

Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет информатики и
радиоэлектроники»

Кафедра инженерной психологии и эргономики

Пользовательские интерфейсы информационных систем

Отчет по практическим занятиям на тему
«Образовательный курс GitHowTo»

Выполнила:
студентка гр.
210901
Маслова Ю.Ю.

Проверил:
Давыдович К. И.

Минск 2024

Цель: сформировать понимание и специфику работы с инструментом контроля версий «Git» и научиться пользоваться его основным функционалом. Курс располагается по ссылке <https://githowto.com/ru>.

Прохождение курса

Часть I: Основы Git

Задание 1. Приготовления

Если вы никогда ранее не использовали Git, для начала вам необходимо осуществить установку. Выполните следующие команды, чтобы Git узнал ваше имя и электронную почту. Эти данные используются для подписи изменений сделанных вами, что позволит отслеживать, кто и когда сделал изменения в файле.

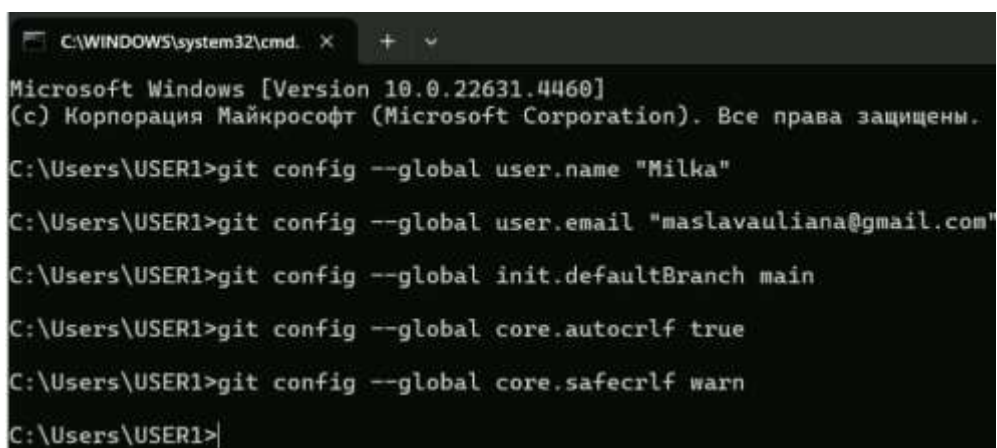
```
git config --global user.name "Your Name"
git config --global user.email "your_email@whatever.com"
```

Мы будем использовать main в качестве имени ветки по умолчанию. Чтобы установить его, выполните следующую команду:

```
git config --global init.defaultBranch main
```

Также нужно выполнить данные команды для корректной обработки окончаний строк (для Windows):

```
git config --global core.autocrlf true
git config --global core.safecrlf warn
```

A screenshot of a Windows command prompt window. The title bar shows 'C:\WINDOWS\system32\cmd. X'. The window content shows the following text: 'Microsoft Windows [Version 10.0.22631.4460] (c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены. C:\Users\USER1>git config --global user.name "Milka" C:\Users\USER1>git config --global user.email "maslavauliana@gmail.com" C:\Users\USER1>git config --global init.defaultBranch main C:\Users\USER1>git config --global core.autocrlf true C:\Users\USER1>git config --global core.safecrlf warn C:\Users\USER1>'.

```
C:\WINDOWS\system32\cmd. X
Microsoft Windows [Version 10.0.22631.4460]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.
C:\Users\USER1>git config --global user.name "Milka"
C:\Users\USER1>git config --global user.email "maslavauliana@gmail.com"
C:\Users\USER1>git config --global init.defaultBranch main
C:\Users\USER1>git config --global core.autocrlf true
C:\Users\USER1>git config --global core.safecrlf warn
C:\Users\USER1>
```

Рисунок 1 – Начало работы с Git

Задание 2. Создание проекта

Начните работу в пустой директории (например, repositories, если вы скачали архив с предыдущего шага) с создания пустой поддиректории work, затем войдите в неё и создайте там файл hello.html с таким содержанием:

```
mkdir work
```

```
cd work
touch hello.html
```

Содержание файла:

Hello, World

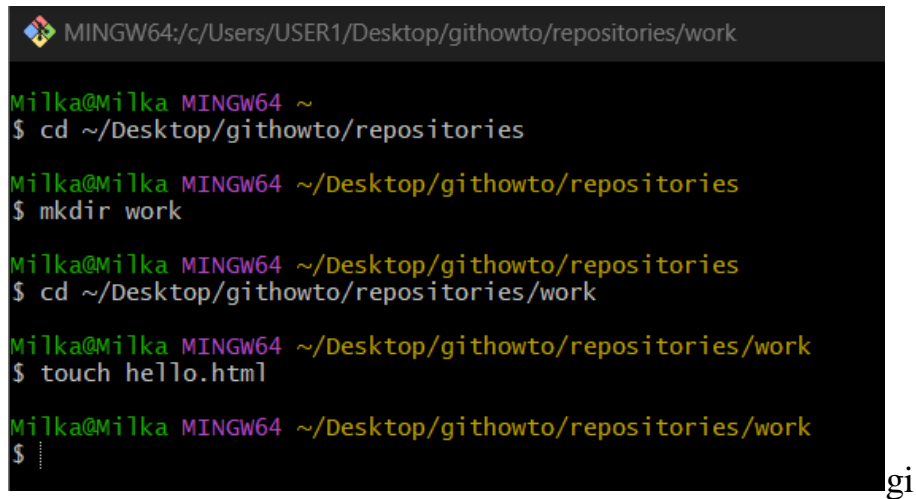
A screenshot of a terminal window with a dark background. The title bar shows the path 'MINGW64:/c/Users/USER1/Desktop/githowto/repositories/work'. The terminal shows a series of commands and their outputs: 'cd ~/Desktop/githowto/repositories', 'mkdir work', 'cd ~/Desktop/githowto/repositories/work', 'touch hello.html', and a prompt '\$' followed by a vertical ellipsis '...'.

Рисунок 2 – Создание файла hello.html

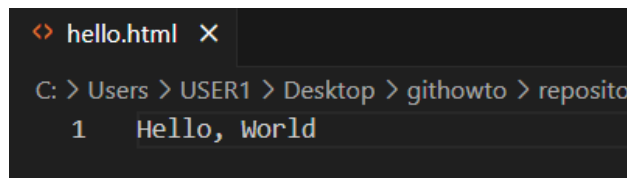
A screenshot of a code editor window. The title bar shows 'hello.html' and a close button. The editor shows a single line of code: '1 Hello, World'.

Рисунок 3 – Содержание файла hello.html

Теперь у вас есть директория с одним файлом. Чтобы создать Git-репозиторий из этой директории, выполните команду `git init`.

```
git init
```

Теперь давайте добавим в репозиторий страницу «Hello, World».

```
git add hello.html
git commit -m "Initial Commit"
```

