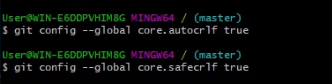
1. Задание

Выполнение отчета по Git.

2. Ход работы

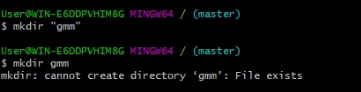
2.1 Подготовка к работе с Git

После установки имени с маилом, я установил параметры окончания строк.

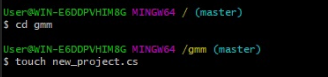


2.2 Создание проекта и репозитория

Я сделал каталог с названием своих инициалов, в котором будут храниться данные об изменениях версий проекта.



После выбора каталога создал файл new\_project.cs.



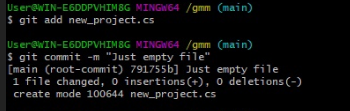


Далее создал репозиторий.





Потом добавил файл в репозиторий и добавил коммит.



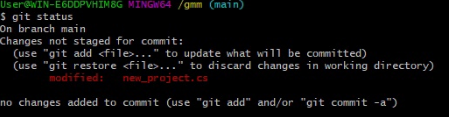
2.3 Проверка состояния репозитория

Проверка состояния репозитория.

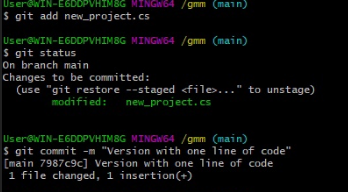


2.4 Индексация изменений

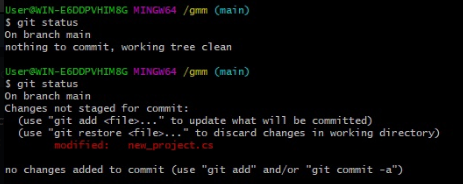
После внесения изменений, проверил состояние репозитория.



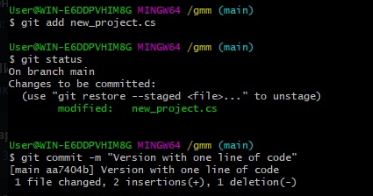
Добавил текущую версию в систему контроля, через индексацию и добавления коммита.



Проверка того, что Git работает с изменениями, а не с целыми файлами. Проверка состояния после и до новой индексации изменений.

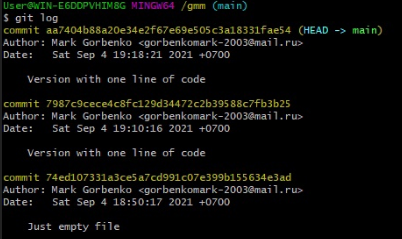


Добавление коммита (назвал так же как и прошлый) к проиндексированной версии.

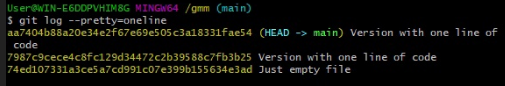


2.5 История версий

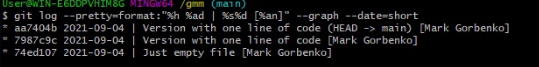
Посмотрел историю версий.



Посмотрел историю версий в режиме «одной строки».

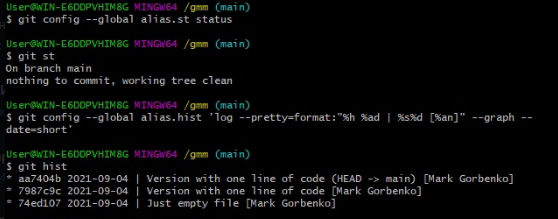


Посмотрел форматированную историю версий.



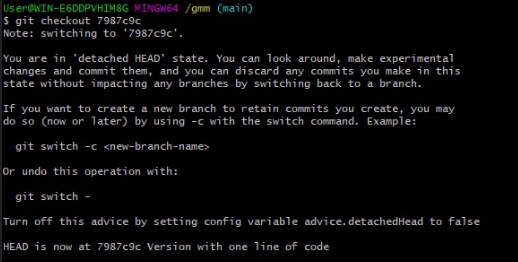
2.6 Алиасы

Использование алиасов.



2.7 Возврат к прошлым версиям

Возврат к прошлым версиям.



