Министерство науки и высшего образования Российской Федерации Федеральное государственное автономное образовательное учреждение высшего образования

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт перспективной инженерии

Департамент цифровых, робототехнических систем и электроники

**ОТЧЕТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ №2 дисциплины**

**«Основы кросcплатформенного программирования» Вариант**

Выполнил:

Кравчук Мирослав Витальевич

2 курс, группа ИТС-б-о-23-1, 11.03.02 «Инфокоммуникационные технологии и системы связи», очная форма обучения

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Проверил:

Доцент департамента цифровых, робототехнических систем и

электроники Воронкин Р.А.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Отчет защищен с оценкой \_\_\_\_\_\_\_\_\_\_\_ Дата защиты\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ставрополь, 2024 г.

**Тема: Исследование возможностей Git для работы с локальными репозиториями**

**Цель работы:** исследовать базовые возможности системы контроля версий Git для работы с локальными репозиториями.

**Конспект теоретического материала:**

Полезные опции для git log --pretty=format отображает наиболее полезные опции для изменения формата. Опция

Описания вывода %H

Хеш коммита

%h

Сокращенный хеш коммита %T

Хеш дерева

%t

Сокращенный хеш дерева %P

Хеш родителей

%p

Сокращенный хеш родителей

%an

Имя автора

%ae

Электронная почта автора %ad

Дата автора (формат даты можно задать опцией --date=option) %ar

Относительная дата автора

%cn

Имя коммитера

%ce

Электронная почта коммитера

%cd

Дата коммитера

%cr

Относительная дата коммитера

%s

Содержание

Вам, наверное, интересно, какая же разница между автором и коммитером. Автор — это человек, изначально сделавший работу, а коммитер — это человек, который последним применил эту работу. Другими словами, если вы создадите патч для какого-то проекта, а один из основных членов команды этого проекта применит этот патч, вы оба получите статус участника — вы как автор и основной член команды как коммитер.

Опции oneline и format являются особенно полезными с опцией --graph команды log . С этой опцией вы сможете увидеть небольшой граф в формате ASCII, который показывает текущую ветку и историю слияний.

Для того, чтобы внести вклад в какой-либо Git-проект, вам необходимо уметь работать с удалёнными репозиториями. Удалённые репозитории представляют собой версии вашего проекта, сохранённые в интернете или ещё где-то в сети. У вас может быть несколько удалённых репозиториев, каждый из которых может быть доступен для чтения или для чтения-записи.

Взаимодействие с другими пользователями предполагает управление удалёнными репозиториями, а также отправку и получение данных из них. Управление репозиториями включает в себя как умение добавлять новые, так и умение удалять устаревшие репозитории, а также умение управлять различными удалёнными ветками, объявлять их отслеживаемыми или

нет и так далее. В данном разделе мы рассмотрим некоторые из этих

навыков.

Для того, чтобы просмотреть список настроенных удалённых репозиториев, вы можете запустить команду git remote. Она выведет названия доступных удалённых репозиториев. Если вы клонировали репозиторий, то увидите, как минимум origin — имя по умолчанию, которое Git даёт серверу, с которого производилось клонирование.

Как и большинство СКВ, Git имеет возможность помечать определённые моменты в истории как важные. Как правило, эта функциональность используется для отметки моментов выпуска версий (v1.0, и т. п.). Такие пометки в Git называются тегами. В этом разделе вы узнаете, как посмотреть имеющиеся теги, как создать новые или удалить существующие, а также какие типы тегов существуют в Git.

Git использует два основных типа тегов: легковесные и аннотированные.

Легковесный тег — это что-то очень похожее на ветку, которая не изменяется — просто указатель на определённый коммит.

А вот аннотированные теги хранятся в базе данных Git как полноценные объекты. Они имеют контрольную сумму, содержат имя автора, его e-mail и дату создания, имеют комментарий и могут быть подписаны и проверены с помощью GNU Privacy Guard (GPG). Обычно рекомендуется создавать аннотированные теги, чтобы иметь всю перечисленную информацию; но если вы хотите сделать временную метку или по какой-то причине не хотите сохранять остальную информацию, то для этого годятся и легковесные.

Если вы хотите получить версии файлов, на которые указывает тег, то вы можете сделать git checkout для тега. Однако, это переведёт репозиторий в состояние «detached HEAD», которое имеет ряд неприятных побочных эффектов.

**Порядок выполнения работы:**

1.Изучил теоретический материал.

2. Создал общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и выбранный язык программирования.

