Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет

информатики и радиоэлектроники»

Факультет компьютерного проектирования

Кафедра инженерной психологии и эргономики

ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ ПРИЛОЖЕНИЙ

ОТЧЁТ

по практической работе №1

«Основы Git»

Студент гр.210901 Казеко Д.А.

Проверил Давыдович К.И.

Минск 2024

**Цель:** изучить команды Git и научаться применять их на практике.

Таблица 1. Основные команды Git

|  |  |
| --- | --- |
| git branch -m old new | Переименовать ветку |
| git reset | Откатить команду git add |
| git log    git log –follow css/style.css  git log --all --graph | История коммитов  Посмотреть в каких коммитах изменялся файл: надо дописать название файла  История коммитов файла включая до перемещения/ переназвания файла  Просмотр веток и расхождений |
| git checkout <hash> | Вернуться к прошлому коммиту |
| git switch main | Вернуться к поледней версии кода нужной ветки |
| git tag  git tag v1  git tag -d v1  git checkout v1  git checkout v1^ | Все тэги  Создание тэга  Удалить тэг  Переход  Переход на предыдущий коммит перед тегом |
| git checkout hello.html | Отменить изменение файла перед его git add |
| git reset HEAD hello.html | Откатить изменение после git add на состояние до git add |
| git revert HEAD | Откатить коммит созданием нового |
| git reset --hard v1  git reset --hard HEAD~2 | Откатить коммит без создания нового. Если он имеет тэг, его можно будет увидеть в git log –all  Откатиться на 2 коммита назад |
| git reset --soft HEAD~1 | Откатит коммит и оставит изменения файлов |
| git add hello.html  git commit --amend -m "text" | Добавить файлы в прошлый коммит или переименовать его |
| git checkout -b  git switch -c | Создание ветки |
| git checkout  git switch | Переключение веток |
| git show v1 | Посмотреть все изменения в коммите |
| git merge main | Слить в текущую ветку изменения из main |
| git merge --abort | Отмена слияния |
| git rebase main | Слияние через преобразование rebase  !! Переписывает историю коммитов, не использовать при разработке командой |
| git add .  git rebase --continue | После разрешения конфликта |

Таблица 2. Команды Git для работы с репозиториями

|  |  |
| --- | --- |
| git clone work home | Клонирование репозитория work. Даём ему название home |
| git remote  git remote show origin | Узнать имя по умолчанию удалённого репозитория  Подробная информация |
| git branch -a | Список удалённых веток |
| git fetch  git merge origin/main | Подтянуть изменения из удалённой ветки, но не сливать их с текущей  А теперь сливаем |
| git pull | Подтягивает и сливает |
| git branch --track style origin/style | Добавление локальной ветки, которая отслеживает изменения удаленной ветки |
| git clone --bare work work.git | Создание чистого репозитория (названия заканчиваются на .git) |
| git remote add shared ../work.git | Добавить чистый репозиторий в качестве удаленного к нашему оригинальному. |
| git push shared main | Отправить изменения на общий репозиторий |
| git remote add shared ../work.git  git branch --track shared main  git pull shared main | Подтянуть изменения из общего репозитория |

Первое, с чего началось обучение в курсе GitHowTo - это добавление страницы в репозиторий (рис. 1)

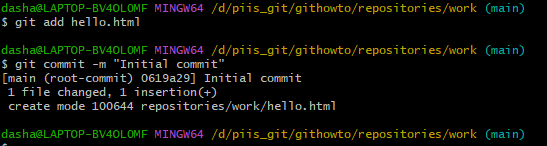


Рисунок 1 - добавление страницы «Hello, World» в репозиторий

После добавления страницы в репозиторий, я научилась проверять состояние репозитория с помощью команды git status (рис. 2)



Рисунок 2 - проверка состояния.