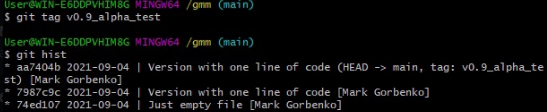
Возврат к последней версии файла (и проверка содержимого).





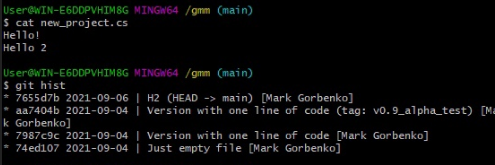
2.8 Тегирование версий

Задал тег и отследил версии.

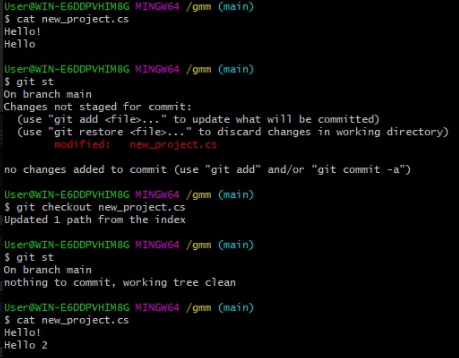


2.9 Отмена локальных изменений

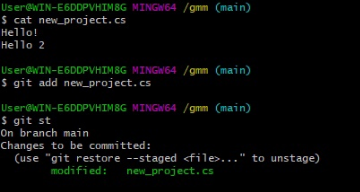
Сделал изменения с индексацией и коммитом в файле.

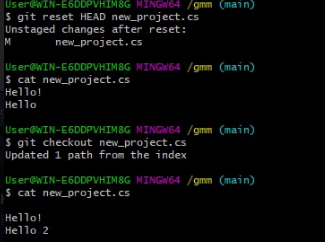


Сделал изменения и отменил их.

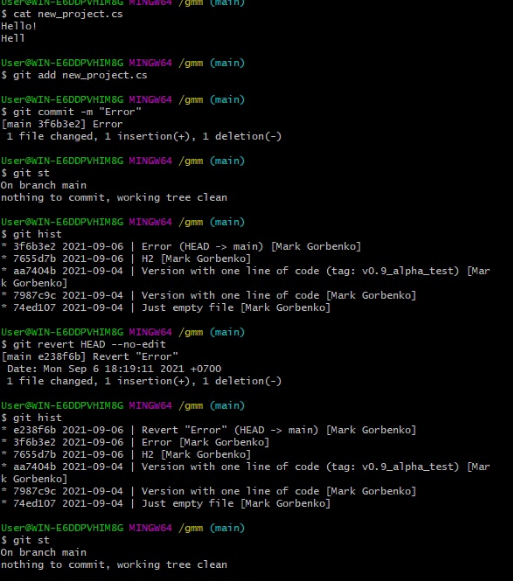


Сделал изменения с индексацией и отменил их.

