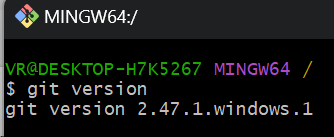
**Лабораторная работа 6**

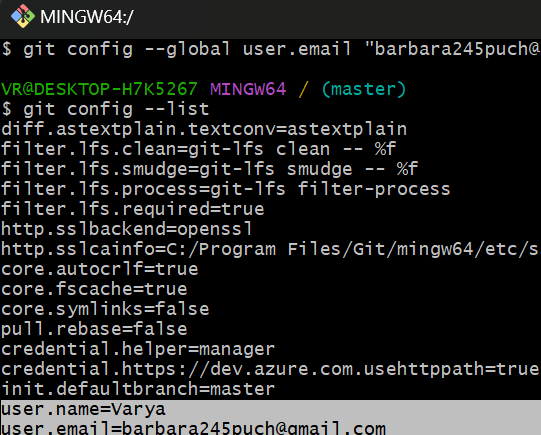
**Системы контроля версий. Основы работы в Git**

4.

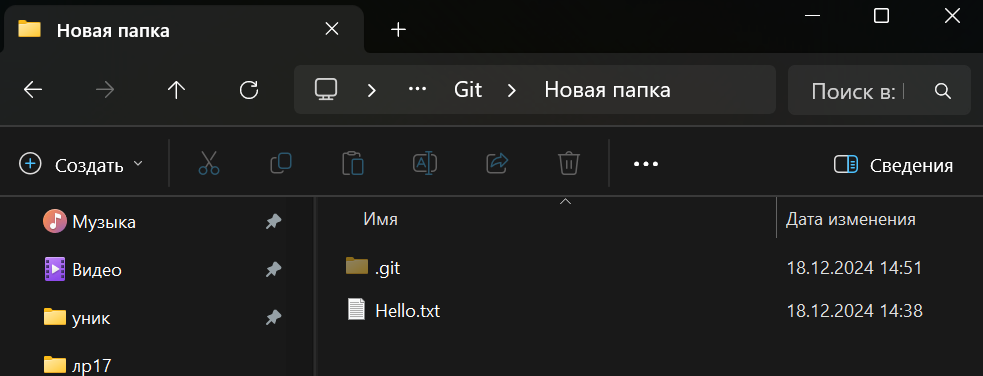


1. Git init

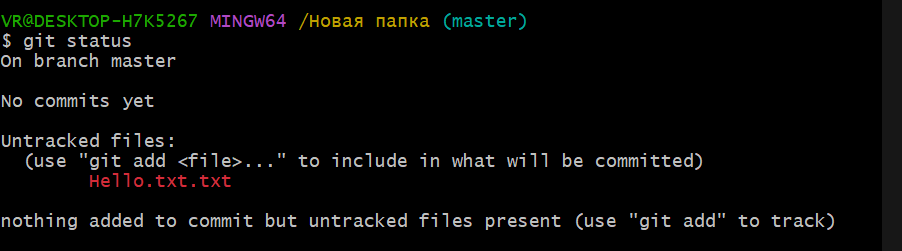
7.



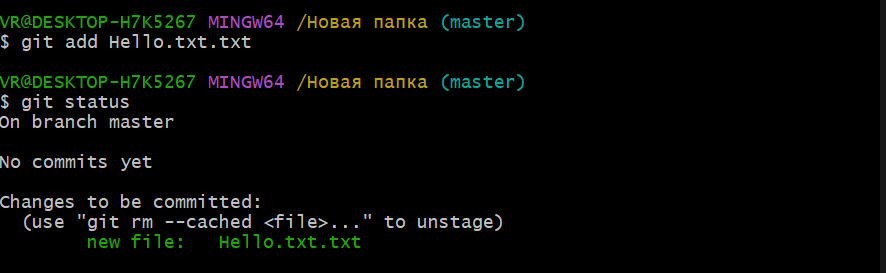
9.