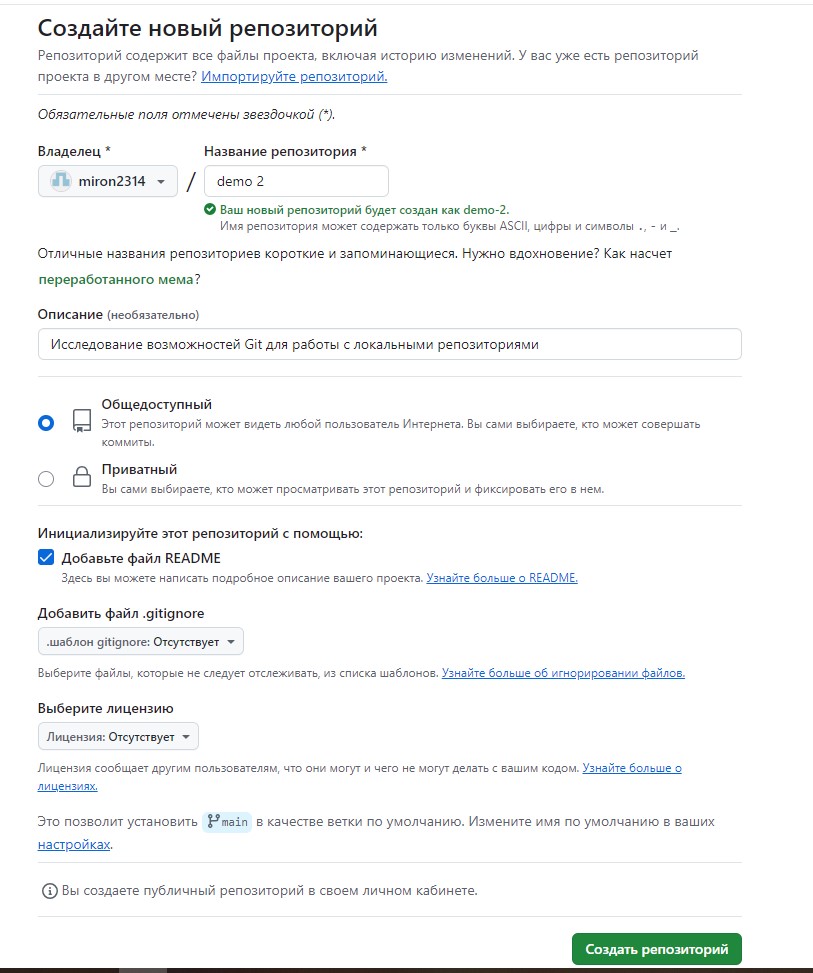


Рисунок 1. Создание репозитория

3.Проработал примеры лабораторной работы. Отразил вывод на консоли при выполнении команд git в отчете для лабораторной работы.

4.Выполнил клонирование созданного репозитория на рабочий компьютер.

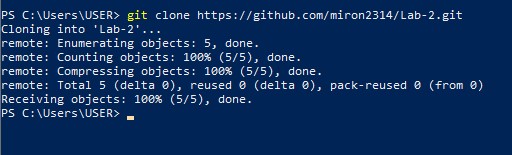


Рисунок 2. Клонирование репозитория

5. Добавил в файл README.md информацию о дисциплине, группе и ФИО студента, выполняющего лабораторную работу.

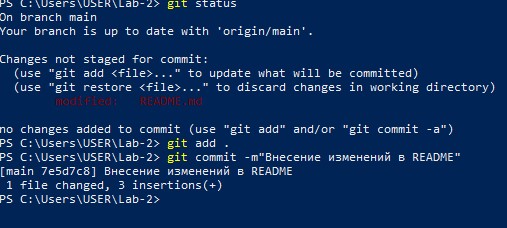


Рисунок 3. Добавление информации в файл

6.Написал небольшую программу на выбранном языке программирования. Фиксировал изменения при написании программы в локальном репозитории. Сделано не менее 7 коммитов, отмеченных не менее 3 тэгами.

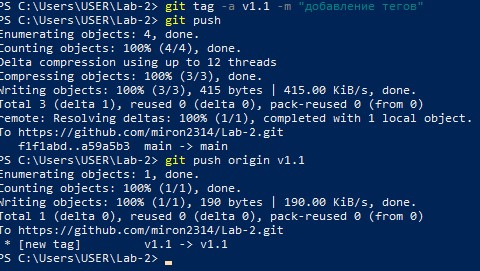


Рисунок 4. Добавление тегов

7. Просмотрел историю хранилища командой git log. Добавил скриншот консоли с выводом в отчет по лабораторной работе.