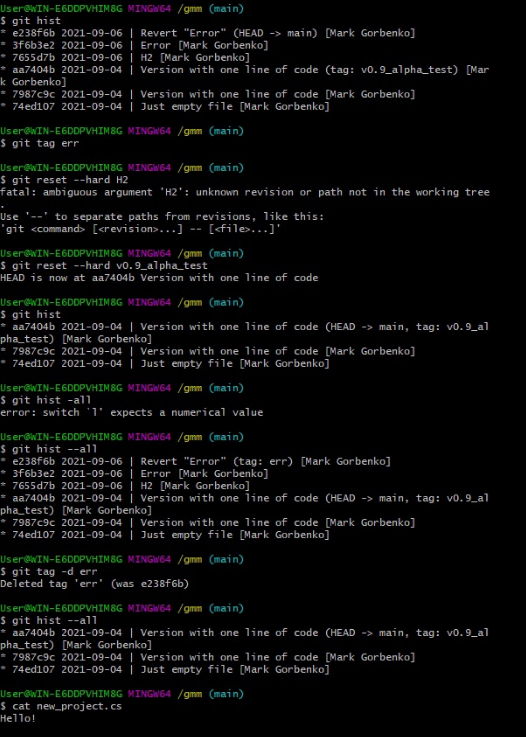




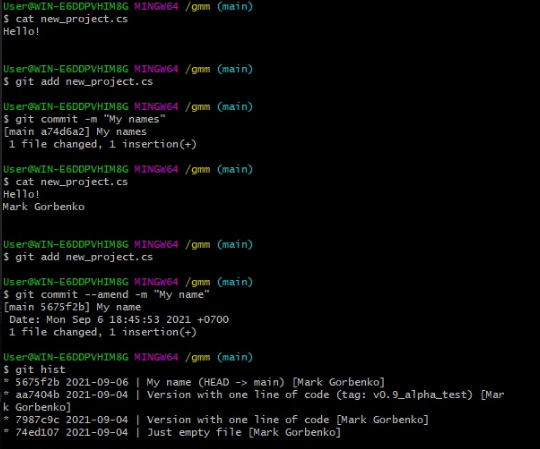
Сделал изменения с индексацией и коммитом и отменил их.



Для удаления истории отмененного коммита протегировал текущую версию файла и выполнил сброс к предыдущей версии. (Последней версией (H2) пришлось пожертвовать, т. к. ее не тегировал.)



Исправление коммита без его отмены и удаления.

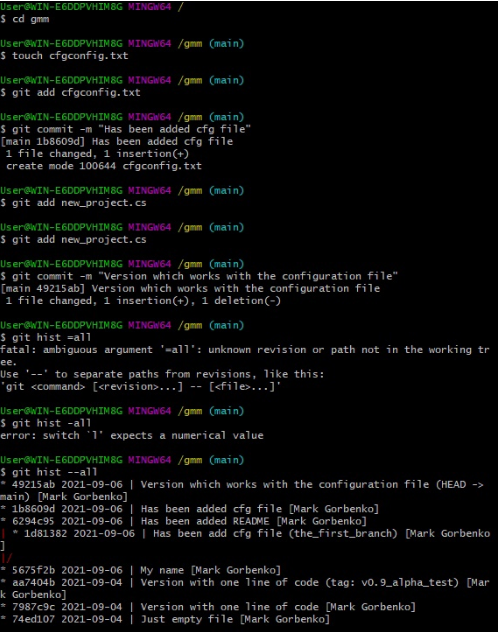


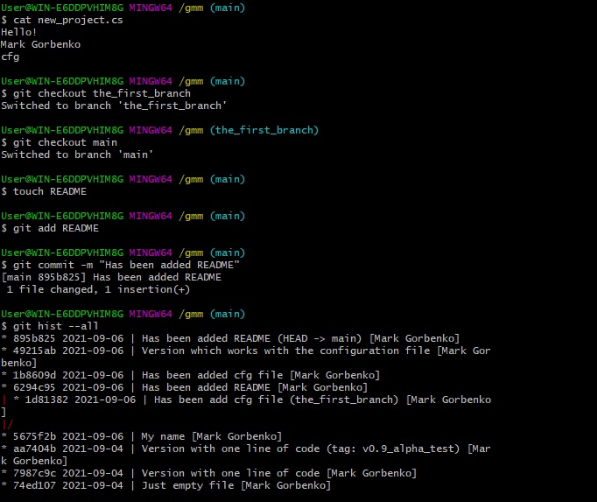
2.10 Создание ветки

Создание ветки the\_first\_branch.



Работа с ветками, предварительно создав файл cfgconfig.txt.



