Министерство наук и высшего образования Российской Федерации Пензенский государственный университет Кафедры «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №5

по курсу «ИТвПД»

на тему «СИСТЕМА КОНТРОЛЯ ВЕРСИЙ GIT. ОПЕРАЦИИ ОТМЕНЫ И ИСКЛЮЧЕНИЯ ИЗ КОНТРОЛЯ ФАЙЛОВ»

**Вариант 6**

**Выполнили Студенты группы 24ВВВ3:**

Пяткин Руслан

Гусаров Егор

**Приняли:**

к. т. н. доцент Юрова О. В.

к. т. н. доцент Митрохина Н. Ю.

Пенза 2025

**Цель работы:** научиться работать с веб-сервисом для хостинга проектов и их совместной разработки GitHub. Создать свой публичный репозиторий.

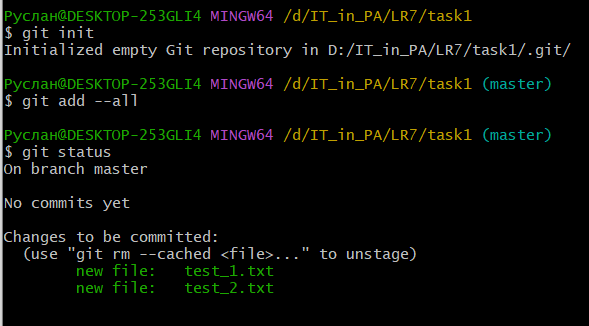
**Задание:**

1. Создали в папке бригады папку LR7 и в ней папку task1.

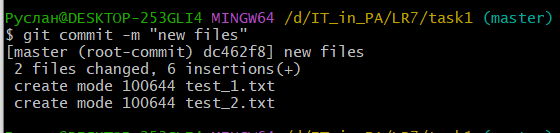
2. Создали внутри папки task1 два файла с расширением .txt и добавили в них текст.

3. Перешли в папку task1, используя команду cd <путь>. Инициализировали репозиторий.

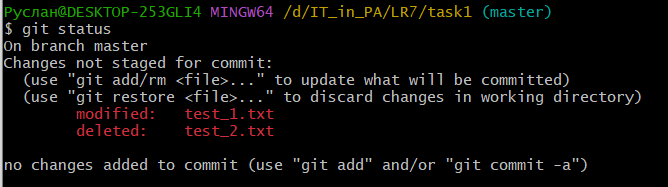
4. Добавили данные файлы в версионный контроль. Убедились в этом.



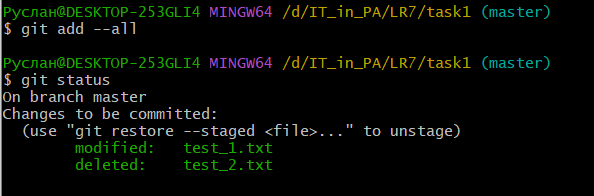
5. Сделайте коммит.



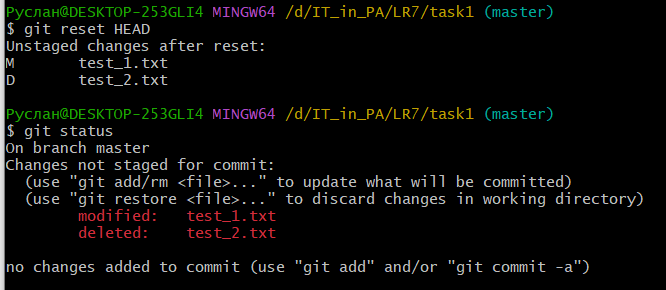
6. Внесите изменения в один из файлов, а другой файл удалите. Выведите текущий статус.



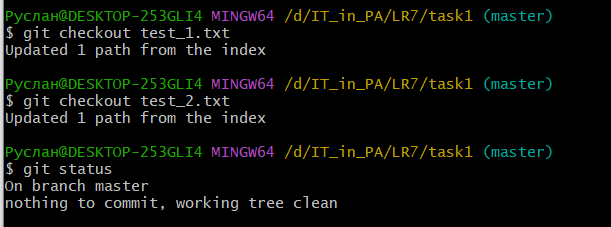
7. Выполните команду $ git add для обоих файлов, чтобы окончательно проиндексировать изменения.