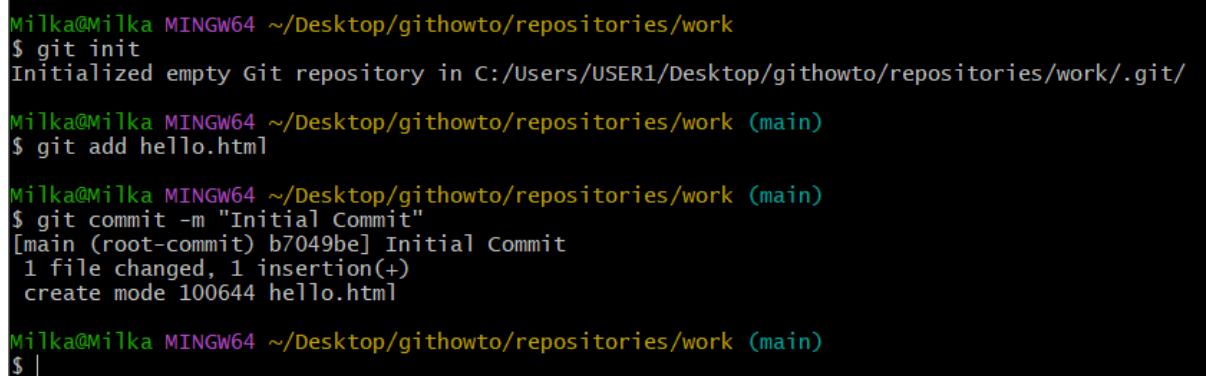
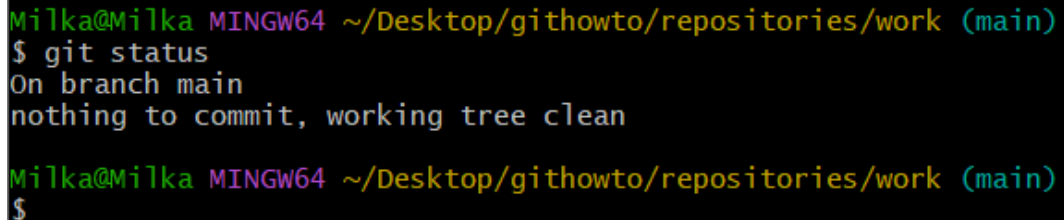
A screenshot of a terminal window with a dark background. The title bar shows the path 'MINGW64:/c/Users/USER1/Desktop/githowto/repositories/work'. The terminal shows the following commands and outputs: 'git init' (Initialized empty Git repository in C:/Users/USER1/Desktop/githowto/repositories/work/.git/), 'git add hello.html', 'git commit -m "Initial Commit"' ([main (root-commit) b7049be] Initial Commit, 1 file changed, 1 insertion(+), create mode 100644 hello.html), and a final prompt '\$'.

Рисунок 4 – Добавление файла hello.html в Git-репозиторий

Задание 3. Проверка состояния

Используйте команду `git status`, чтобы проверить текущее состояние репозитория.

```
git status
```



```
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$ git status
On branch main
nothing to commit, working tree clean

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$
```

Рисунок 5 – Проверка текущего состояния репозитория

Задание 4. Внесение изменений

Добавим кое-какие HTML-теги к нашему приветствию. Измените содержимое файла на:

```
<h1>Hello, World!</h1>
```

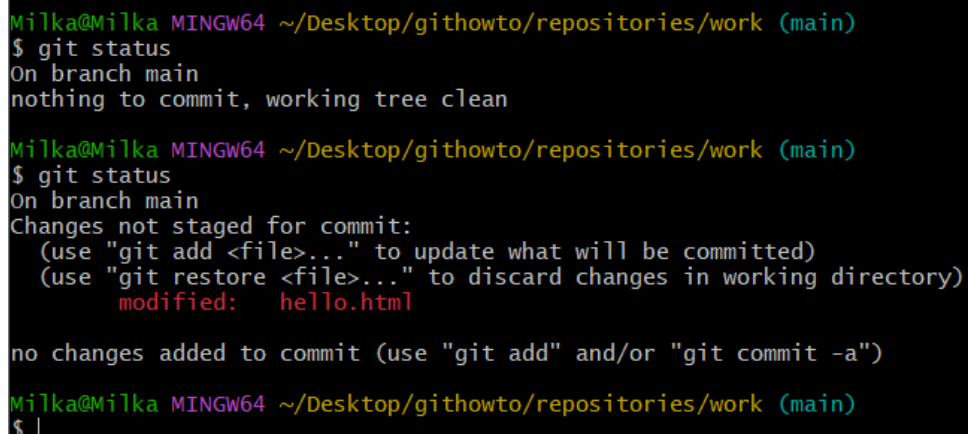


```
<> hello.html X
C: > Users > USER1 > Desktop > githowto > repositories >
1  <h1>Hello, world</h1>
```

Рисунок 6 – Изменение файла hello.html

Теперь проверьте состояние рабочей директории.

```
git status
```



```
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$ git status
On branch main
nothing to commit, working tree clean

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$ git status
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   hello.html

no changes added to commit (use "git add" and/or "git commit -a")

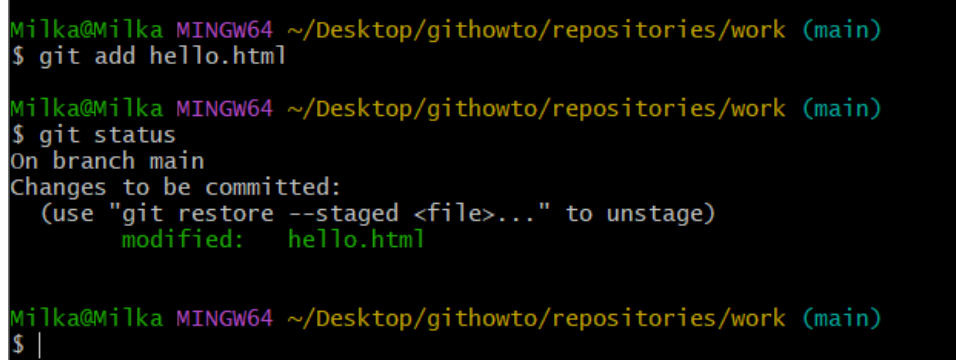
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$ |
```

Рисунок 7 – Проверка состояния репозитория

Задание 5. Индексация изменений

Теперь дайте команду Git проиндексировать изменения. Проверьте состояние:

```
git add hello.html
git status
```



```
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$ git add hello.html

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   hello.html

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$ |
```

Рисунок 8 – Индексация изменений в репозитории

Задание 6. Индексация и коммит

Отдельный шаг индексации в Git позволяет вам разделять большие изменения на маленькие коммиты.

Предположим, что вы отредактировали три файла (a.html, b.html, и c.html). Теперь вы хотите закоммитить все изменения, при этом чтобы изменения в a.html и b.html были одним коммитом, в то время как изменения в c.html логически не связаны с первыми двумя файлами и должны идти отдельным коммитом.

В теории, вы можете сделать следующее:

```
git add a.html
git add b.html
git commit -m "Changes for a and b"

git add c.html
git commit -m "Unrelated change to c"
```

Разделяя индексацию и коммит, вы имеете возможность с легкостью настроить, что идет в какой коммит.

Задание 7. Коммит изменений

Достаточно об индексации. Давайте сделаем коммит того, что мы проиндексировали, в репозиторий.

Когда вы ранее использовали `git commit` для коммита первоначальной версии файла `hello.html` в репозиторий, вы включили метку `-m`, которая делает комментарий в командной строке. Команда `commit` позволит вам интерактивно редактировать комментарии для коммита. Теперь давайте это проверим.

Если вы опустите метку `-m` из командной строки, Git перенесет вас в редактор по вашему выбору. Редактор выбирается из следующего списка (в порядке приоритета):

- переменная среды `GIT_EDITOR`
- параметр конфигурации `core.editor`
- переменная среды `VISUAL`
- переменная среды `EDITOR`

Сделайте коммит сейчас и проверьте состояние.

```
git commit
```

В первой строке введите комментарий: Added h1 tag. Сохраните файл и выйдите из редактора.

```
MINGW64:/c/Users/USER1/Desktop/githowto/repositories/work
Added h1 tag
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch main
# Changes to be committed:
#   modified:   hello.html
#
~
~
~
~
~
~
~
~
~
~
.git/COMMIT_EDITMSG[+] [unix] (17:59 21/11/2024)
:wq
```

Рисунок 9 – Коммит

```
[main e58b204] Added h1 tag
1 file changed, 1 insertion(+), 1 deletion(-)
```

Рисунок 10 – Проверка состояния репозитория

В конце давайте еще раз проверим состояние.

```
git status
```

```

[main e58b204] Added h1 tag
1 file changed, 1 insertion(+), 1 deletion(-)

Milka@Milka MINGW64 ~/Desktop/ghitowto/repositories/work (main)
$ git status
On branch main
nothing to commit, working tree clean

Milka@Milka MINGW64 ~/Desktop/ghitowto/repositories/work (main)
$

```

Рисунок 11 – Проверка состояния репозитория

Рабочая директория чиста, можем продолжить работу.