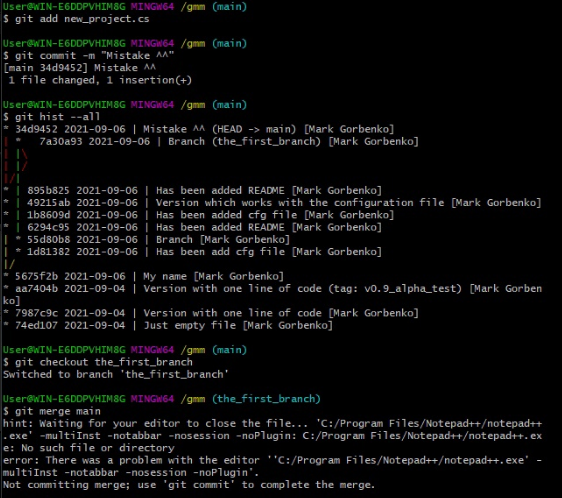


2.11 Слияние веток



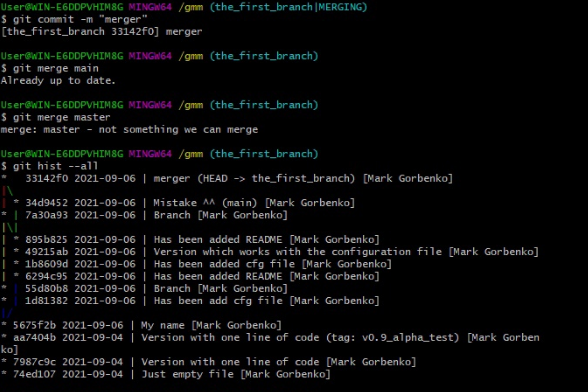
2.12 Конфликт

Создание конфликта веток.



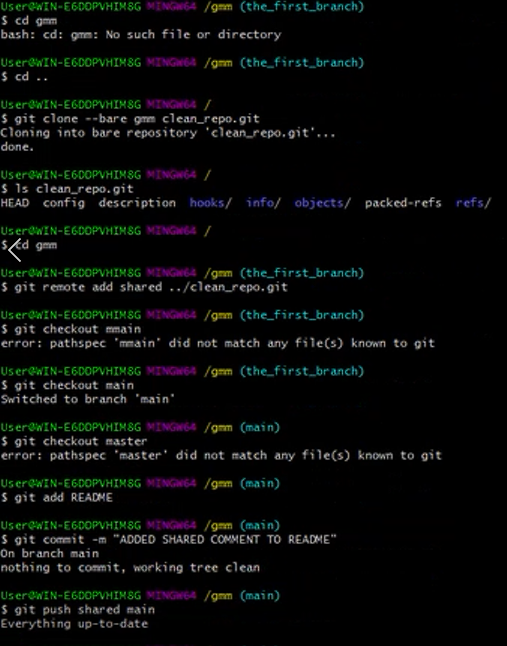
2.13 Разрешение конфликтов

Приведя файл к версии одной из веток и проиндексировав, добавил коммит и решил конфликт веток. (Предварительно создав дополнительную версию)



2.14 Создание чистого репозитория

Создание чистого репозитория clean\_repo. и работа с ним.



Работа с удаленным репозиторием.

Вывод: я выполнил основные виды работ по управлению версиями ПО с использованием системы контроля версий GIT, понял как и что работает, сталкивался с трудностями и разрешал их.

Контрольные вопросы и ответы:

1.Что представляет собой система GIT?

Это ПО, позволяющее отслеживать изменения в документах, при необходимости производить их откат, определять, кто и когда внес исправления и т.п.

2.Перечислите возможности систем управления версиями.

Возможности: позволяет команде работать одновременно над одним проектом, минимизирует конфликты между разработчиками, создает архив каждой версии с изменениями проекта.

3.Что такое коммит и когда он выполняется?

Коммит – операция отправки изменений. Он выполняется после каких-либо изменений проекта.

4.Как посмотреть историю коммитов?

Нужно выполнить команду «git log» или другие схожие команды.

5.Как создать пустой GIT репозиторий?

Нужно создать каталог, где будут храниться данные об изменениях версий проекта, создать там файл, а потом через команду «git init» создать репозиторий.

6.Какую архитектуру имеет система контроля версий GIT?

[Content-addressable storage (CAS) — архитектура хранения, в которой адресация осуществляется образом хранимых данных. Образ данных хешируется и хеш используется для его нахождения на устройствах или системах хранения.](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%82%D0%B5%D0%BD%D1%82%D0%BD%D0%BE-%D0%B0%D0%B4%D1%80%D0%B5%D1%81%D1%83%D0%B5%D0%BC%D0%BE%D0%B5_%D1%85%D1%80%D0%B0%D0%BD%D0%B8%D0%BB%D0%B8%D1%89%D0%B5_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85)

7.Как создать новую ветку в GIT и соединить ветки?

Чтобы создать ветку нужно использовать команду «git checkout –b “название ветки».

Чтобы слить ветки нужно использовать команду «git merge main»

8.Поясните понятие «staging area».

Staging area – область временного хранения копии данных из систем источников.

9.Как сравнить версии файла в GIT?

Нужно использовать команды git diff.