8. Отмените индексирование файлов. Убедитесь, в том, что все вернулось в состояние до индексации, проверив текущий статус.

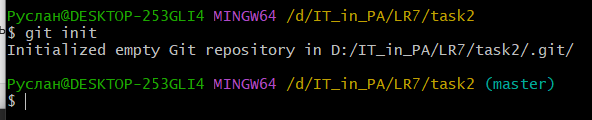
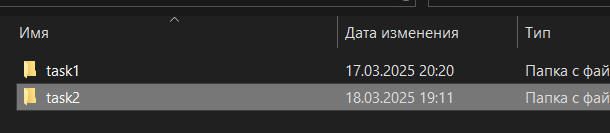


9. Вернитесь в состояние до изменений, используя команду ***$ git checkout***, и убедитесь, что в первом файле отменились изменения, а удалённый файл вернулся на место.

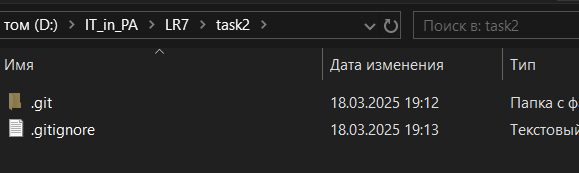


**2. Игнорирование файлов**

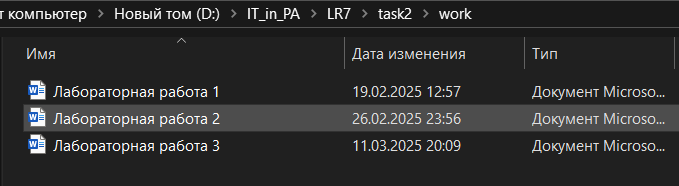
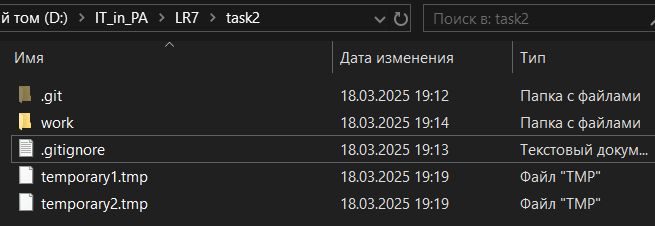
Создали в папке вашей бригады папку LR7 и в ней папку task2 (непосредственно из под Windows, не используя GitBash). Инициализировали репозиторий.



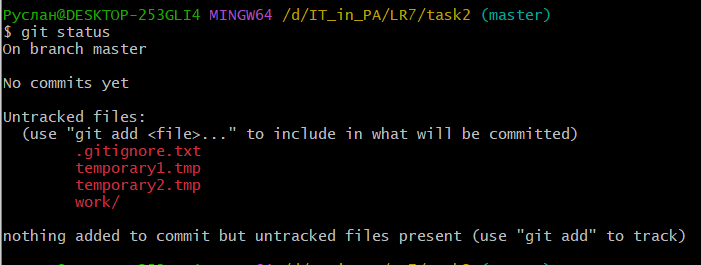
 Создали в этом репозитории файл «.gitignore» для постоянного исключения из числа файлов с отслеживаемыми изменениями временные и автоматически создаваемые файлы компиляторов и компоновщиков.



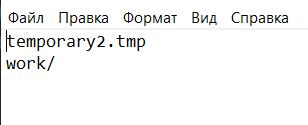
Создали в своем репозитории папку work, в которую добавили любые файлы лабораторных работ, и 2 файла с расширением \*.tmp.



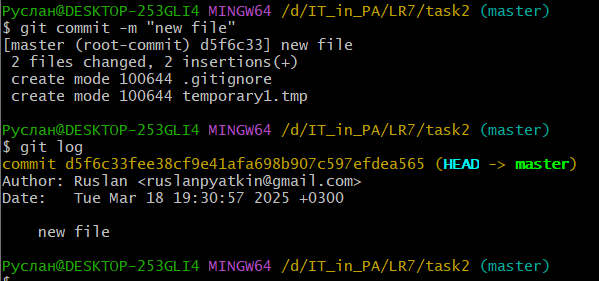
Проверили текущий статус.