Задание 8. Изменения, а не файлы

Большинство систем контроля версий работает с файлами: вы добавляете файл в систему, и она отслеживает изменения файла с этого момента.

Git фокусируется на изменениях в файле, а не самом файле. Когда вы осуществляете команду `git add file`, вы не говорите Git добавить файл в репозиторий. Скорее вы говорите, что Git надо отметить текущее состояние файла, коммит которого будет произведен позже.

Мы попытаемся исследовать эту разницу в данном уроке.

Измените страницу «Hello, World», чтобы она содержала стандартные теги `<html>` и `<body>`.

```

<html>
  <body>
    <h1>Hello, World!</h1>
  </body>
</html>

```



```

hello.html X
C:\Users\USER1\Desktop> ghitowto > repositories > w
1:  <html>
2:    <body>
3:      <h1>Hello, World!</h1>
4:    </body>
5:  </html>

```

Рисунок 12 – Изменение файла hello.html

Теперь добавьте это изменение в индекс Git.

```
git add hello.html
```

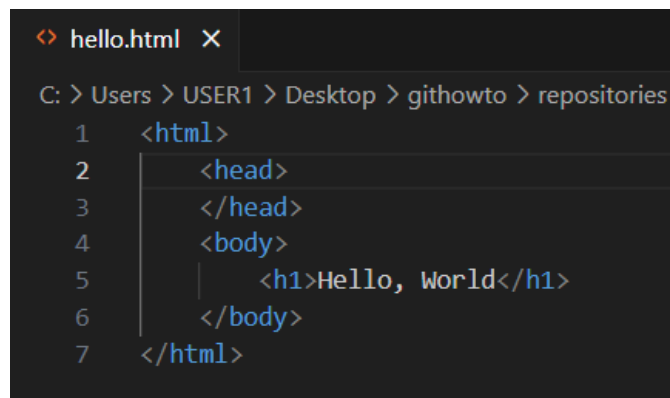
```
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$ git add hello.html

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$ |
```

Рисунок 13 – Добавление изменения в индекс Git

Теперь добавьте заголовки HTML (секцию <head>) к странице «Hello, World».

```
<html>
  <head>
</head>
  <body>
    <h1>Hello, World!</h1>
  </body>
</html>
```



```
<> hello.html X
C: > Users > USER1 > Desktop > githowto > repositories
1  <html>
2    <head>
3    </head>
4    <body>
5      <h1>Hello, World</h1>
6    </body>
7  </html>
```

Рисунок 14 – Изменение файла hello.html

Проверьте текущий статус

```
git status
```

```
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$ git status
On branch main
nothing to commit, working tree clean

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$ git add hello.html

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   hello.html

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   hello.html
```

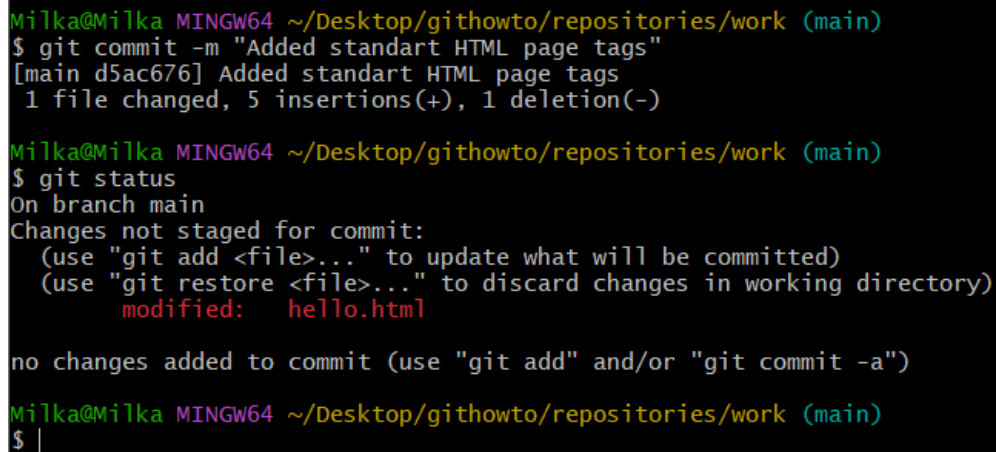
Рисунок 15 – Проверка состояния репозитория

Обратите внимание на то, что `hello.html` указан дважды в состоянии. Первое изменение (добавление стандартных тегов) проиндексировано и готово к коммиту. Второе изменение (добавление заголовков HTML) является неиндексированным. Если бы вы делали коммит сейчас, заголовки не были бы сохранены в репозиторий.

Давайте проверим.

Произведите коммит проиндексированного изменения (значение по умолчанию), а затем еще раз проверьте состояние.

```
git commit -m "Added standard HTML page tags"
git status
```



```
Milka@Milka MINGW64 ~/Desktop/ghithowto/repositories/work (main)
$ git commit -m "Added standart HTML page tags"
[main d5ac676] Added standart HTML page tags
1 file changed, 5 insertions(+), 1 deletion(-)

Milka@Milka MINGW64 ~/Desktop/ghithowto/repositories/work (main)
$ git status
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   hello.html

no changes added to commit (use "git add" and/or "git commit -a")

Milka@Milka MINGW64 ~/Desktop/ghithowto/repositories/work (main)
$ |
```

Рисунок 16 – Коммит и проверка состояния репозитория

Команда `status` показывает, что в файле `hello.html` ещё есть незаписанные изменения, но область подготовки уже пуста.

Теперь добавьте второе изменение в индекс, а затем проверьте состояние с помощью команды `git status`.

```
git add .
git status
```



```
Milka@Milka MINGW64 ~/Desktop/ghithowto/repositories/work (main)
$ git add .

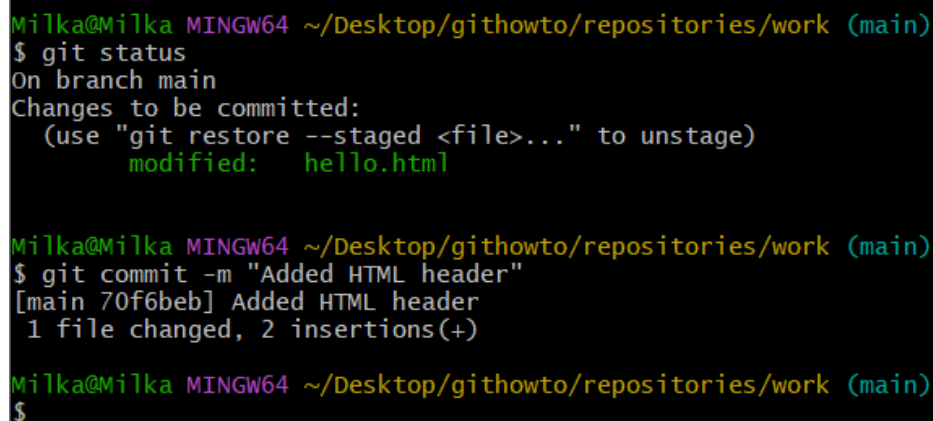
Milka@Milka MINGW64 ~/Desktop/ghithowto/repositories/work (main)
$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   hello.html

Milka@Milka MINGW64 ~/Desktop/ghithowto/repositories/work (main)
$ |
```

Рисунок 17 – Добавление второго изменения в индекс и проверка состояния репозитория

