-репозитории и затем верните его в исходное состояние.
3. Выполните следующие действия с удаленным репозиторием:
   1. создайте удаленный репозиторий (на github.com)
   2. просмотрите информацию о нем, внесите изменения и опять просмотрите (remote, fetch).
   3. измените несколько файлов в вашем проекте и перенесите их в удаленный репозиторий,
   4. переименуйте свой удаленный репозиторий
   5. получите данные из удаленного репозитория
   6. просмотреть историю изменений
4. Индивидуальное задание:





1. Порядок выполнения задания
2. Установка и базовая настройка git уже была произведена к моменту начала выполнения лабораторной работы. Порядок действий при установке и базовой настройке git был аналогичен данному в инструкции к выполнению лабораторной работы.
3. Лог выполнения задания проиллюстрирован на рисунке 1 и рисунке 2.

При выполнении задания были использованы команды:

1. mkdir <dirname> – создание новой директории.
2. echo <file content> > <filename> – создание или пересоздание нового текстового файла с размещенеим в нем указанного содержимого.
3. cd <dirname> – переход в существующую директорию.

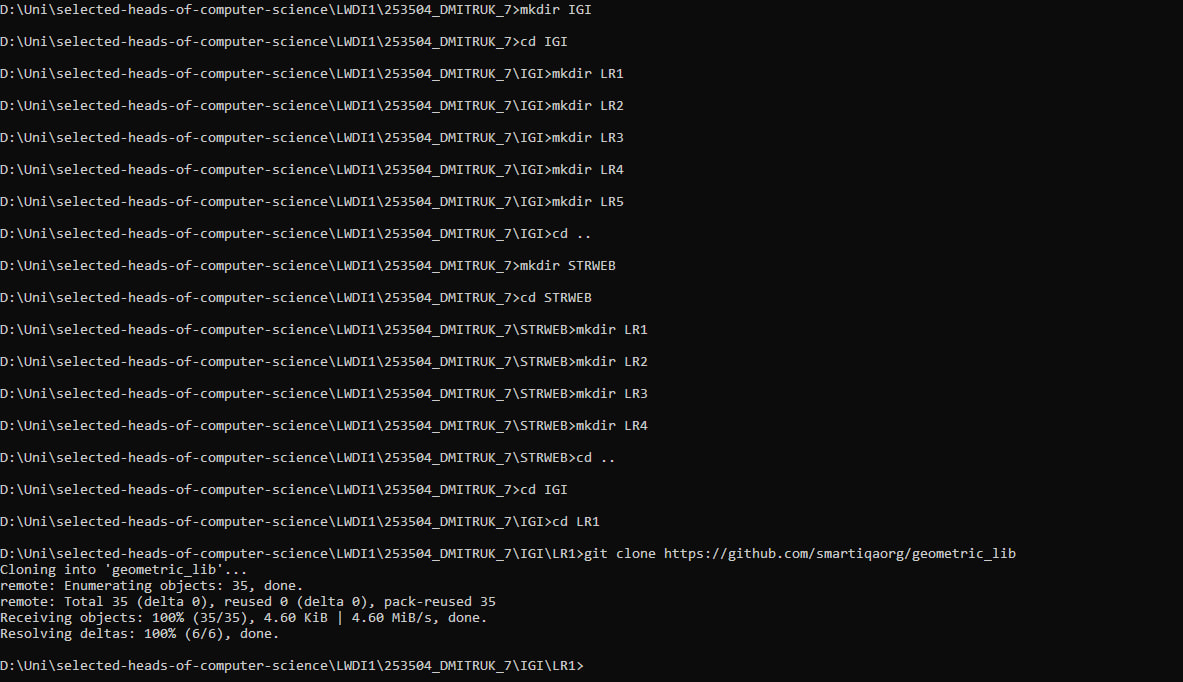


Рисунок 1 – Процесс создания директорий в соответствии с общим заданием

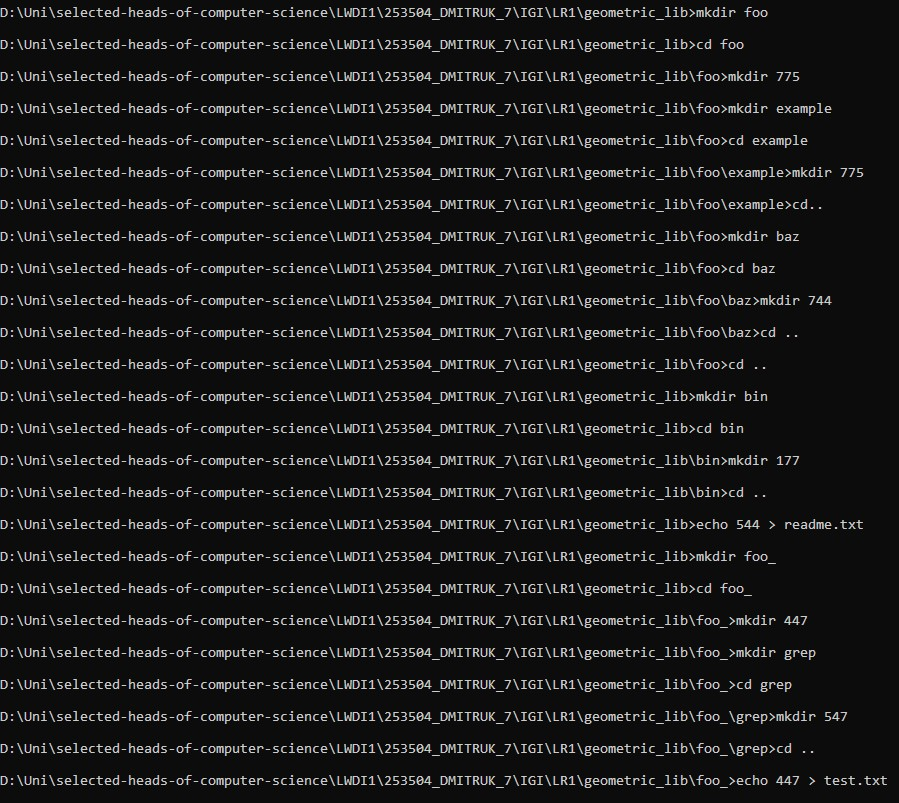


Рисунок 2 – Процесс создания директорий и файлов

в соответствии с индивидуальным заданием

1. Лог выполнения задания проиллюстрирован на рисунках 3, 4, 5.

При выполнении задания были использована команда:

1) git status выводит состояние изменений: не отслеживаются, изменены или подготовлены.

2)git clone <repository URL> – создание локальной копии удаленного директории.