Следующим шагом в изучении было научиться вносить изменения. В ходе чего мы изменяем наполненность файла Hello.html и прописываем git status (рис. 3)

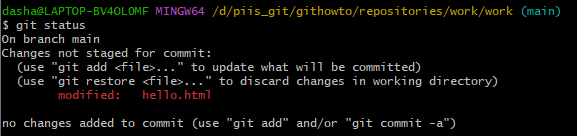


Рисунок 3 - Внесение изменений в файл Hello.html

Однако изменения не были индексированными, а чтобы их проиндексировать мы использовали команду git add и только потом уже проверили статус (рис.4)

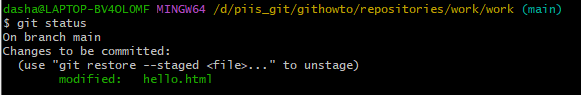


Рисунок 4 - Индексация изменений

Далее я училась коммитить изменения, для этого нужно было ввести git commit, после чего открывался редактор, в котором мне нужно было ввести added h1 tag, сохранить файл и выйти из редактора, как результат мы получили коммит изменения и после проверки статуса рабочая директория оказалась пуста (рис.5)

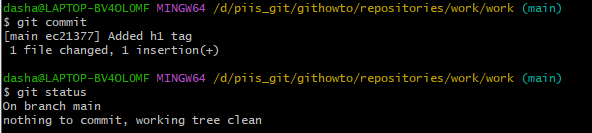


Рисунок 5 - коммит изменений

Далее автор курса хотел дать понять то, что Git работает с изменениями, а не с файлами.

Первое, что я сделала - это изменила страницу Hello.html, теперь она содержала стандартные теги  <html> и <body>, после выполнила команду git add hello.html, а после еще раз добавила изменения, теперь файл содержал еще и заголовок, после я выполнила команду git status (рис. 6)

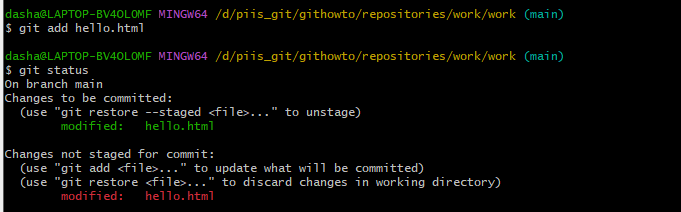


Рисунок 6 - результат выполнения двух изменений и проверки статуса

Далее я произвела коммит проиндексированного изменения, а зачем еще раз проверила состояние (рис.7)

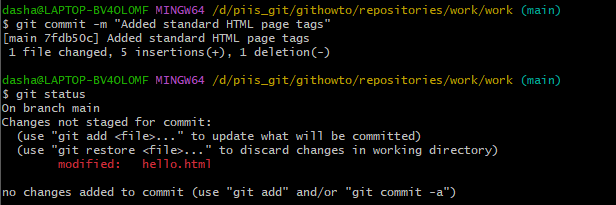


Рисунок 7 - коммит проиндексированного изменения и проверка состояния

Второе изменение мы добавили в индекс через команду git add . , использовав текущую директорию (.) в качестве аргумента для добавления (рис. 8). После чего я сделала коммит.

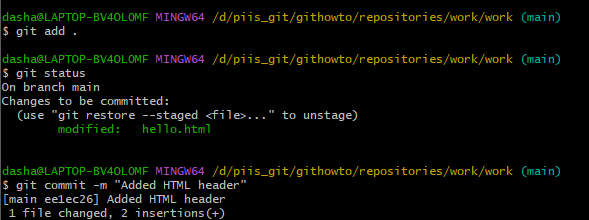


Рисунок 8 - идексация изменения через git add . и коммит.

Далее мной было изучено, как просматривать историю изменений проекта, это делается через команду git log, можно также сократить количество текста через команду однострочного просмотра (git log --pretty=oneline), ну и самый «крутой» вариант просмотра есть команда git log --pretty=format:"%h %ad | %s%d [%an]" --date=short (рис. 9)

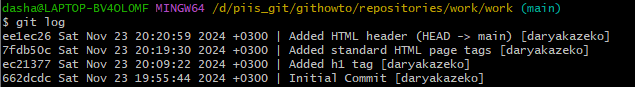


Рисунок 9 - просмотр истории изменений проекта

Следующим этапом курса было научиться возвращать рабочую директорию к любому предыдущему состоянию, для начала нужно было получить хеш предыдущих коммитов (рис. 10)

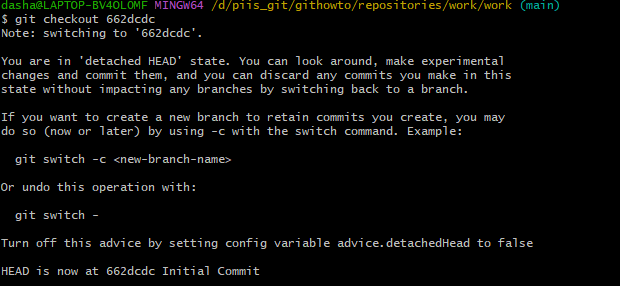


Рисунок 9 - Хеш предыдущих коммитов.