Второе изменение было проиндексировано и готово к коммиту. Сделайте коммит второго изменения.

```
git commit -m "Added HTML header"
```



```
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   hello.html

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$ git commit -m "Added HTML header"
[main 70f6beb] Added HTML header
1 file changed, 2 insertions(+)

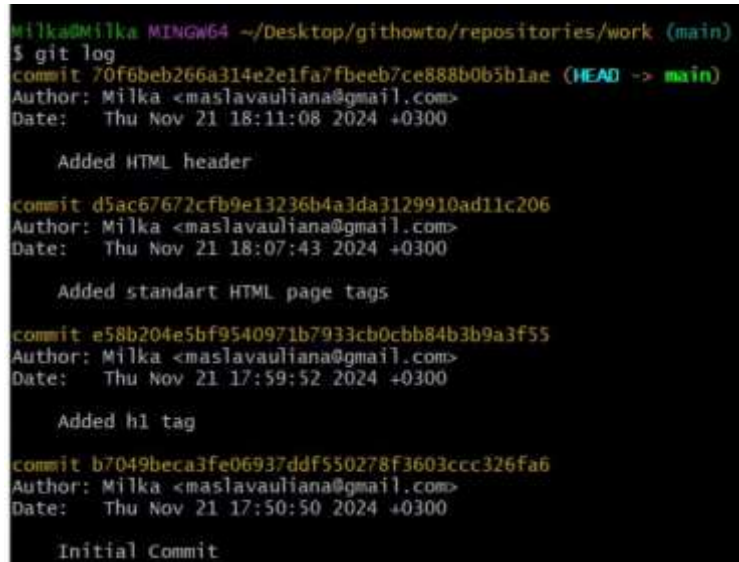
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$
```

Рисунок 18 – Коммит второго изменения

Задание 9. История

Получение списка произведенных изменений – функция команды git log.

```
git log
```



```
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$ git log
commit 70f6beb266a314e2e1fa7fbee7ce888b0b5blae (HEAD -> main)
Author: Milka <maslavauliana@gmail.com>
Date:   Thu Nov 21 18:11:08 2024 +0300

    Added HTML header

commit d5ac67672cfb9e13236b4a3da3129910ad11c206
Author: Milka <maslavauliana@gmail.com>
Date:   Thu Nov 21 18:07:43 2024 +0300

    Added standart HTML page tags

commit e58b204e5bf9540971b7933cb0cbb84b3b9a3f55
Author: Milka <maslavauliana@gmail.com>
Date:   Thu Nov 21 17:59:52 2024 +0300

    Added h1 tag

commit b7049beca3fe06937ddf550278f3603ccc326fa6
Author: Milka <maslavauliana@gmail.com>
Date:   Thu Nov 21 17:50:50 2024 +0300

    Initial Commit
```

Рисунок 19 – Получение списка изменений

Вот список всех четырех коммитов в репозиторий, которые мы успели совершить.

Вы полностью контролируете то, что отображает log. Например, однострочный формат:

```
git log --pretty=oneline
```

```

Milka@Milka MINGW64 ~/Desktop/ghowto/repositories/work (main)
$ git log --pretty=oneline
70f6beb266a314e2e1fa7fbee7ce888b0b5b1ae (HEAD -> main) Added HTML header
d5ac67672cfb9e13236b4a3da3129910ad11c206 Added standart HTML page tags
e58b204e5bf9540971b7933cb0cbb84b3b9a3f55 Added h1 tag
b7049beca3fe06937ddf550278f3603ccc326fa6 Initial Commit

Milka@Milka MINGW64 ~/Desktop/ghowto/repositories/work (main)
$

```

Рисунок 20 – Получение списка изменений в однострочном формате

Вот еще интересные варианты просмотра истории:

```

git log --oneline --max-count=2
git log --oneline --since="5 minutes ago"
git log --oneline --until="5 minutes ago"
git log --oneline --author="Your Name"
git log --oneline -all

```

Существует огромное количество вариантов просмотра истории, вы можете порыться на странице руководства git-log, чтобы увидеть их все.

Вот что я использую для просмотра изменений, сделанных за последнюю неделю. Я добавлю --author=Alexander, если я хочу увидеть только изменения, которые сделал я.

```
git log --all --pretty=format:"%h %cd %s (%an)" --since="7 days ago"
```

Со временем, я решил, что для большей части моей работы мне подходит следующий формат лога.

```
git log --pretty=format:"%h %ad | %s%d [%an]" --date=short
```

```

Milka@Milka MINGW64 ~/Desktop/ghowto/repositories/work (main)
$ git log --pretty=format: "%h %ad | %s%d [%an]" --date=short
fatal: option '--date=short' must come before non-option arguments

Milka@Milka MINGW64 ~/Desktop/ghowto/repositories/work (main)
$ git log --date=short --pretty=format: "%h %ad | %s%d [%an]"

Milka@Milka MINGW64 ~/Desktop/ghowto/repositories/work (main)
$

```

Рисунок 21 – Получение списка изменений в подходящем формате

Давайте рассмотрим его в деталях:

--pretty="..." — определяет формат вывода.

%h — укороченный хеш коммита.

%ad — дата коммита.

| — просто визуальный разделитель.

%s — комментарий.

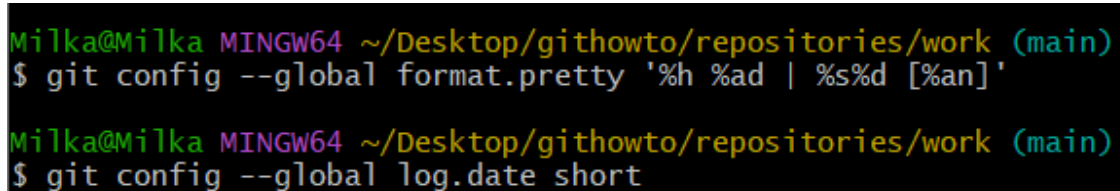
%d — дополнения коммита («головы» веток или теги).

%an — имя автора.

--date=short — сохраняет формат даты коротким и симпатичным.

Таким образом, каждый раз, когда вы захотите посмотреть лог, вам придется много печатать. К счастью, существует несколько опций конфигурации Git, позволяющих настроить формат вывода истории по умолчанию:

```
git config --global format.pretty '%h %ad | %s%d [%an]'  
git config --global log.date short
```



```
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)  
$ git config --global format.pretty '%h %ad | %s%d [%an]'  
  
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)  
$ git config --global log.date short
```

Рисунок 22 – Настройка формата вывода истории по умолчанию

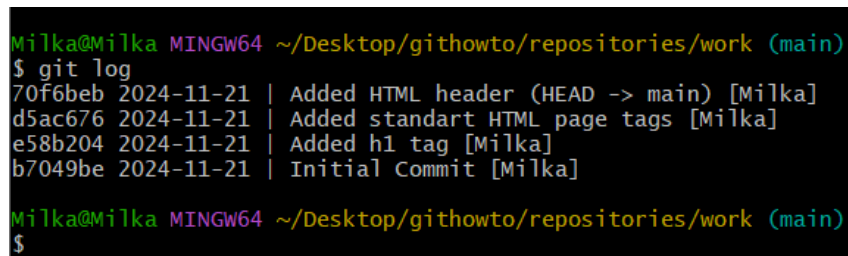
Оба gitx (для Mac) и gitk (для любой платформы) полезны в изучении истории изменений.

Задание 10. Получение старых версий

Git позволяет очень просто путешествовать во времени, по крайней мере, для вашего проекта. Команда checkout обновит вашу рабочую директорию до любого предыдущего коммита.

Получите хеши предыдущих коммитов:

```
git log
```



```
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)  
$ git log  
70f6beb 2024-11-21 | Added HTML header (HEAD -> main) [Milka]  
d5ac676 2024-11-21 | Added standart HTML page tags [Milka]  
e58b204 2024-11-21 | Added h1 tag [Milka]  
b7049be 2024-11-21 | Initial Commit [Milka]  
  
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)  
$
```

Рисунок 23 – Получение хешей предыдущих коммитов

Просмотрите историю изменений и найдите хеш первого коммита. Он должен быть в последней строке результата git log. Используйте этот хеш (достаточно первых 7 символов) в команде ниже. Затем проверьте содержимое файла hello.html.

```
git checkout <hash>  
cat hello.html
```

```
MINGW64~/c/Users/USER1/Desktop/ghowto/repositories/work

Milka@Milka MINGW64 ~/Desktop/ghowto/repositories/work (main)
$ git checkout b7049be
Note: switching to 'b7049be'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

    git switch -c <new-branch-name>

Or undo this operation with:

    git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at b7049be Initial Commit

Milka@Milka MINGW64 ~/Desktop/ghowto/repositories/work ((b7049be...))
$ cat hello.html
Hello, world
Milka@Milka MINGW64 ~/Desktop/ghowto/repositories/work ((b7049be...))
$ |
```

Рисунок 24 – Возвращение к первой версии файла hello.html

```
hello.html X
C: > Users > USER1 > Desktop > githowto > repositories
1 Hello, World
```

Рисунок 25 – Содержание файла hello.html

Обратите внимание, что сейчас содержимое файла hello.html — это тот самый текст, с которого мы начинали.