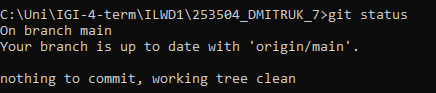


Рисунок 3 – Статус репозитория до выполнения текущего пункта задания

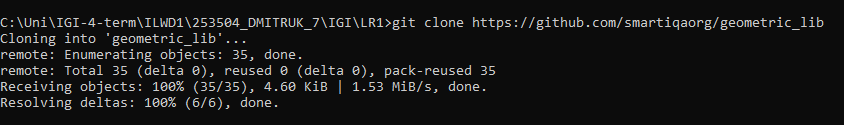


Рисунок 4 – Клонирование удаленного репозитория

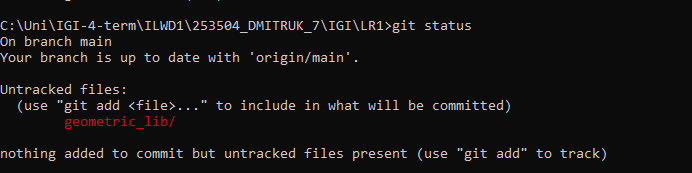


Рисунок 5 – Статус репозитория после выполнения текущего пункта задания

1. Дальнейший процесс выполнения лабораторной работы описан в подпунктах:
2. Лог выполнения задания проиллюстрирован на рисунках 6, 7, 8. Так как я выполнял коммит после каждого пункта задания, чтобы что-то добавить в index/stage\_area, мне сначала необходимо выполнить какие-то изменения в репозитории. Поэтому хронологически иллюстрация к этому пункту появилась после выполнения пункта 4.b.

При выполнении задания были использована команда:

1. git add <filename1> <filename2>… - добавляет файлы в index, переводит их из состояния modified в состояние staged.

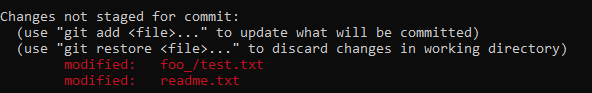


Рисунок 6 – Статус репозитория до выполнения текущего пункта задания

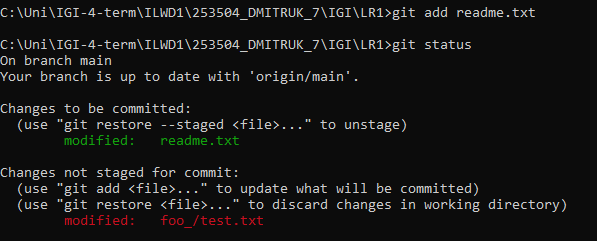


Рисунок 7 – Добавление в индекс конкретного файла

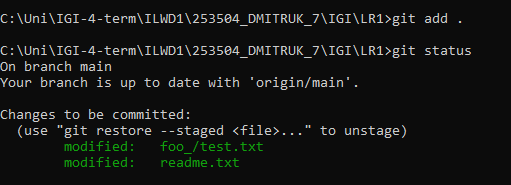


Рисунок 8 – Добавление в index всех unstaged файлов текущей директории и её поддиректорий

1. Лог выполнения задания проиллюстрирован на рисунке 9.

При выполнении задания были использована команда:

1)echo <file content> > <filename> - – создание нового или перезапись существующего текстового файла с размещением в нем указанного содержимого.

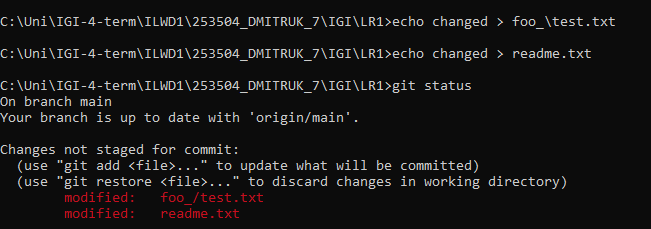


Рисунок 9 – Изменение содержимого файлов. Статус репозитория после выполнения текущего пункта задания

1. Лог выполнения задания проиллюстрирован на рисунке 10.

При выполнении задания были использованы команды:

1. git commit -m”some message” – создание коммита с прикрепленным к нему пользовательским сообщением
2. git log -n <количество> - вывод лога последнего указанного количества коммитов.

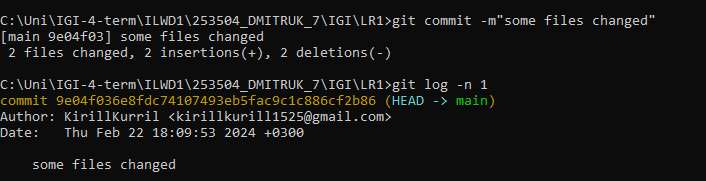


Рисунок 10 – Создание коммита и вывод последнего коммита

1. Лог выполнения задания проиллюстрирован на рисунках 11, 12. Для упрощения выполнения следующего пункта я создал два дополнительных файла: file1.txt и file2.txt.

При выполнении задания были использованы команды:

1. ren <filename1> <filename2> – позволяет переименовать файл.

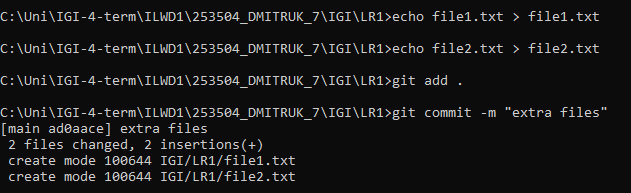


Рисунок 11 – Создание дополнительных файлов

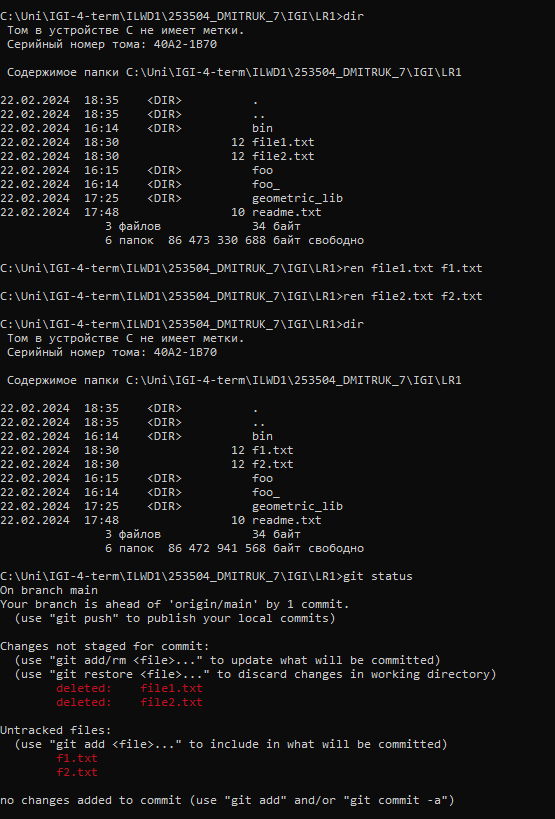


Рисунок 12 – Изменение имен файлов

1. Лог выполнения задания проиллюстрирован на рисунках 13, 14. Так как данное задание является повторяющимся действием, иллюстрация приведена только для действий, проподимых по созданию специальной ветки для IGI/LR1.