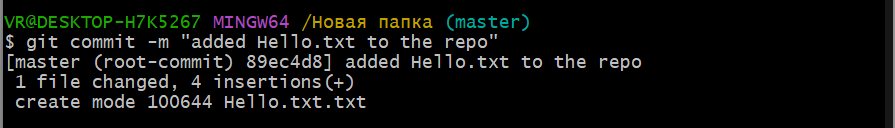
1. информацию о статусе (status) репозитория



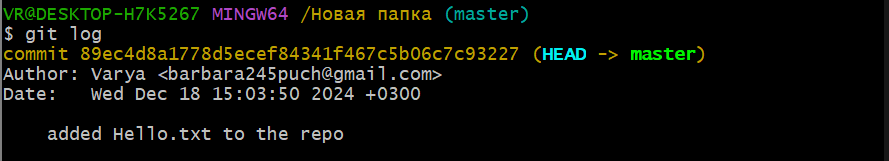
1. Проиндексируйте файл Hello.txt



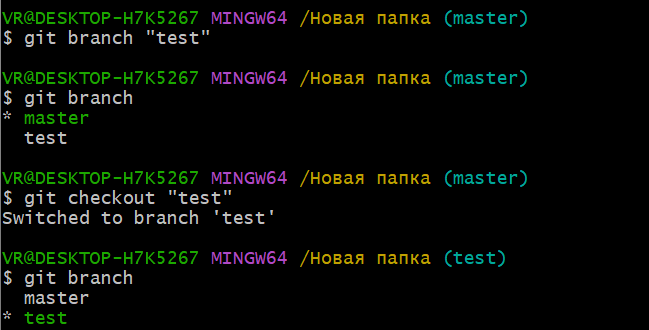
1. Зафиксируйте изменения индексированных файлов в репозитории



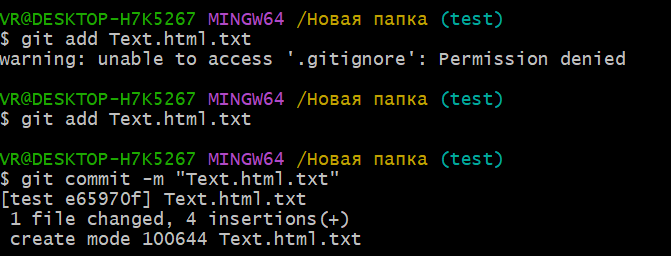
1. журнал с помощью команды git log



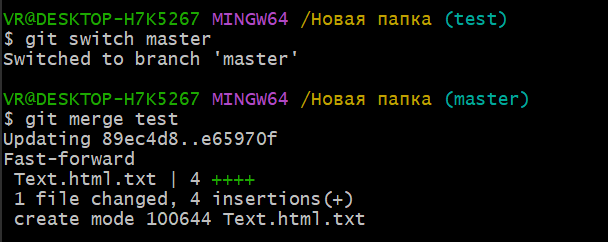
17.Оно поkностью игнорирует папку «текст» благодаря gitignore



19.



20.



▎1. Что такое система контроля версий, для чего ее используют?

Система контроля версий (СКВ) — это инструмент, который помогает отслеживать изменения в файлах и управлять версиями этих файлов. Она используется для:

• Сохранения истории изменений.

• Совместной работы над проектами.

• Возврата к предыдущим версиям файлов.

• Управления изменениями в коде.

▎2. Какой принцип хранения файлов использует Git?

Git использует принцип хранения файлов на основе снимков (snapshot). Каждый раз, когда вы фиксируете изменения, Git создает новый снимок текущего состояния всех файлов в репозитории и сохраняет его. Если файл не изменился, Git просто ссылается на уже существующий снимок.

▎3. В чем отличие Git от других систем контроля версий?

• Распределенность: В отличие от централизованных систем (например, SVN), Git является распределенной системой, где каждый разработчик имеет полную копию репозитория на своем компьютере.

• Эффективность: Git хранит данные более эффективно, используя алгоритмы сжатия и хранение только изменений.

• Гибкость: Git предлагает мощные возможности ветвления и слияния, что упрощает работу с параллельными разработками.

▎4. В каких трех основных состояниях файлы могут находиться в Git?