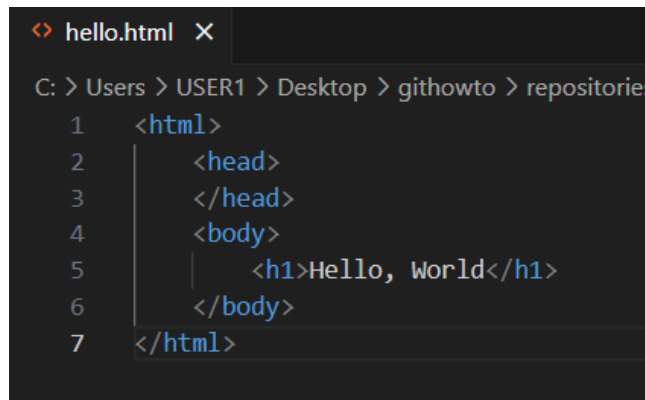
Чтобы вернуться к последней версии нашего кода, нам нужно переключиться на ветку по умолчанию, main. Для переключения между ветками можно воспользоваться командой switch.

```
git switch main
cat hello.html
```

```
Milka@Milka MINGW64 ~/Desktop/ghowto/repositories/work ((b7049be...))
$ git switch main
Previous HEAD position was b7049be Initial Commit
Switched to branch 'main'

Milka@Milka MINGW64 ~/Desktop/ghowto/repositories/work (main)
$ cat hello.html
<html>
  <head>
  </head>
  <body>
    <h1>Hello, world</h1>
  </body>
</html>
Milka@Milka MINGW64 ~/Desktop/ghowto/repositories/work (main)
$ |
```

Рисунок 26 – Возвращение к последней версии в ветке main

A screenshot of a code editor window titled 'hello.html'. The editor shows the following HTML code:

```
<html>
  <head>
</head>
  <body>
    <h1>Hello, World</h1>
  </body>
</html>
```

Рисунок 27 – Содержание файла hello.html

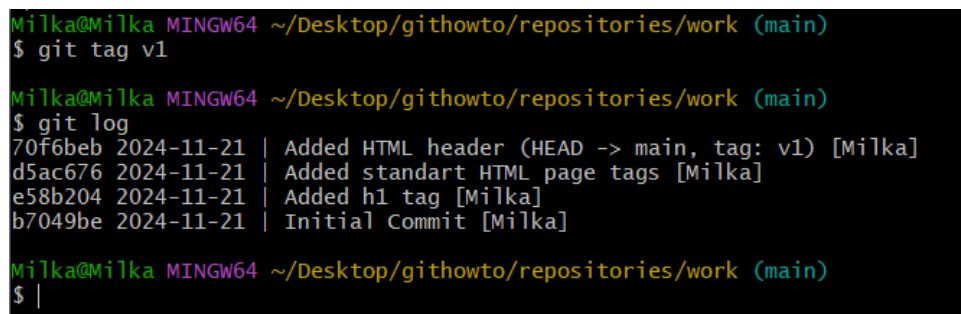
Здесь main — имя ветки по умолчанию. Переключаясь на ветку, вы попадаете на её последнюю версию.

Задание 11. Создание тегов версий

Думаю, вы согласитесь, что работать с хешами коммитов напрямую просто неудобно. Разве не было бы здорово, если бы вы могли обозначать конкретные коммиты понятными для человека названиями? Таким образом, вы могли бы четко видеть важные вехи в истории проекта. Кроме того, вы могли бы легко перейти к определенной версии проекта по ее названию. Именно для этого в Git придумали теги.

Давайте назовем текущую версию страницы hello.html первой, то есть v1. Создайте тег первой версии:

```
git tag v1
git log
```

A screenshot of a terminal window showing the execution of git commands. The prompt is 'Milka@Milka MINGW64 ~/Desktop/ghowto/repositories/work (main)'. The commands and their outputs are:

```
$ git tag v1

Milka@Milka MINGW64 ~/Desktop/ghowto/repositories/work (main)
$ git log
70f6beb 2024-11-21 | Added HTML header (HEAD -> main, tag: v1) [Milka]
d5ac676 2024-11-21 | Added standart HTML page tags [Milka]
e58b204 2024-11-21 | Added h1 tag [Milka]
b7049be 2024-11-21 | Initial Commit [Milka]

Milka@Milka MINGW64 ~/Desktop/ghowto/repositories/work (main)
$ |
```

Рисунок 28 – Создание тега первой версии

Теперь текущая версия страницы называется v1.

Обозначим версию, предшествующую текущей, названием v1-beta. Прежде всего, мы переключимся на предыдущую версию. Вместо того чтобы искать хеш коммита, мы будем использовать обозначение ^, а именно v1^, указывающее на коммит, предшествующий v1.

```
git checkout v1^
```

```
cat hello.html
```



```
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$ git checkout v1A
Note: switching to 'v1A'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

  git switch -c <new-branch-name>

Or undo this operation with:

  git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at d5ac676 Added standart HTML page tags
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work ((d5ac676...))
$ cat hello.html
<html>
  <body>
    <h1>Hello, world</h1>
  </body>
</html>
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work ((d5ac676...))
$
```

Рисунок 29 – Переключение на предыдущую версию

Это версия с тегами `<html>` и `<body>`, но еще пока без `<head>`. Давайте сделаем ее версией `v1-beta`.

```
git tag v1-beta
git log
```



```
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work ((d5ac676...))
$ git tag v1-beta

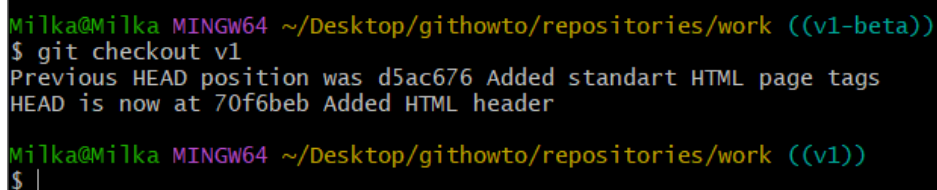
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work ((v1-beta))
$ git log
d5ac676 2024-11-21 | Added standart HTML page tags (HEAD, tag: v1-beta) [Milka]
e58b204 2024-11-21 | Added h1 tag [Milka]
b7049be 2024-11-21 | Initial Commit [Milka]

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work ((v1-beta))
$
```

Рисунок 30 – Обозначение версии `v1-beta`

Теперь попробуйте попереключаться между двумя отмеченными версиями.