При выполнении задания были использованы команды:

1. git checkout -b <branchname> – выполняет переход на ветку branchname. Флаг -b позволяет создать ветку branchname и сразу переключиться на неё.

2)git log – показывает лог коммитов.

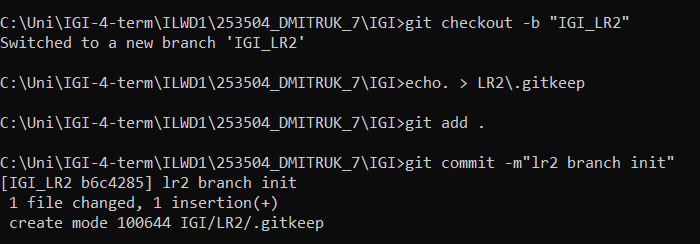


Рисунок 13 – Создание новой ветки и создание на ней нового коммита

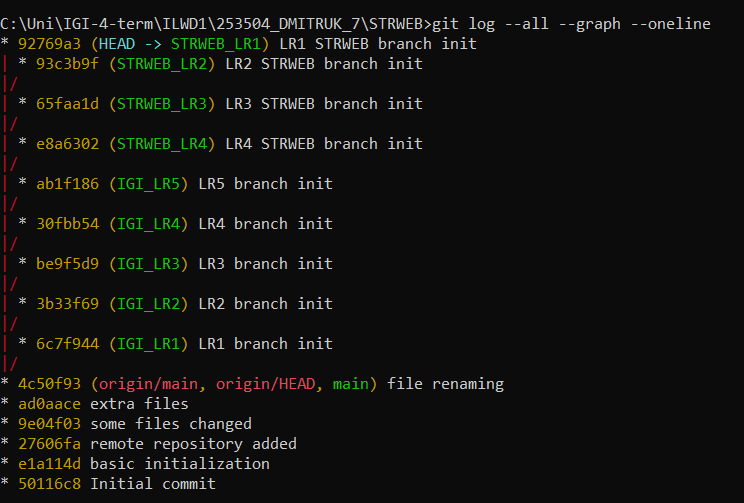


Рисунок 14 – Статус репозитория до выполнения текущего пункта задания

1. Лог выполнения задания проиллюстрирован на рисунках 15, 16, 17, 18, 19, 20, 21, 22.

При выполнении задания были использованы команды:

1. git log -n – показывает последние n коммитов.
2. git log -–since - показывает коммиты, сделанные после указанной даты.
3. git log –-after - показывает коммиты, сделанные после указанной даты.
4. git log -–until - показывает коммиты, сделанные до указанной даты.
5. git log –-before – показывает коммиты, сделанные до указанной даты.
6. git log --author – показывает коммиты, сделанные указанным автором.
7. git log --committer - показывает коммиты, сделанные указанным коммиттером.
8. git log --grep - показывает коммиты, содержащие указанный шаблон в своих сообщениях коммитов.