Создали в файле «.gitignore» список файлов и папок, отслеживать которые не нужно.



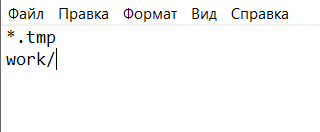
Сделали коммит. Сделали выводы.



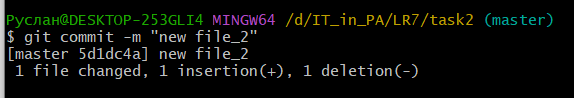
**Вывод:** в этом разделе мы научились использовать файл. gitignore для исключения ненужных файлов из контроля версий. Это особенно полезно для игнорирования временных файлов, логов или результатов сборки. Основной вывод: .gitignore помогает поддерживать чистоту репозитория, избегая добавления файлов, которые не должны отслеживаться Git.

**Задание для самостоятельного выполнения:**

Изменили по своему усмотрению шаблоны игнорирования файлов и папок (измените список шаблонов в файле «.gitignore»).



 Сделали коммит.



**Вывод:** в этом задании мы изучили, как гибко настраивать файл .gitignore для игнорирования определённых файлов и папок в Git. Мы использовали различные шаблоны, такие как:

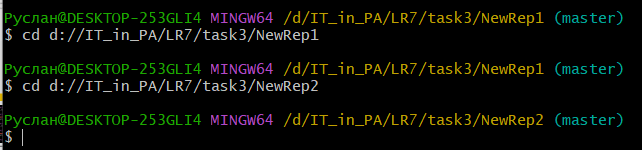
Игнорирование файлов по имени (temporary2.tmp).

Игнорирование всех файлов с определённым расширением (\*.tmp).

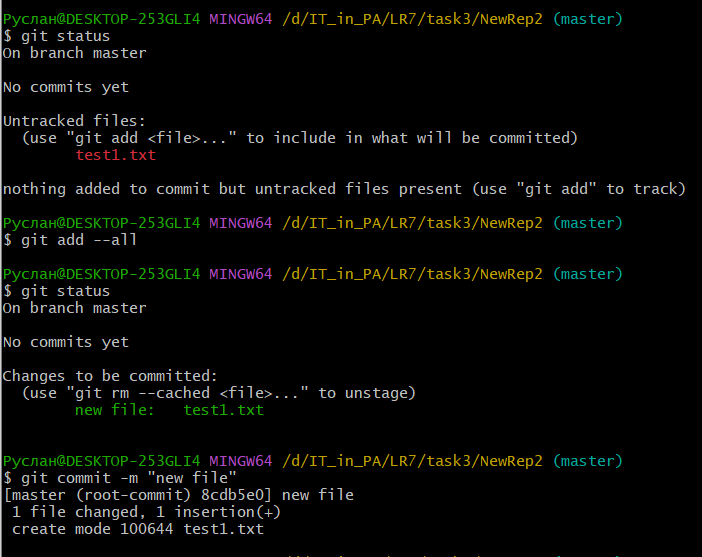
файл .gitignore позволяет эффективно управлять тем, какие файлы и папки Git будет отслеживать, что помогает избежать захламления репозитория ненужными или временными файлами. После настройки .gitignore мы зафиксировали изменения коммитом, что подтвердило правильность работы шаблонов.

**Удалённые репозитории**

Создали в папке LR7 новую папку task3, а внутри этой папки две папки NewRep1 и NewRep2. Инициализировали данные репозитории командой git init.



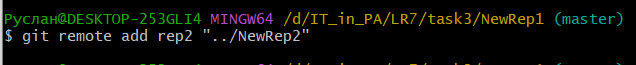
Добавили в репозиторий NewRep2 файл test1.txt с каким-либо текстом, проверили статус, сделали коммит.