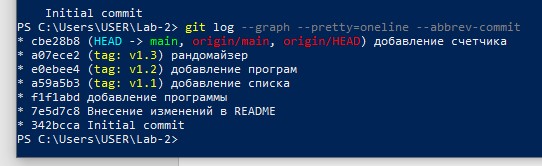
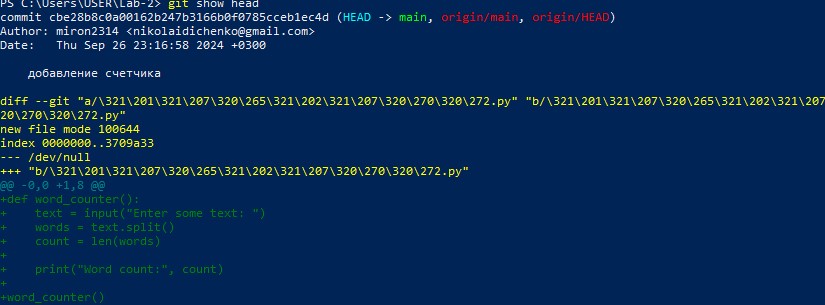


Рисунок 5. Использование команды git log

8.Просмотрел содержимое коммитов командой git show. Отобразил результаты работы команд в отчете.



# Рисунок 6. Использование команды git show head

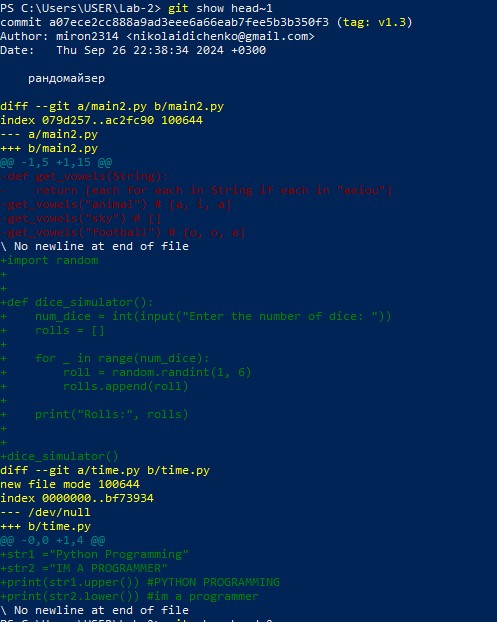
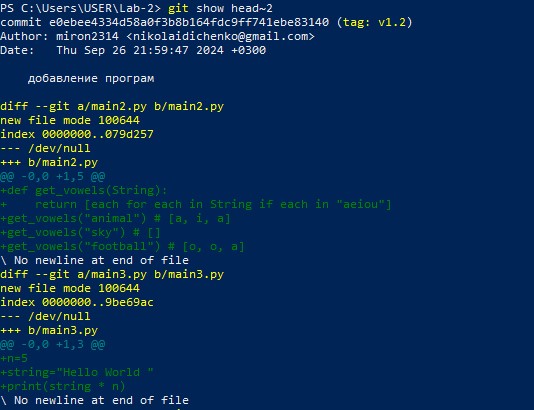
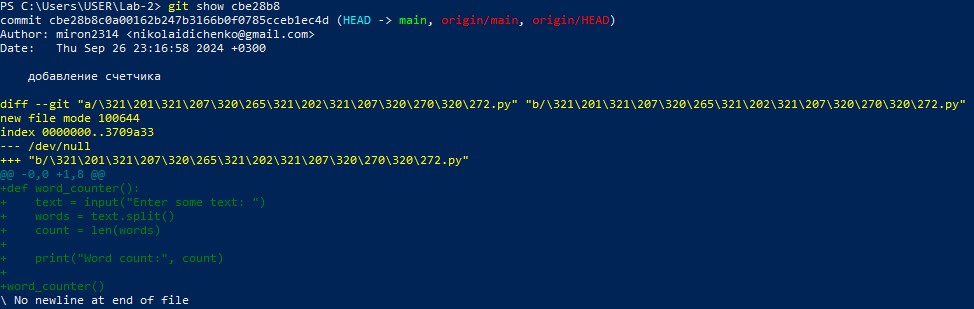


Рисунок 7. Использование команды get show head~1



# Рисунок 8. Использование команды git show head~2



# Рисунок 9. Использование команды git show для коммита с хэшем

9.Освоил возможность отката к заданной версии. Удалил весь код из одного из файлов программы репозитория и сохранил этот файл.