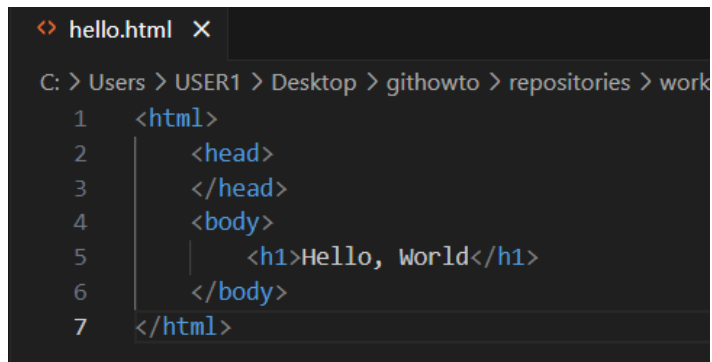
```
git checkout v1
git checkout v1-beta
```



```
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work ((v1-beta))
$ git checkout v1
Previous HEAD position was d5ac676 Added standart HTML page tags
HEAD is now at 70f6beb Added HTML header

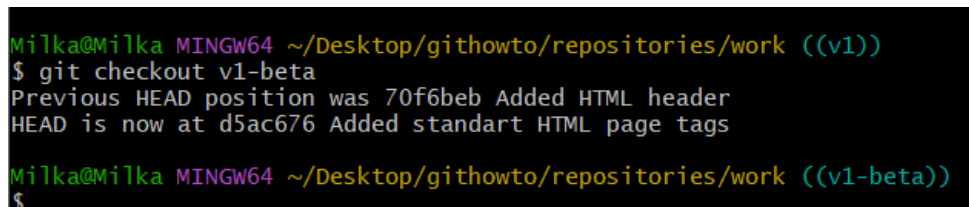
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work ((v1))
$ |
```

Рисунок 31 – Переключение на версию `v1`



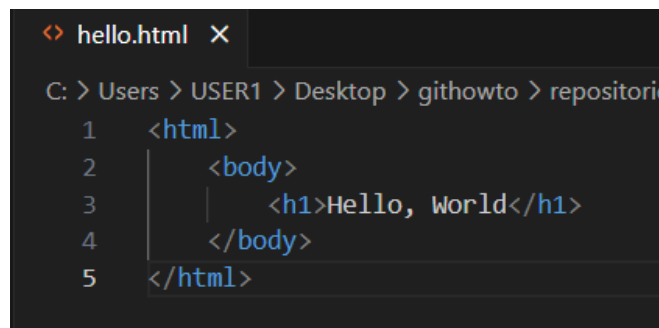
```
<> hello.html X
C: > Users > USER1 > Desktop > githowto > repositories > work
1  <html>
2    <head>
3    </head>
4    <body>
5      <h1>Hello, World</h1>
6    </body>
7  </html>
```

Рисунок 32 – Файл hello.html версии v1



```
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work ((v1))
$ git checkout v1-beta
Previous HEAD position was 70f6beb Added HTML header
HEAD is now at d5ac676 Added standart HTML page tags
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work ((v1-beta))
$
```

Рисунок 33 – Переключение на версию v1-beta

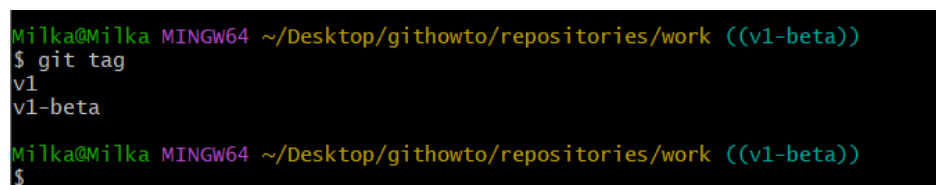


```
<> hello.html X
C: > Users > USER1 > Desktop > githowto > repositories
1  <html>
2    <body>
3      <h1>Hello, World</h1>
4    </body>
5  </html>
```

Рисунок 34 – Файл hello.html версии v1-beta

Вы можете увидеть, какие теги доступны, используя команду `git tag`.

```
git tag
```



```
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work ((v1-beta))
$ git tag
v1
v1-beta
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work ((v1-beta))
$
```

Рисунок 35 – Просмотр доступных тегов

Вы также можете посмотреть теги в логе.

```
git log main --all
```



```

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work ((v1-beta))
$ git log main --all
70f6beb 2024-11-21 | Added HTML header (tag: v1, main) [Milka]
d5ac676 2024-11-21 | Added standart HTML page tags (HEAD, tag: v1-beta) [Milka]
e58b204 2024-11-21 | Added h1 tag [Milka]
b7049be 2024-11-21 | Initial Commit [Milka]

```

Рисунок 36 – Просмотр тегов в логе

Вы можете видеть теги (v1 и v1-beta) в логе вместе с именем ветки (main). Кроме того, метка HEAD показывает коммит, на который вы переключились (на данный момент это v1-beta).

Задание 12. Отмена локальных изменений (до индексации)

Убедитесь, что вы находитесь на последнем коммите ветки main, прежде чем продолжить работу.

```
git switch main
```

```

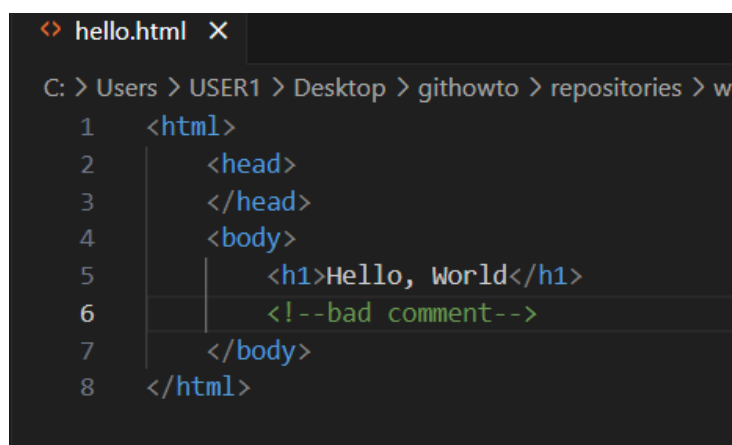
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work ((v1-beta))
$ git switch main
Previous HEAD position was d5ac676 Added standart HTML page tags
Switched to branch 'main'

```

Рисунок 37 – Проверка, что мы на последнем коммите ветки main

Иногда после того как вы изменили файл в рабочей директории, вы передумали и хотите просто вернуться к тому, что уже было закоммичено. Команда checkout справится с этой задачей.

Внесите изменение в файл hello.html в виде нежелательного комментария.



```

<> hello.html X
C: > Users > USER1 > Desktop > githowto > repositories > w
1  <html>
2      <head>
3      </head>
4      <body>
5          <h1>Hello, world</h1>
6          <!--bad comment-->
7      </body>
8  </html>

```

Рисунок 38 – Изменение файла hello.html

Сначала проверьте состояние рабочей директории.

```
git status
```

```

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$ git status
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   hello.html

no changes added to commit (use "git add" and/or "git commit -a")
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$

```

Рисунок 39 – Проверка состояния рабочей директории
Мы видим, что файл hello.html был изменен, но еще не проиндексирован.

Используйте команду checkout для переключения в версию файла hello.html в репозитории.

```

git checkout hello.html
git status
cat hello.html

```

```

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$ git checkout hello.html
Updated 1 path from the index

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$ git status
On branch main
nothing to commit, working tree clean

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$ cat hello.html
<html>
  <head>
  </head>
  <body>
    <h1>Hello, world</h1>
  </body>
</html>
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$

```

Рисунок 40 – Переключение в версию файла hello.html в репозитории

Команда status показывает нам, что в рабочей директории не было сделано никаких незафиксированных изменений. И «нежелательный комментарий» больше не является частью содержимого файла.

Задание 13. Отмена проиндексированных изменений (перед коммитом)

Внесите изменение в файл hello.html в виде нежелательного комментария:

```

<html>
  <head>
    <!-- This is an unwanted but staged comment -->
  </head>

```

```
<body>
  <h1>Hello, World</h1>
</body>
</html>
```

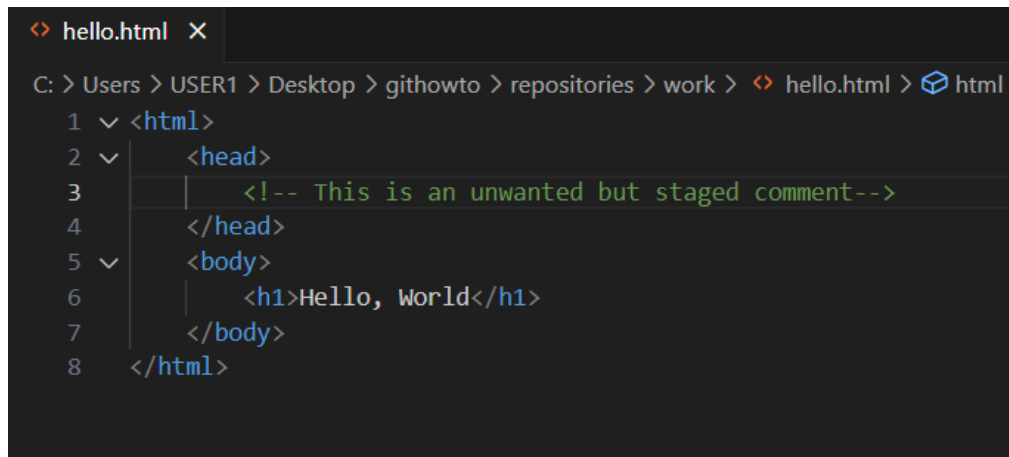


Рисунок 41 – Изменение файла hello.html

Проиндексируйте это изменение.

```
git add hello.html
```

Проверьте состояние нежелательного изменения.

```
git status
```

```
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$ git add hello.html

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   hello.html
```

Рисунок 42 – Индексирование изменения и проверка состояния

Состояние показывает, что изменение было проиндексировано и готово к коммиту.