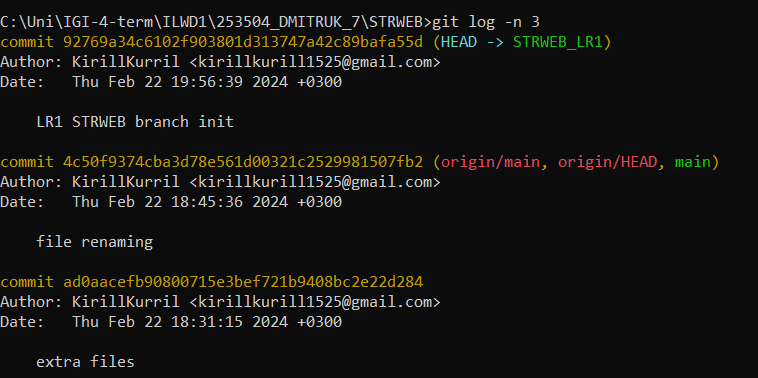


Рисунок 15 – Пример вывода команды git log --n

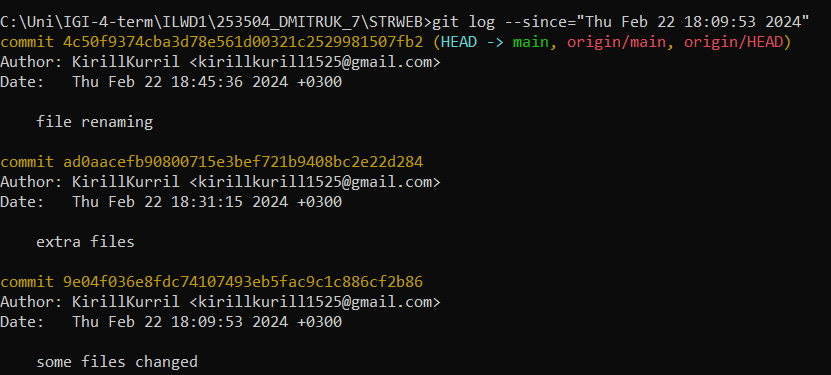


Рисунок 16 – Пример вывода команды git log --since

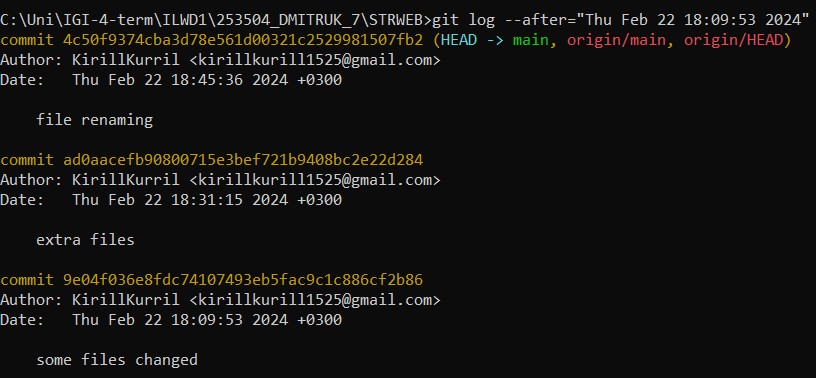


Рисунок 17 – Пример вывода команды git log --after

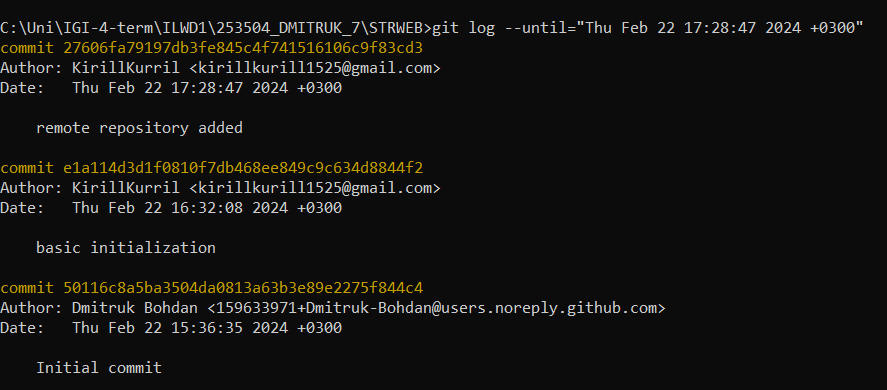


Рисунок 18 – Пример вывода команды git log --until

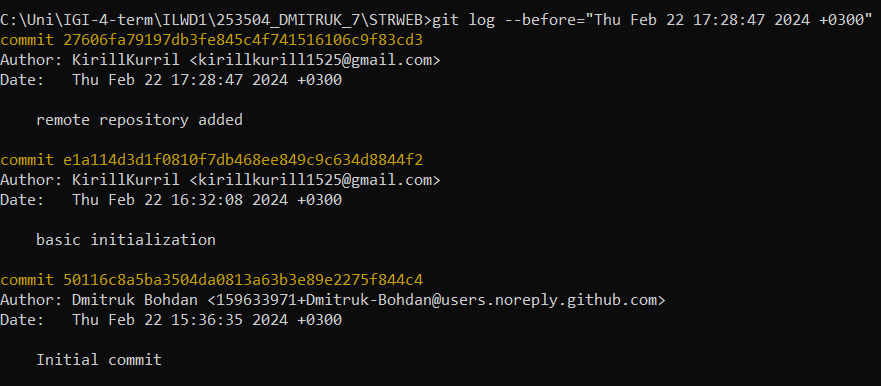


Рисунок 19 – Пример вывода команды git log --before

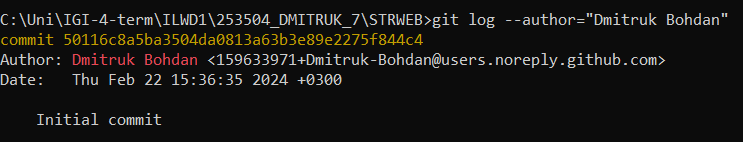


Рисунок 20 – Пример вывода команды git log --author

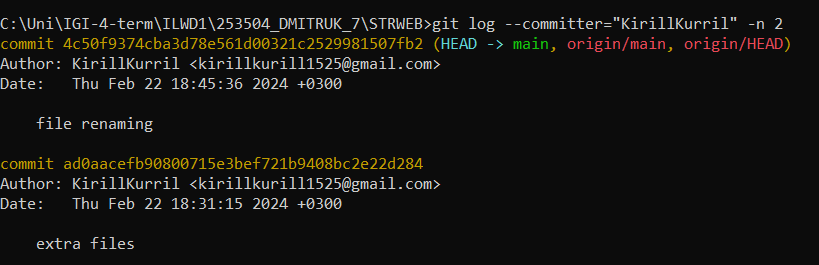


Рисунок 21 – Пример вывода команды git log --committer

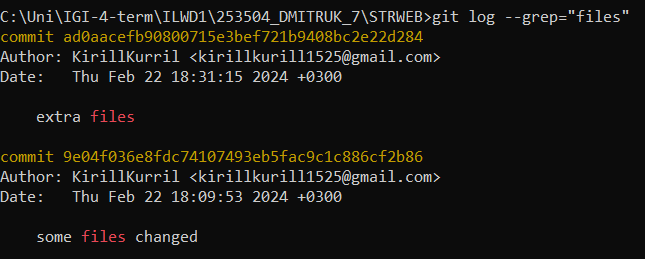


Рисунок 22 – Пример вывода команды git log –files

1. Лог выполнения задания проиллюстрирован на рисунках 23, 24, 25, 26, 27, 28, 29, 30.

При выполнении задания были использованы команды:

1. git log -p - показывает патч для каждого коммита, показывая изменения файлов, внесенные в каждом коммите
2. git log –-stat - показывает статистику изменений для каждого коммита. Выводит список измененных файлов и количество внесенных в них изменений.
3. git log --shortstat – показывает краткую статистику изменений для каждого коммита. Он показывает только общее количество измененных файлов и количество внесенных в них изменений.
4. git log --name-only – показывает только список измененных файлов для каждого коммита, без дополнительной статистики или информации об изменениях.
5. git log --name-status – показывает список измененных файлов для каждого коммита, а также статус изменений (добавлен, изменен или удален) для каждого файла.
6. git log --relative-date – показывает даты коммитов в относительном формате.
7. git log --pretty - позволяет настроить формат вывода лога коммитов. Дает возможность указывать различные шаблоны форматирования.
8. git log --graph – показывает список коммитов в виде графа, используя для этого ASCII символы.