Далее я вернулась к последней версии в ветке main c помощью команды switch (рис. 10)

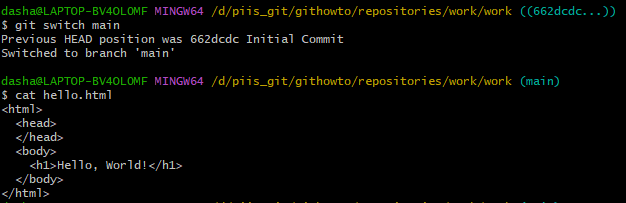


Рисунок 10 - переключение на последнюю версию ветки main.

Далее я училась создавать теги для коммитов для использования в будущем. Назвала текущую версию страницы v1, она же и являлась первой (рис. 11)

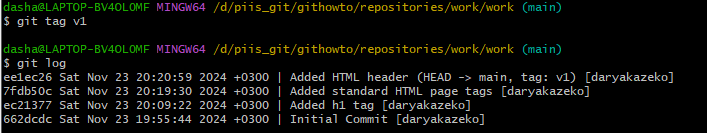


Рисунок 11 - тэг для текущей версии страницы

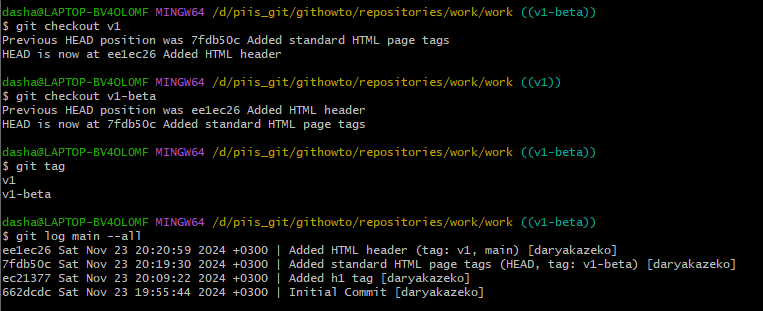
После я создала еще одну версию с <head>, она же v1-beta. После переключалась и просматривала версии кода через tag, а также просмотр всех тегов в логе (рис.12)  


Рисунок 12 - создание, просмотр тегов

Далее нужно было научиться отменять локальные изменения до индексации и отменять проиндексированные изменения перед коммитом.

Для отмены локальных изменений до индексации нужно находиться на ветке main, произвести изменения в коде, проверить состояние, а после отменить изменения в рабочем каталоге (рис.13)

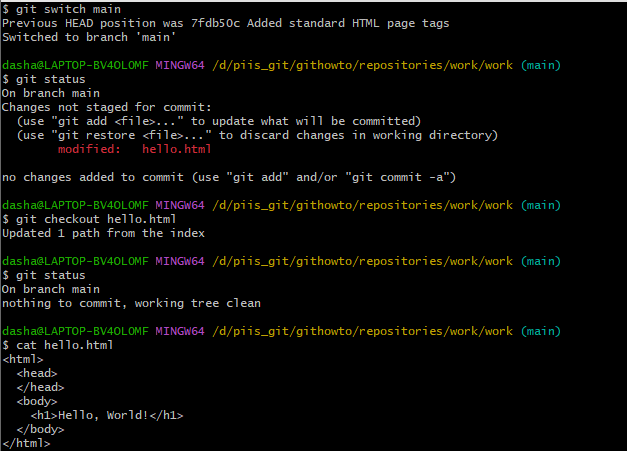


Рисунок 13 - отмена локального изменения до индексации

Отмена проиндексированных изменений производитс через reset, а потом добить надо командой checkout (рис.14)