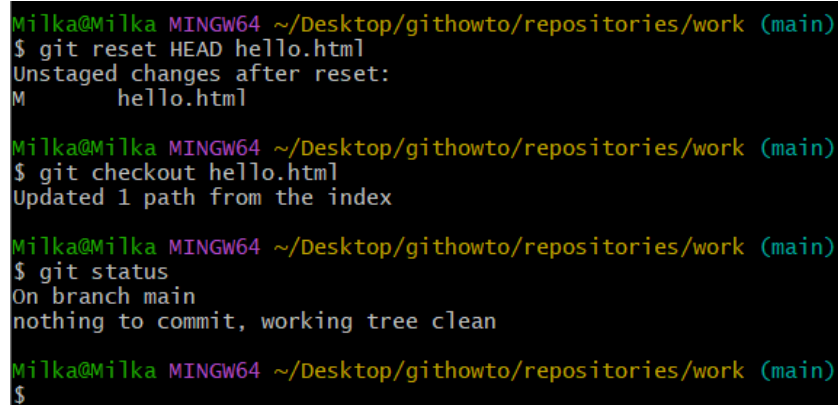
Команда `reset` сбрасывает область подготовки к HEAD. Это очищает область подготовки от изменений, которые мы только что проиндексировали.

```
git reset HEAD hello.html
```

Команда `reset` (по умолчанию) не изменяет рабочую директорию. Поэтому рабочая директория всё еще содержит нежелательный комментарий. Мы можем использовать команду `checkout` из предыдущего урока, чтобы убрать нежелательные изменения в рабочей директории.

Переключитесь на версию коммита:

```
git checkout hello.html
git status
```



```
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$ git reset HEAD hello.html
Unstaged changes after reset:
M   hello.html

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$ git checkout hello.html
Updated 1 path from the index

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$ git status
On branch main
nothing to commit, working tree clean

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$
```

Рисунок 43 – Сброс области подготовки и убирание нежелательного комментария

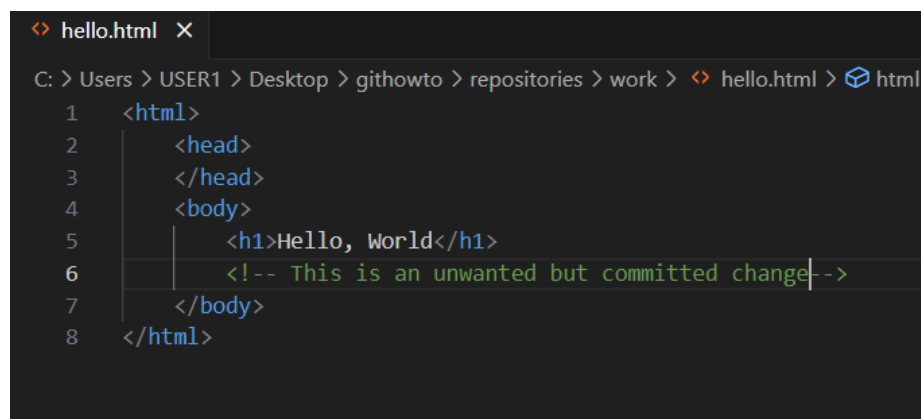
Наша рабочая директория опять чиста.

Задание 14. Отмена коммитов

Иногда вы понимаете, что новые коммиты являются неверными, и хотите их отменить. Есть несколько способов решения этого вопроса, здесь мы будем использовать самый безопасный.

Мы отменим коммит путем создания нового коммита, отменяющего нежелательные изменения.

Измените файл hello.html на следующий.



```
<> hello.html X
C: > Users > USER1 > Desktop > githowto > repositories > work > <> hello.html > html
1  <html>
2      <head>
3      </head>
4      <body>
5          <h1>Hello, World</h1>
6          <!-- This is an unwanted but committed change -->
7      </body>
8  </html>
```

Рисунок 44 – Изменение файла hello.html

Затем выполните команды:

```
git add hello.html
git commit -m "Oops, we didn't want this commit"
```

Чтобы отменить коммит, нам необходимо сделать коммит, который удаляет изменения, сохраненные нежелательным коммитом.

```
git revert HEAD
```

Перейдите в редактор, где вы можете отредактировать коммит-сообщение по умолчанию или оставить все как есть.

```
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$ git add hello.html

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$ git commit -m "Oops, we didn't want this commit"
[main 930a711] Oops, we didn't want this commit
1 file changed, 1 insertion(+)

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$ |
```

Рисунок 45 – Добавление нежелательного коммита

```
[main 4807475] Revert "Oops, we didn't want this commit"
1 file changed, 1 deletion(-)

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$ |
```

Рисунок 46 – Удаление нежелательного коммита

Так как мы отменили самый последний произведенный коммит, мы смогли использовать метку HEAD в качестве аргумента для отмены коммита. Мы можем отменить любой произвольный коммит в истории, указав его хеш.

Проверка лога показывает нежелательные и отмененные коммиты в наш репозиторий.

```
[main 4807475] Revert "Oops, we didn't want this commit"
1 file changed, 1 deletion(-)

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$ git log
4807475 2024-11-21 | Revert "Oops, we didn't want this commit" (HEAD -> main) [Milka]
930a711 2024-11-21 | Oops, we didn't want this commit [Milka]
70f6beb 2024-11-21 | Added HTML header (tag: v1) [Milka]
d5ac676 2024-11-21 | Added standart HTML page tags (tag: v1-beta) [Milka]
e58b204 2024-11-21 | Added h1 tag [Milka]
b7049be 2024-11-21 | Initial Commit [Milka]

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$ |
```

Рисунок 47 – Проверка лога

Эта техника будет работать с любым коммитом (хотя, возможно, возникнут конфликты). Она безопасна в использовании даже в публичных ветках удаленных репозиториях.

Далее давайте посмотрим на технику, которая может быть использована для удаления последних коммитов из истории репозитория.

Задание 15. Удаление коммитов из ветки

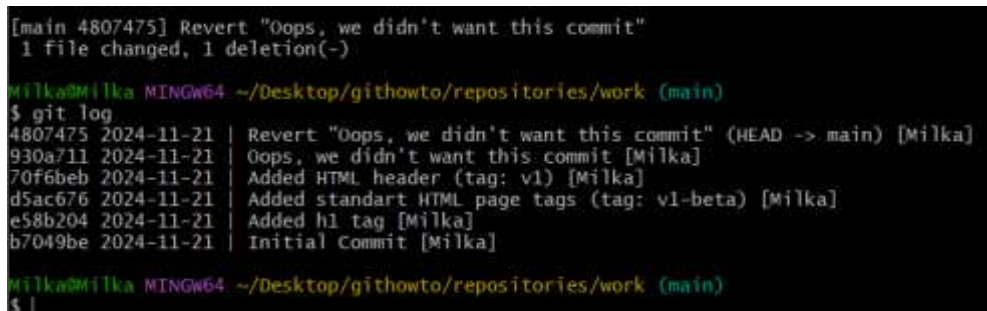
Мы уже видели команду `reset` и использовали ее для согласования области подготовки с выбранным коммитом (в предыдущем уроке мы использовали коммит HEAD).