1. Рабочее состояние (Working Directory): Файлы находятся в рабочем каталоге и могут быть изменены.

2. Индекс (Staging Area): Файлы подготовлены для фиксации, но еще не зафиксированы.

3. Зафиксированное состояние (Committed): Файлы сохранены в репозитории с определенной версией.

▎5. Что такое индексация файла? Какой командой она выполняется?

Индексация файла — это процесс добавления изменений файла в индекс (staging area), чтобы подготовить их к фиксации. Эта команда выполняется с помощью:

git add <имя\_файла>

▎6. Что такое фиксация файла? Какой командой она выполняется?

Фиксация файла — это процесс сохранения изменений в репозитории с созданием новой версии. Эта команда выполняется с помощью:

git commit -m "Описание изменений"

▎7. Продемонстрируйте команду проверки выбранных настроек.

Для проверки настроек используйте команду:

git config --list

▎8. Как инициализировать репозиторий в существующей папке?

Для инициализации репозитория используйте команду:

git init

▎9. Как указать файлы, за которыми должна следить система?

Файлы добавляются в отслеживание с помощью команды:

git add <имя\_файла>

▎10. Как выполнить фиксацию изменений?

Для фиксации изменений используйте команду:

git commit -m "Описание изменений"

▎11. Какой командой определить состояния файлов?

Для определения состояния файлов используйте команду:

git status

▎12. Для чего создается файл .gitignore? Поясните его структуру.

Файл .gitignore используется для указания файлов и папок, которые не должны отслеживаться Git. Структура файла состоит из строк, каждая из которых представляет собой путь или шаблон для игнорируемых файлов. Например:

\*.log # Игнорировать все .log файлы

temp/ # Игнорировать папку temp

▎13. Как используется команда git log?

Команда git log используется для отображения истории коммитов в репозитории. Она показывает список коммитов с их хэшами, авторами, датами и сообщениями.

▎14. Что такое ветвление?

Ветвление — это создание независимых линий разработки в репозитории, позволяющее работать над различными функциями или исправлениями одновременно без влияния на основную кодовую базу.

▎15. Как используется команда git checkout?

Команда git checkout используется для переключения между ветками или для восстановления файлов из определенного коммита. Например, чтобы переключиться на ветку feature, выполните:

git checkout feature

▎16. Что такое указатель HEAD?

Указатель HEAD указывает на текущую ветку или последний коммит в рабочем состоянии. Он показывает, где вы находитесь в истории репозитория.

▎17. Продемонстрируйте создание новых веток в Git?

Для создания новой ветки используйте команду:

git branch <имя\_ветки>

Затем можно переключиться на новую ветку с помощью:

git checkout <имя\_ветки>

Или объединить обе команды:

git checkout -b <имя\_ветки>

▎18. Какая команда осуществляет переключение веток?

Для переключения между ветками используется команда:

git checkout <имя\_ветки>

▎19. Для чего используется команда merge?

Команда merge используется для объединения изменений из одной ветки в другую. Это позволяет слить изменения из одной ветки (например, feature) в основную ветку (например, main).

▎20. Какая команда позволяет отобразить историю репозитория?

Команда git log позволяет отобразить историю коммитов в репозитории.

▎21. Как выполнить слияние изменений из одной ветки в другую?

Чтобы выполнить слияние, сначала переключитесь на целевую ветку (например, main), а затем выполните команду:

git merge <имя\_ветки>

▎22. Какие проблемы могут быть при слиянии и как они разрешаются?

При слиянии могут возникнуть конфликты, если изменения в одной ветке противоречат изменениям в другой. Конфликты разрешаются вручную:

1. Git пометит конфликтующие файлы.

2. Откройте конфликтующие файлы и вручную исправьте конфликты.

3. После разрешения конфликтов добавьте файлы в индекс и выполните фиксацию.

Пример команд для разрешения конфликта:

git add <имя\_файла>

git commit -m "Разрешение конфликта"

Если у вас есть дополнительные вопросы или вам нужно больше информации по какой-либо теме, дайте знать!