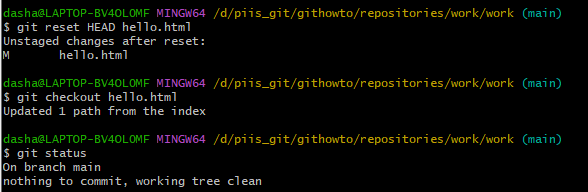


Рисунок 14 - отмена проиндексированных изменений

Удаление коммитов из ветки (рис. 15)

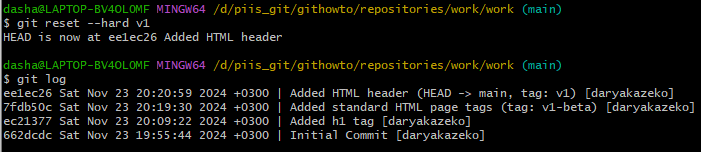


Рисунок 15 - удаление коммита из ветки

Я создавала тег Oops для того, чтобы проверить удаление коммитов из ветки, тег свою функцию выполнил, теперь удалим этот тег (Рис. 16)

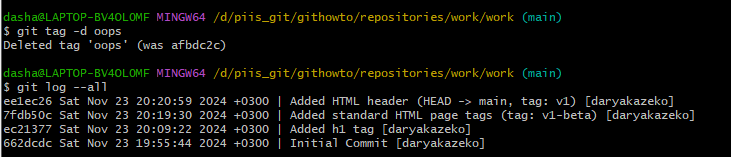


Рисунок 16 - удаление тега

Далее нужно было научиться вносить изменения в коммиты. Я добавила комментарий в коде, проиндексировала изменения и сделала коммит, однако после захотела добавить почту, чтобы не делать отдельный коммит мы просто вносим изменения в предыдущий коммит (рис. 17)

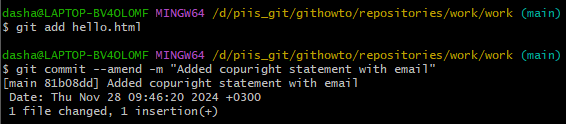


Рисунок 17 - внесение изменений в коммит

Следующее, что нужно было научиться делать - это создание ветки. Чтобы стилизовать наш проект нужно добавить стили css, для этого мы вынесли это в отдельную ветку, а после проиндексировали изменения и сделали коммит (рис.16)



Рисунок 16 - создание ветки css



Рисунок 17 - переключение веток

Перемещение файлов: мне нужно было перенести файл css в отдельный репозиторий, а также сделать ренейм файла hello.html на index.html (рисунок 18)

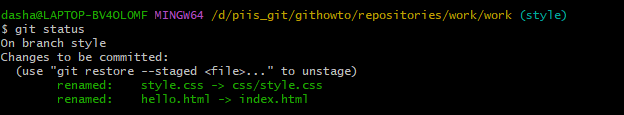


Рисунок 18 - перемещение и ренейм файлов

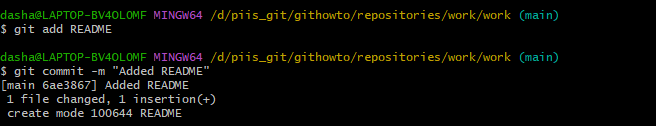


Рисунок 19 - создание, индексация и коммит файла README

После разделения веток нужно было просмотреть отличающиеся ветки, что я сделала с помощью команды git log --all --graph (рис. 20)

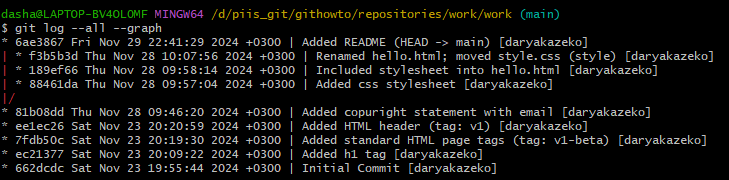


Рисунок 20 - просмотр отличающихся веток

Следующим шагом было слияние двух веток, что делается через команду git merge main, следствием чего является слияние двух веток (рис. 21)