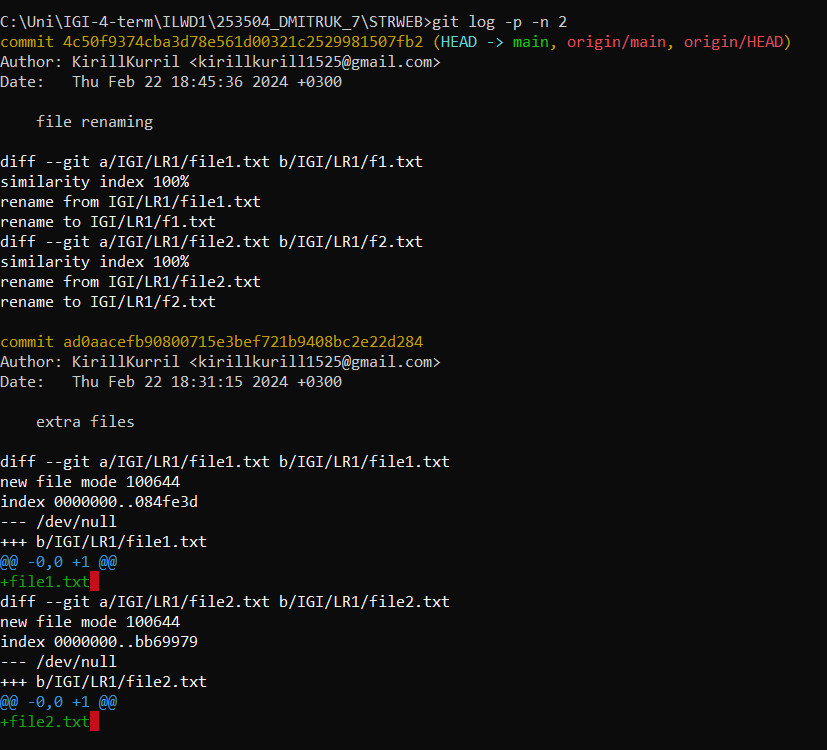


Рисунок 23 – Пример вывода команды git log -p

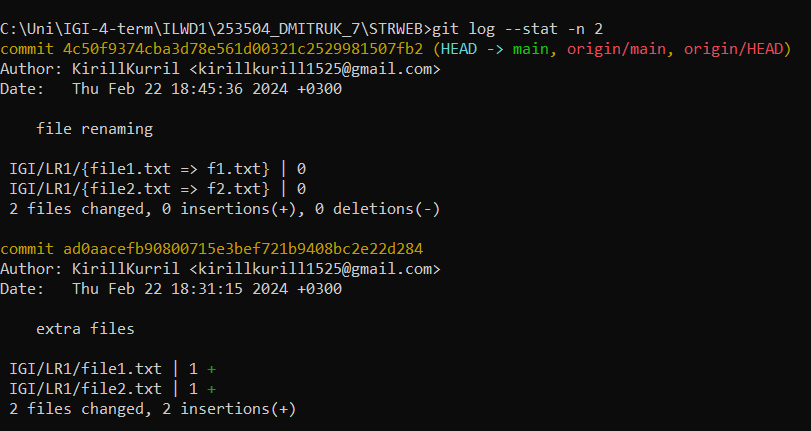


Рисунок 24 – Пример вывода команды git log --stat

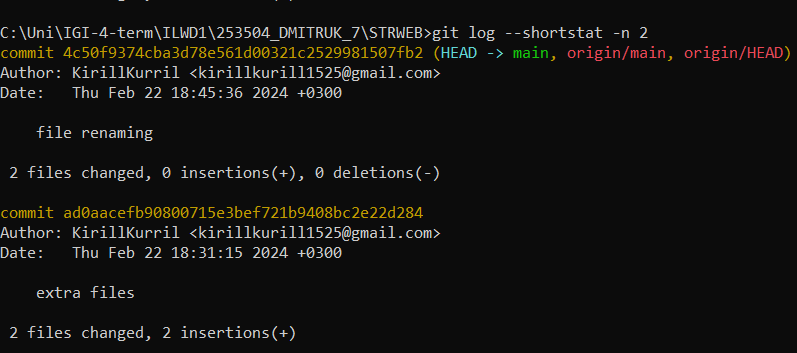


Рисунок 25 – Пример вывода команды git log --shortstat

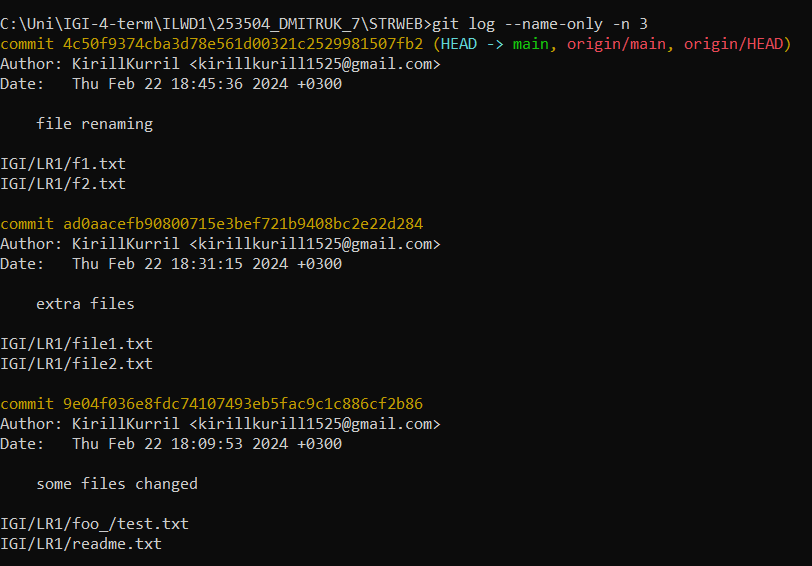


Рисунок 26 – Пример вывода команды git log --name-only

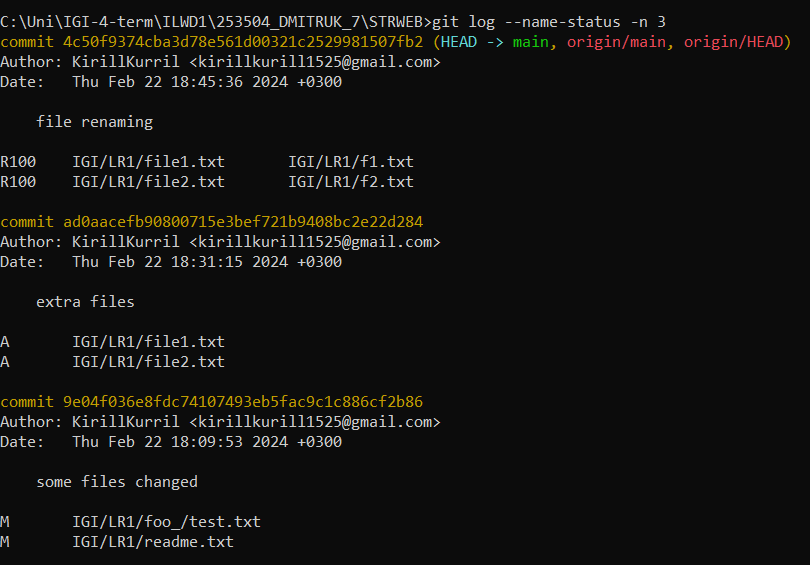


Рисунок 27 – Пример вывода команды git log --name-status

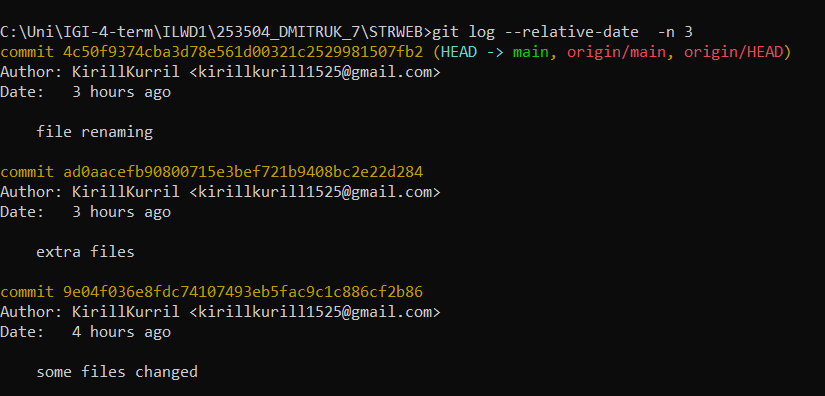


Рисунок 28 – Пример вывода команды git log --relative-date

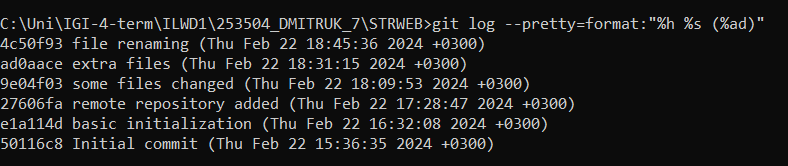


Рисунок 29 – Пример вывода команды git log --pretty

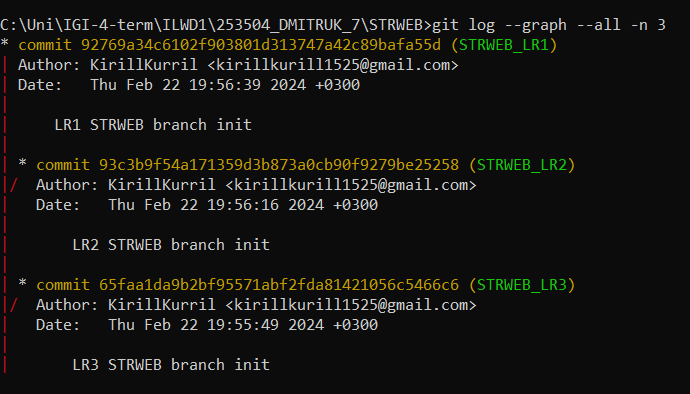


Рисунок 30 – Пример вывода команды git log –all

1. Лог выполнения задания проиллюстрирован на рисунке 31. При выполнении задания были использованы команда git log с использованием флага –stat, охарактеризованная в предыдущем подзадании.