Удалил все несохраненные изменения в файле командой: git checkout

сделал выводы об изменении содержимого файла программы после выполнения отката. Отразил эти выводы в отчете.



Рисунок 10. Откат состояния хранилища

**Вывод:** в результате использования команды git reset –hard отменяется

последние коммиты и сбрасывается рабочая копия до определенного состояния. Git checkout позволяет управлять содержимым рабочего каталога и возвращаться к прошлым состояниям файлов и веток.

10.Зафиксировал сделанные изменения.

11.Добавил отчет по лабораторной работе в формате PDF в папку doc репозитория. Зафиксировал изменения.

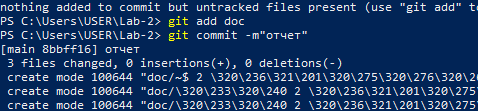


Рисунок 11. Добавление отчета в репозиторий

12.Отправил изменения из локального репозитория в удаленный репозиторий GitHub.

13. Проконтролировал изменения, произошедшие в репозитории GitHub.

14.Самостоятельно изучил работу с сервисами GitLab. Создал репозиторий на одном из этих сервисов. Создал зеркало репозитория на GitHub с использованием созданного репозитория.

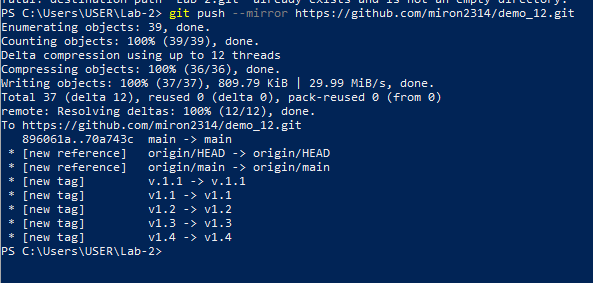


Рисунок 12. Создание зеркала

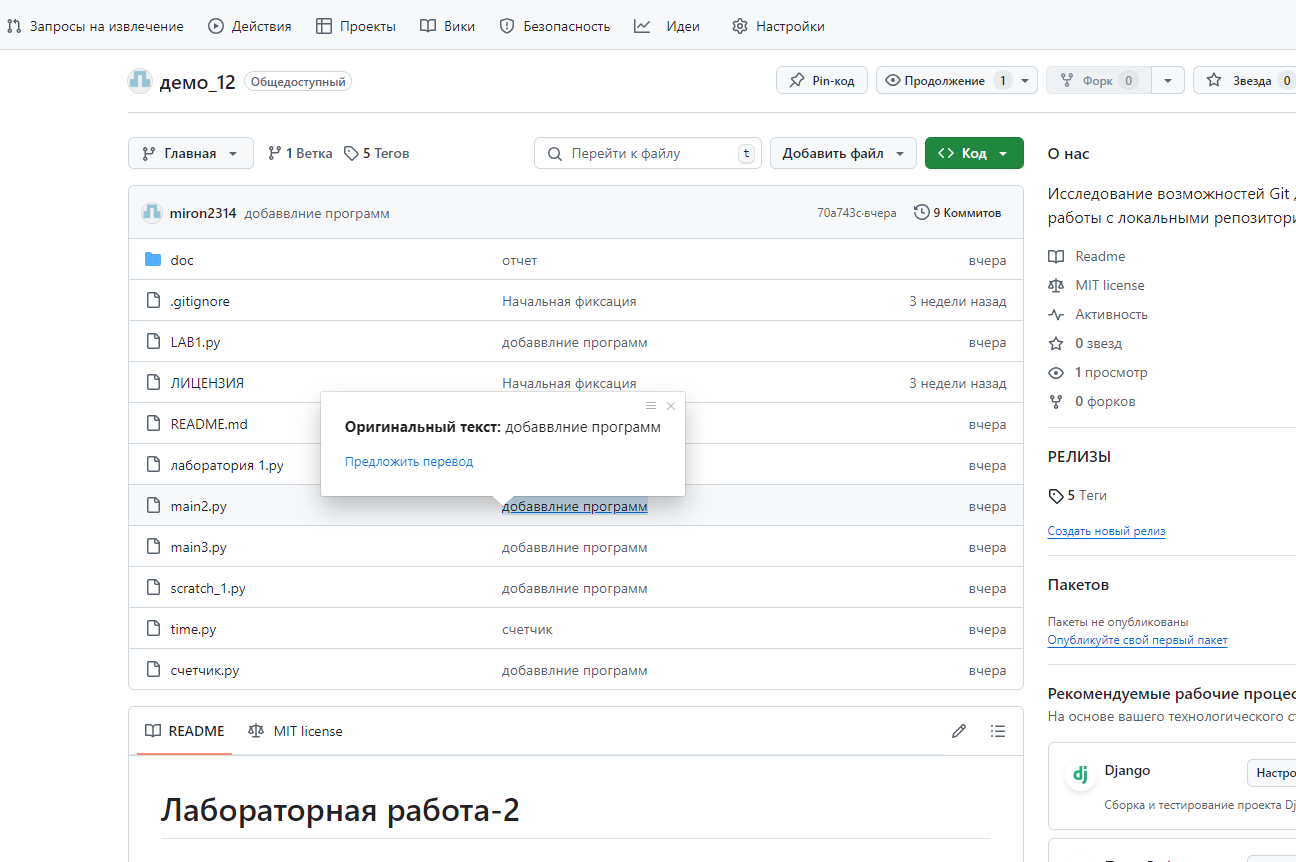


Рисунок 13. Отзеркаленный репозиторий

**Ответы на контрольные вопросы:**

1 Как выполнить историю коммитов в Git? Какие существуют дополнительные опции для просмотра истории коммитов? – Для просмотра истории коммитов используется команда git log. Она показывает все предыдущие коммиты, их автора, сообщения коммита и дату создания.

2 Как ограничить вывод при просмотре истории коммитов? – Использование ключей --limit и --max-count позволяет ограничить количество отображаемых коммитов.

3 Как внести изменения в уже сделанный коммит? – Для внесения изменений в существующий коммит можно использовать команду git commit --amend. Эта команда изменяет последний коммит и перезаписывает его.

4 Как отменить индексацию файла в Git? – Если файл был добавлен в индекс, но еще не зафиксирован в коммите, его можно убрать из индекса командой git rm --cached.

5 Как отменить изменения в файле? Чтобы вернуть файл к исходному состоянию, можно использовать команды git checkout или git reset. Команда git checkout позволяет восстановить предыдущую версию файла, а git reset отменяет последние изменения

6 Что такое удаленный репозиторий Git? – Удаленный репозиторий – это другой репозиторий Git, который находится на другом компьютере или сервере. Это может быть частный сервер, GitHub, GitLab или любой другой хостинг Git.