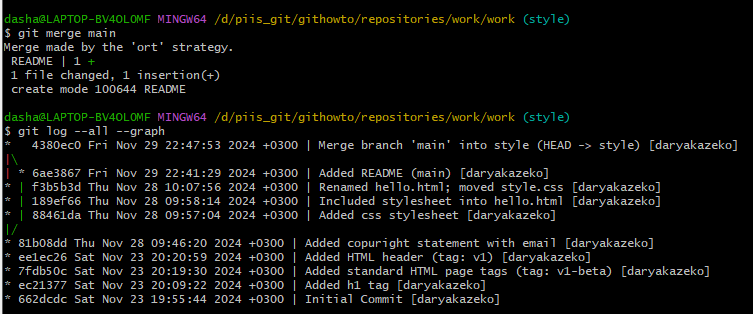


Рисунок 21 - слияние двух веток

Несколько следующих уроков были посвящены разнице слияния и перебазирования, то есть сравнение команд rebase и merge, в ходе тех уроков я выяснила, что команду rebase лучше использовать когда получаем изменения из удаленного репозитория и хотим применить их к своей локальной ветке, а также если хотим, чтобы история коммитов была линейной и легко читаемой. Команду merge лучше использовать если текущая ветка является публичной и общей. Переписывание таких веток будет мешать работе других членов команды или же если важна точная история ветки коммитов (поскольку команда rebase переписывает историю коммитов).

Далее я слила все в ветку main (рис. 22)

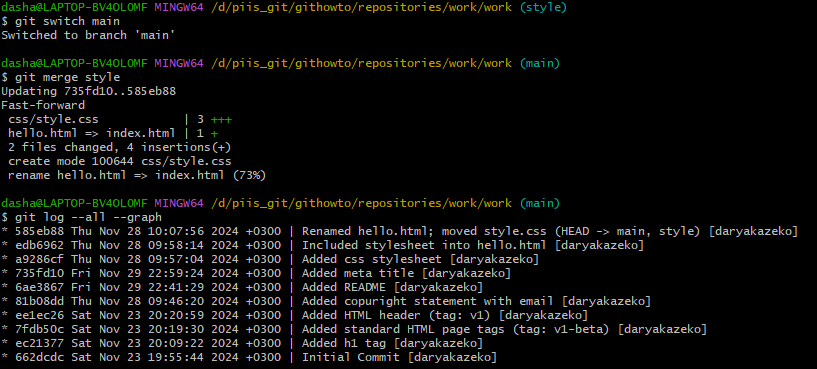


Рисунок 22 - слияние в ветку main

Во второй части курса рассказывалось о том, как работать с несколькими репозиториями.

Первое, что нужно было сделать - это склонировать репозиторий и проверить его сходство с оригинальным (рис. 23)

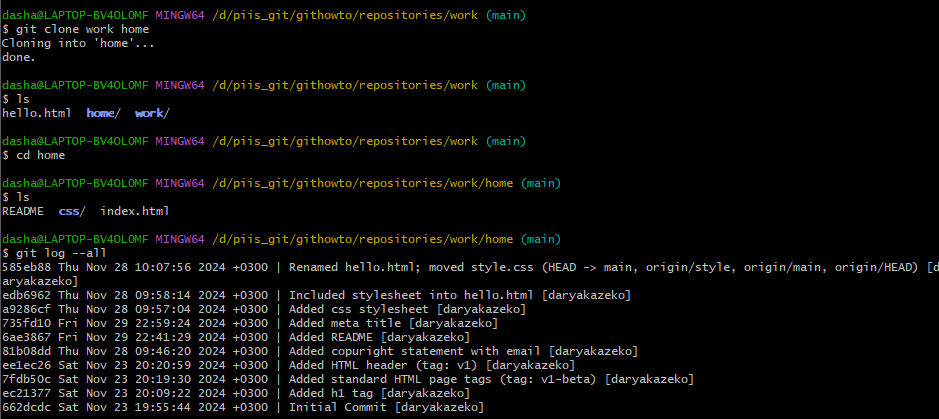


Рисунок 23 - Клонирование репозитория work

Далее я разбиралась, как узнать об именах удаленных репозиториев. При удалении клонированный репозиторий знает об имени по умолчанию, но также можно получить более подробную информацию об имени по умолчанию (рис. 24)