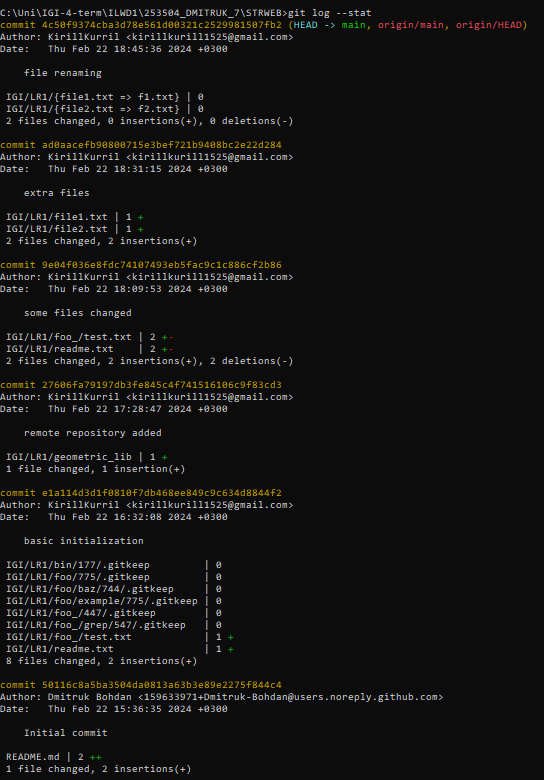


Рисунок 31 – Вывод команды git log -- stat, показывающей разницу в изменениях для каждого коммита

j. Лог выполнения задания проиллюстрирован на рисунках 32, 33.

При выполнении задания были использована команда:

1. git reset –-mixed <commit> - отменяет коммиты и сбрасывает index, но сохраняет изменения в рабочем каталоге.

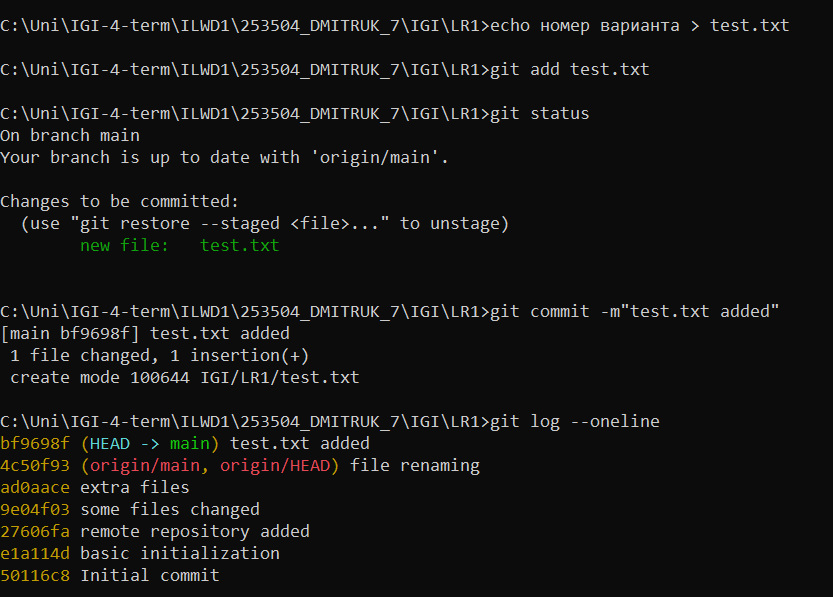


Рисунок 32 – Создание файла test.txt и выполнения коммита с ним

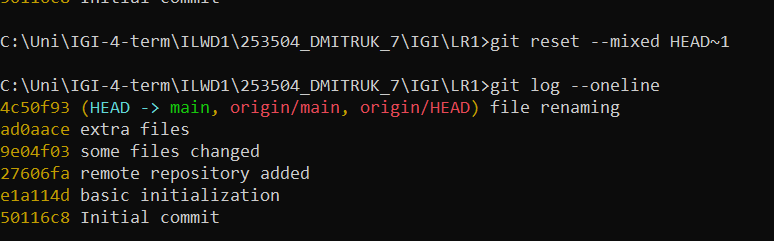


Рисунок 33 – Отмена последнего коммита

k. Лог выполнения задания проиллюстрирован на рисунках 34, 35

При выполнении задания были использована команда:

1. git reset <filename> - убирает указанный файл из индекса, меняет его статус с untracked на staged.