7 Как выполнить просмотр удаленных репозиториев данного локального репозитория? – Для просмотра удаленных репозиториев используется команда git remote -v.

8 Как добавить удаленный репозиторий для данного локального репозитория? – Для добавления удаленного репозитория используется команда git remote add.

9 Как выполнить отправку/получение изменений с удаленного репозитория? – Для отправки изменений используется команда git push, а для получения изменений – git pull.

10 Как выполнить просмотр удаленного репозитория? – Просмотр удаленного репозитория возможен через интерфейсы GitHub, GitLab или другого хостинга. Также можно использовать SSH-ключ для доступа к удаленному репозиторию.

11 Каково назначение тэгов Git? – Тэги в Git используются для отметки определенных моментов в истории репозитория, таких как выпуски версий, важные события или стабильные сборки. Они позволяют легко вернуться к определенной версии проекта и упрощают процесс управления версиями.

12 Как осуществляется работа с тэгами Git? – Работа с тэгами Git включает создание, использование и управление ими. Создание тэга выполняется с помощью команды git tag. Например, для создания тэга с именем v1.0.0 нужно ввести следующую команду:

$ git tag v1.0.0

После создания тэга его можно использовать для перехода к определенной версии проекта. Для этого нужно использовать команду git checkout:

$ git checkout v1.0.0

13 Самостоятельно изучите назначение флага --prune в командах git fetch и git push. Каково назначение этого флага? – Флаг --prune в командах git fetch и git push используется для удаления удаленных веток и тэгов, которые больше не существуют на удаленном сервере. Этот флаг помогает поддерживать актуальность локальных копий удаленных репозиториев. Он работает следующим образом:

- При использовании вместе с командой git fetch, флаг --prune удаляет удаленные ветки и тэги, которые были удалены на удаленном сервере.

- При использовании вместе с командой git push, флаг --prune предотвращает отправку данных об удаленных ветках и тэгах, которые больше не существуют на локальном репозитории.

Таким образом, этот флаг помогает держать локальную копию репозитория в актуальном состоянии, удаляя ненужные данные о удаленных объектах.

**Вывод:** в ходе выполнения лабораторной работы исследовал базовые возможности системы контроля версий Git для работы с локальными репозиториями.