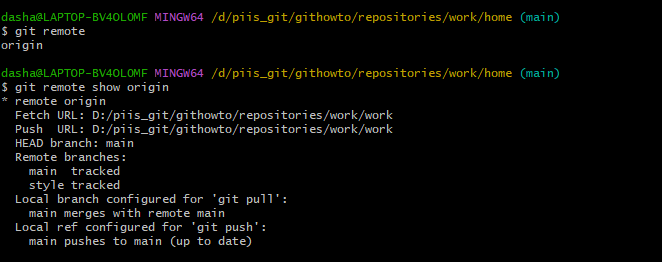
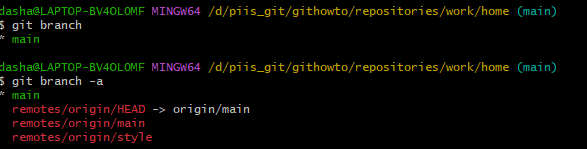


Рисунок 24 - удаление и подробная информация по удаленному репозиторию



Рисуенок 25 - Информация об удаленных ветках

Далее я училась подтягивать изменения в ветки, которые были изменены и объеденены (Рис. 26)

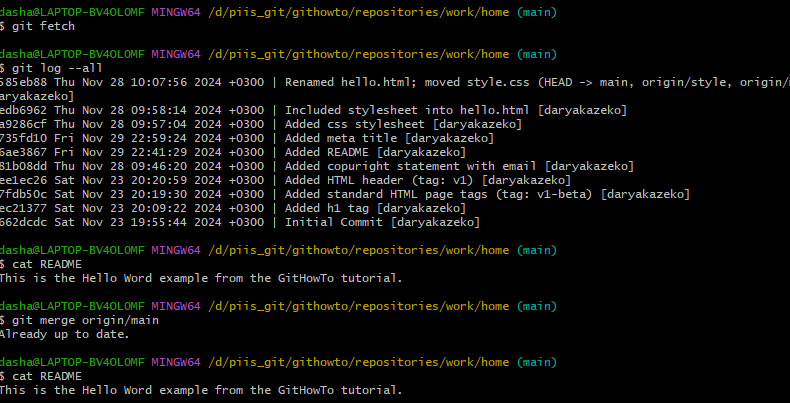


Рисунок 26 - подтягивание изменений

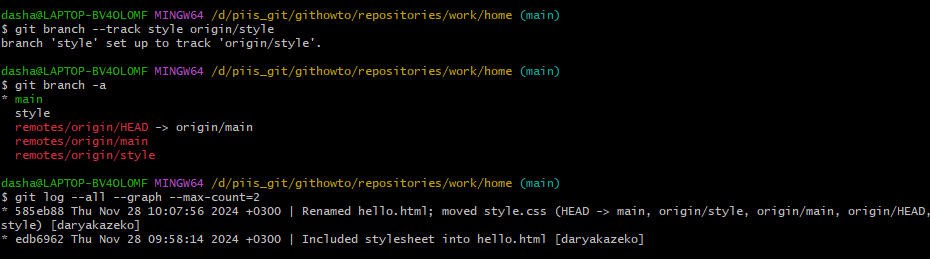


Рисунок 27 - Добавление ветки наблюдения



Рисунок 28 - создание чистого репозитория

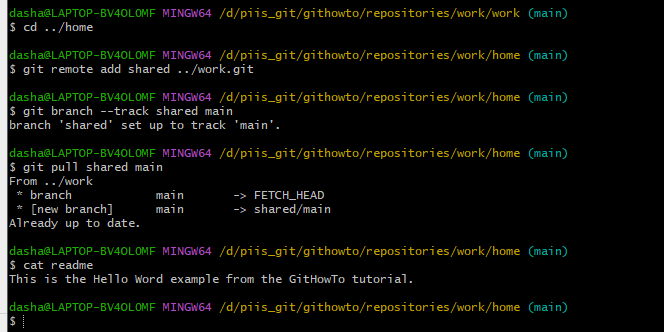


Рисунок 29 - Подтягивание общих изменений