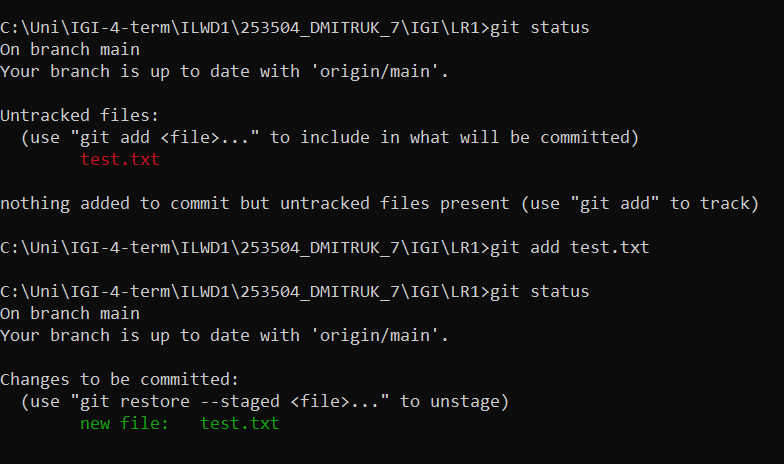


Рисунок 34 – Добавление файла test.txt в index

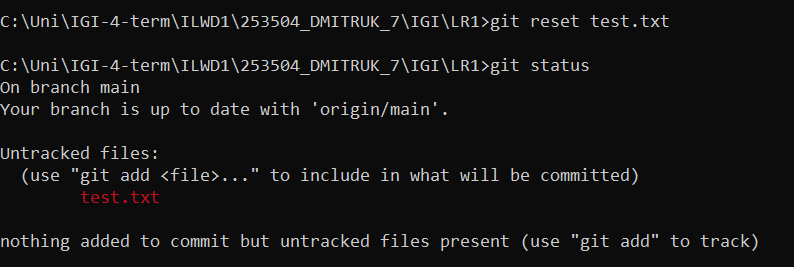


Рисунок 35 – Удаление файла test.txt из индекса

l. Лог выполнения задания проиллюстрирован на рисунках 36, 37, 38.

При выполнении задания были использована команда:

1. git checkout <commit> <filename> - восстанавливает указанный файл до состояния записанного в указанном коммите.

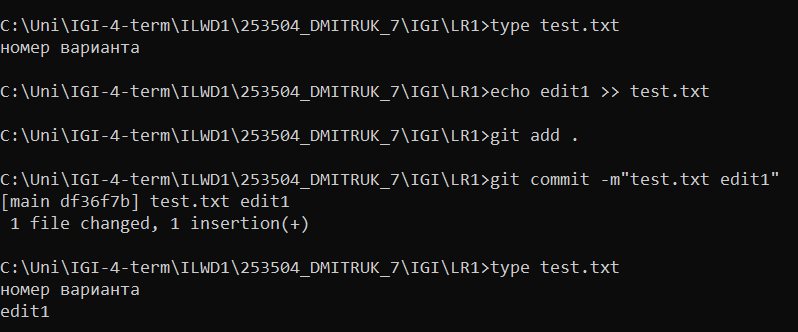


Рисунок 36 – Внесение первого изменения

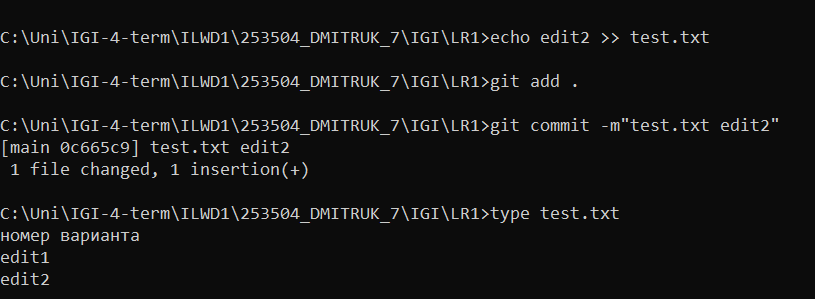


Рисунок 37 – Внесение первого изменения

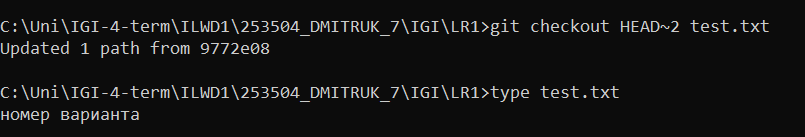


Рисунок 38 – Восстановление файла из предшествующего внесенным изменениям коммита

1. Дальнейший процесс выполнения лабораторной работы описан в подпунктах:
2. Лог выполнения задания проиллюстрирован на рисунке 39.

При выполнении задания были использована команда:

1. git remote -v – показывает список удаленных репозиториев и их URL-адреса.

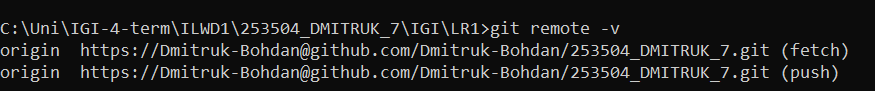


Рисунок 39 – Создан удаленный репозиторий <https://github.com/Dmitruk-Bohdan/253504_DMITRUK_7.git>

1. Лог выполнения задания проиллюстрирован на рисунке 40.

При выполнении задания были использована команда:

1. git fetch - загружает коммиты, файлы и ссылки из удаленного репозитория в локальный репозиторий без выполнения слияния.

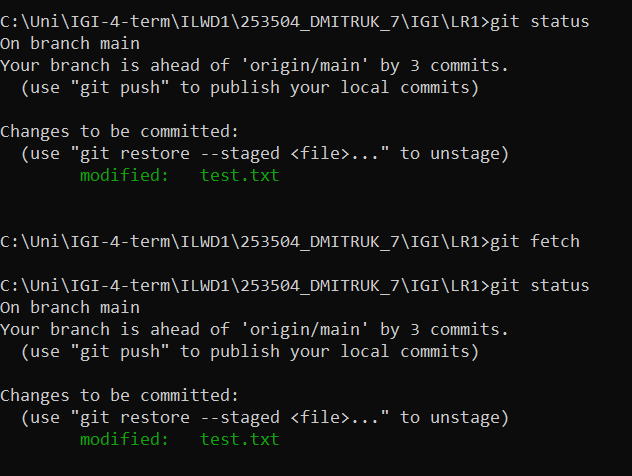


Рисунок 40 – Выполнения команды git fetch

1. Лог выполнения задания проиллюстрирован на рисунке 41.

При выполнении задания были использована команда:

1)git push <repname> <branch> - загружает новые коммиты из указанной локальной ветки в указанный удаленный репозиторий.