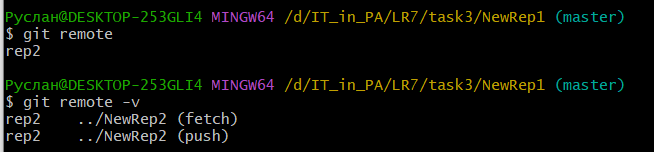
Сделали текущим первый репозиторий.



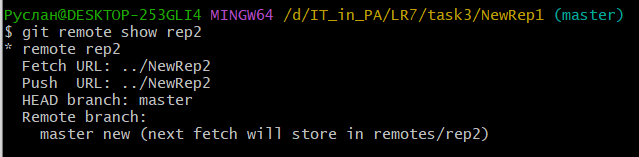
Добавили к текущему репозиторию ссылку на удалённый (находящийся в другом месте) под каким-то именем (назвали его rep2), командой git remote add.



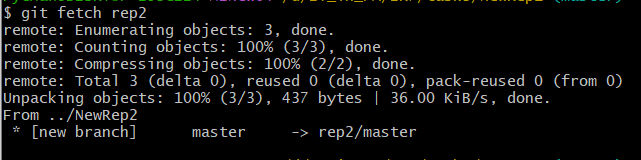
Просмотрим список прикрепленных к репозиторию удалённых репозиториев командой git remote. Ключ –v позволяет выводить не только название, но ещё и адрес удалённого репозитория. В скобках указаны допустимые команды скачивания (fetch) и отправки изменений (push).



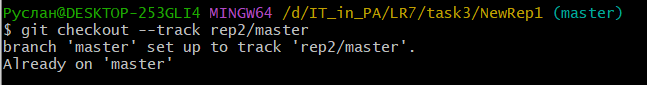
Просмотрим удаленный репозиторий с помощью команды git remote show.



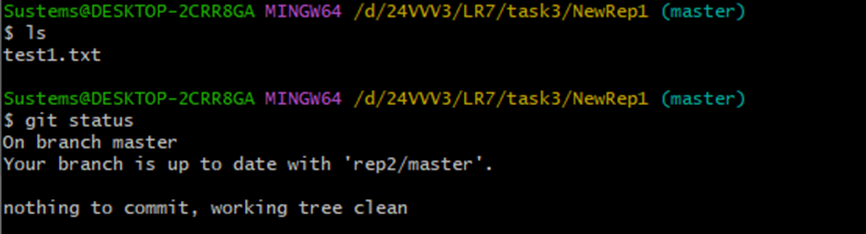
Выполнили команду git fetch rep2. Эта команда возьмет информацию обо всех изменениях в удалённом репозитории и перенесёт в текущий.



Выполнили команду git checkout --track rep2/master. Git должен создать связь текущей ветки данного репозитория с ветвью master репозитория rep2.



Выполнили команды ls и git status.



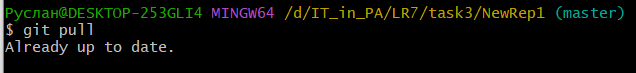
**Вывод:** в этом задании мы изучили работу с удалёнными репозиториями в Git. Мы создали два локальных репозитория (NewRep1 и NewRep2), добавили файл в NewRep2 и закоммитили изменения. Затем связали NewRep1 с NewRep2 как с удалённым репозиторием с помощью команды git remote add. Мы использовали команды:

git fetch для получения информации об изменениях в удалённом репозитории.

git checkout --track для связывания локальной ветки с удалённой.

Git позволяет легко связывать локальные и удалённые репозитории, синхронизировать изменения и работать с ветками. Команды fetch и checkout --track помогают получать данные из удалённого репозитория и настраивать отслеживание веток, что особенно полезно при командной работе.

 Получили изменения командой git pull.



 Удалили ссылку на удаленный репозиторий rep2, используя команду git remote rm rep2.



 Просмотрели список прикрепленных к репозиторию удалённых репозиториев.



**Вывод:** в этом задании мы завершили работу с удалённым репозиторием. С помощью команды git pull мы получили изменения из удалённого репозитория и объединили их с локальной веткой. Затем удалили ссылку на удалённый репозиторий командой git remote rm rep2, что отключило связь между локальным и удалённым репозиторием.