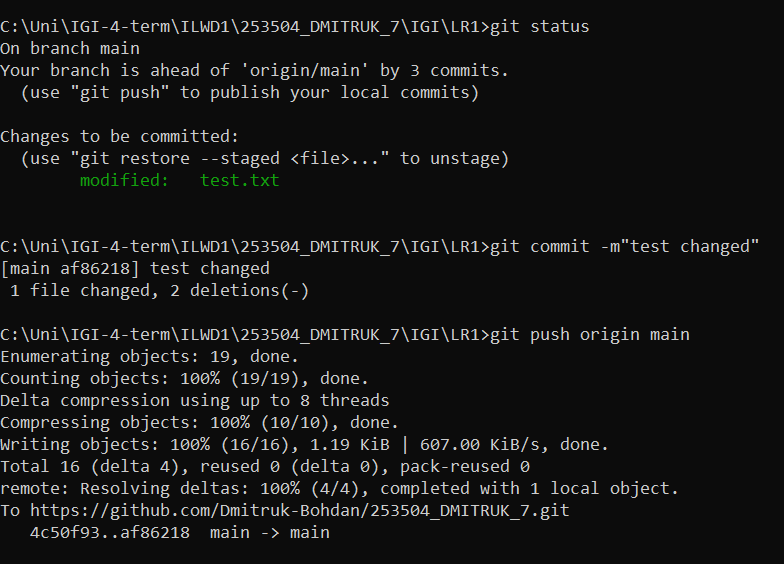


Рисунок 41 – Выполнение переноса изменений в удаленный репозиторий

1. Лог выполнения задания проиллюстрирован на рисунке 42.

При выполнении задания были использована команда:

1)git remote <old-remote-repname> <new-remote-repname> - изменяет название удаленного репозитория.

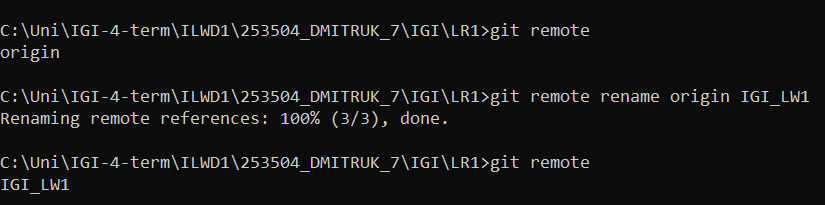


Рисунок 42 – Выполнение изменение названия удаленного репозитория

1. Лог выполнения задания проиллюстрирован на рисунке 43, 44. Для того чтобы на удаленном репозитории появились новые в отношении локального репозитория коммиты, я откатился используя флаг --hard, отменяющий созданные до указанного момента коммиты и отменяющий изменения в директориях устройства вплоть до указанного коммита.

При выполнении задания были использована команда:

1)git pull <remote repname> <remote branchname> - загружает изменения на текущую ветку из указанной удаленной ветки указанного удаленного репозитория.

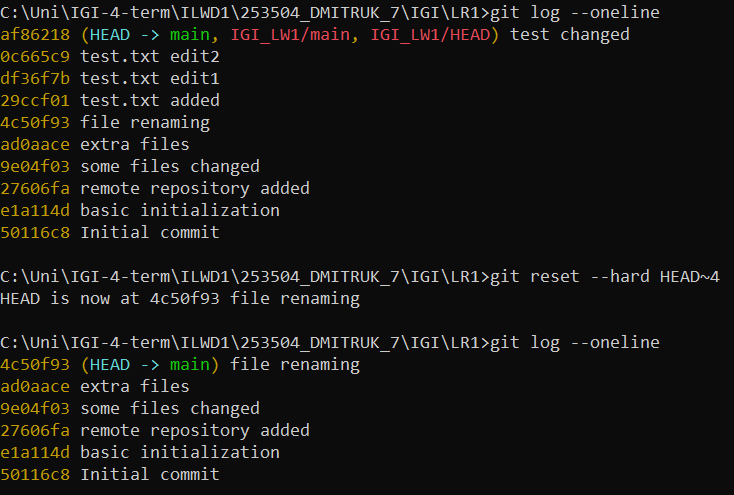


Рисунок 43 – Создание разницы в коммитах с удаленным репозиторием

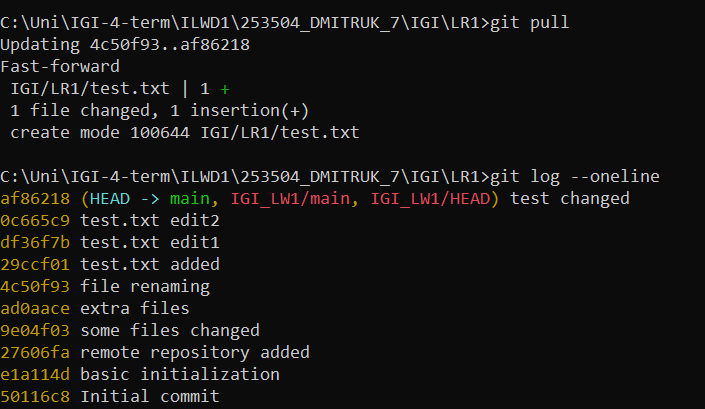


Рисунок 44 – Выполнение команды git pull

1. Лог выполнения задания проиллюстрирован на рисунке 45.

1) git log --all --graph --oneline – показывает историю коммитов --all – всех веток, --graph – в виде графа, --oneline – в качестве информации о коммите выводится только хэш и сообщение от автора.

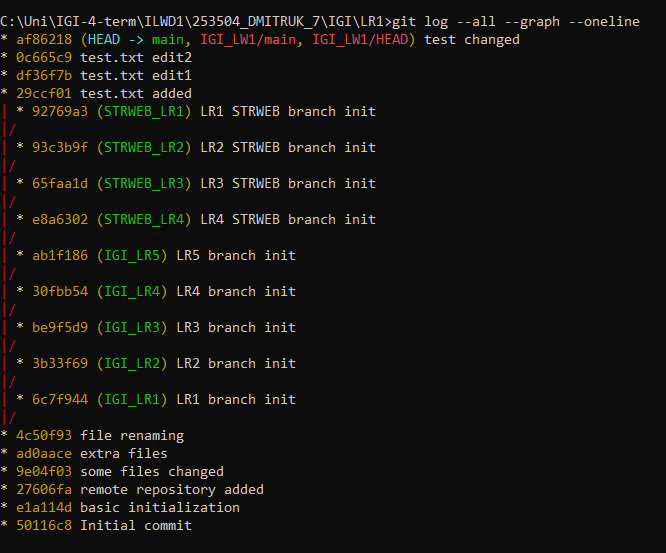


Рисунок 45 – Вывод команды git log --all --graph –oneline

1. Вывод:

В ходе данной лабораторной работы была обзорно изучена система контроля версий Git и получены практические навыки работы с ней в ходе выполнения предписанных заданий.