Если выполнить команду `reset` с указанием ссылки на коммит (т.е. метки HEAD, имени ветки или тега, хеша коммита), то команда...

1. Изменит текущую ветку, чтобы она указывала на указанный коммит.
2. Опционально сбросит область подготовки до соответствия с указанным коммитом.
3. Опционально сбросит рабочую директорию до соответствия с указанным коммитом.

Давайте сделаем быструю проверку нашей истории коммитов.

```
git log
```



```
[main 4807475] Revert "Oops, we didn't want this commit"
1 file changed, 1 deletion(-)

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$ git log
4807475 2024-11-21 | Revert "Oops, we didn't want this commit" (HEAD -> main) [Milka]
930a711 2024-11-21 | Oops, we didn't want this commit [Milka]
70f6beb 2024-11-21 | Added HTML header (tag: v1) [Milka]
d5ac676 2024-11-21 | Added standart HTML page tags (tag: v1-beta) [Milka]
e58b204 2024-11-21 | Added h1 tag [Milka]
b7049be 2024-11-21 | Initial Commit [Milka]

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$ |
```

Рисунок 48 – Проверка истории коммитов

Мы видим, что два последних коммита в этой ветке - «Oops» и «Revert Oops». Давайте удалим их с помощью сброса.

Но прежде чем удалить коммиты, давайте отметим последний коммит тегом, чтобы потом можно было его найти.

```
git tag oops
```

Глядя на историю лога (см. выше), мы видим, что коммит с тегом `v1` является коммитом, предшествующим ошибочному коммиту. Давайте сбросим ветку до этой точки. Поскольку ветка имеет тег, мы можем использовать имя тега в команде сброса `reset` (если она не имеет тега, мы можем использовать хеш коммита).

```
git reset --hard v1
git log
```

```

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$ git tag oops

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$ git reset --hard v1
HEAD is now at 70f6beb Added HTML header

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$ git log
70f6beb 2024-11-21 | Added HTML header (HEAD -> main, tag: v1) [Milka]
d5ac676 2024-11-21 | Added standart HTML page tags (tag: v1-beta) [Milka]
e58b204 2024-11-21 | Added h1 tag [Milka]
b7049be 2024-11-21 | Initial Commit [Milka]

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$ |

```

Рисунок 49 – Сброс к предыдущему коммиту

Наша ветка main теперь указывает на коммит v1, а коммитов "Revert Oops" и "Oops" в ветке уже нет. Параметр --hard указывает, что рабочая директория должна быть приведена к тому состоянию, которое соответствует HEAD-коммиту ветки.

Что же случается с ошибочными коммитами? Оказывается, что коммиты все еще находятся в репозитории. На самом деле, мы все еще можем на них ссылаться. Помните, в начале этого урока мы создали для отмененного коммита тег oops? Давайте посмотрим на все коммиты.

```
git log --all
```

```

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$ git log --all
4807475 2024-11-21 | Revert "Oops, we didn't want this commit" (tag: oops) [Milka]
930a711 2024-11-21 | Oops, we didn't want this commit [Milka]
70f6beb 2024-11-21 | Added HTML header (HEAD -> main, tag: v1) [Milka]
d5ac676 2024-11-21 | Added standart HTML page tags (tag: v1-beta) [Milka]
e58b204 2024-11-21 | Added h1 tag [Milka]
b7049be 2024-11-21 | Initial Commit [Milka]

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$

```

Рисунок 50 – Просмотр всех коммитов

Сброс в локальных ветках, как правило, безопасен. Последствия любой «аварии» как правило, можно восстановить простым сбросом с помощью нужного коммита.

Однако, если ветка уже стала общедоступной на удаленных репозиториях, сброс может сбить с толку других пользователей ветки.

Задание 16. Удаление тега oops

Тег oops свою функцию выполнил, давайте удалим его. Это позволит внутреннему механизму Git убрать остаточные коммиты, на которые теперь не ссылаются никакие ветки или теги.

```
git tag -d oops
```

```
git log --all

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$ git tag -d oops
Deleted tag 'oops' (was 4807475)

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$ git log --all
70f6beb 2024-11-21 | Added HTML header (HEAD -> main, tag: v1) [Milka]
d5ac676 2024-11-21 | Added standart HTML page tags (tag: v1-beta) [Milka]
e58b204 2024-11-21 | Added h1 tag [Milka]
b7049be 2024-11-21 | Initial Commit [Milka]

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$
```

Рисунок 51 – Удаление тега oops

Тег oops больше не будет отображаться в репозитории.

Задание 17. Внесение изменений в коммиты

Добавьте в страницу комментарий автора.

```
<!-- Author: Alexander Shvets -->
<html>
  <head>
  </head>
  <body>
    <h1>Hello, World!</h1>
  </body>
</html>
```

```
2  ✓ <html>
3    <head>
4    </head>
5  ✓ <body>
6    <h1>Hello, World!</h1>
7    </body>
8  </html>|
```

Рисунок 52 – Изменение файла helo.html

Затем выполните:

```
git add hello.html
git commit -m "Added copyright statement"
git log
```



```

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$ git add hello.html

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$ git commit -m "Added copyright statement"
[main 0556de6] Added copyright statement
1 file changed, 2 insertions(+), 1 deletion(-)

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$ git log
0556de6 2024-11-21 | Added copyright statement (HEAD -> main) [Milka]
70f6beb 2024-11-21 | Added HTML header (tag: v1) [Milka]
d5ac676 2024-11-21 | Added standart HTML page tags (tag: v1-beta) [Milka]
e58b204 2024-11-21 | Added h1 tag [Milka]
b7049be 2024-11-21 | Initial Commit [Milka]

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$

```

Рисунок 53 – Создание коммита

Однако после создания коммита вы понимаете, что любой хороший комментарий должен включать электронную почту автора. Обновите страницу `hello.html`, включив в нее email.

```

<!-- Author: Alexander Shvets (alex@githowto.com) -->
<html>
  <head>
  </head>
  <body>
    <h1>Hello, World!</h1>
  </body>
</html>

```

```

2  <html>
3    <head>
4    </head>
5    <body>
6      <h1>Hello, World!</h1>
7    </body>
8  </html>

```

Рисунок 54 – Изменение файла `hello.html`

Мы действительно не хотим создавать отдельный коммит только ради электронной почты. Давайте изменим предыдущий коммит, включив в него адрес электронной почты.

```

git add hello.html
git commit --amend -m "Added copyright statement with email"

```

```

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$ git add hello.html

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$ git commit --amend -m "Added copyright statement with email"
[main fa4bfa0] Added copyright statement with email
Date: Thu Nov 21 19:29:15 2024 +0300
1 file changed, 2 insertions(+), 1 deletion(-)

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$

```

Рисунок 55 – Изменение коммита

Просмотр истории:

`git log`

```

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$ git log
fa4bfa0 2024-11-21 | Added copyright statement with email (HEAD -> main) [Milka]
70f6beb 2024-11-21 | Added HTML header (tag: v1) [Milka]
d5ac676 2024-11-21 | Added standart HTML page tags (tag: v1-beta) [Milka]
e58b204 2024-11-21 | Added h1 tag [Milka]
b7049be 2024-11-21 | Initial Commit [Milka]

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$

```

Рисунок 56 – Просмотр истории

Мы можем увидеть, что оригинальный коммит «автор» заменен коммитом «автор/email». Этого же эффекта можно достичь путем сброса последнего коммита в ветке, и повторного коммита новых изменений.

Задание 18. Создание ветки

Пришло время сделать нашу страницу более стильной с помощью CSS. Мы будем развивать эту возможность в новой ветке под названием style.

`git switch -c style`
`git status`

```

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$ git switch -c style
Switched to a new branch 'style'

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (style)
$ git status
On branch style
nothing to commit, working tree clean

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (style)
$

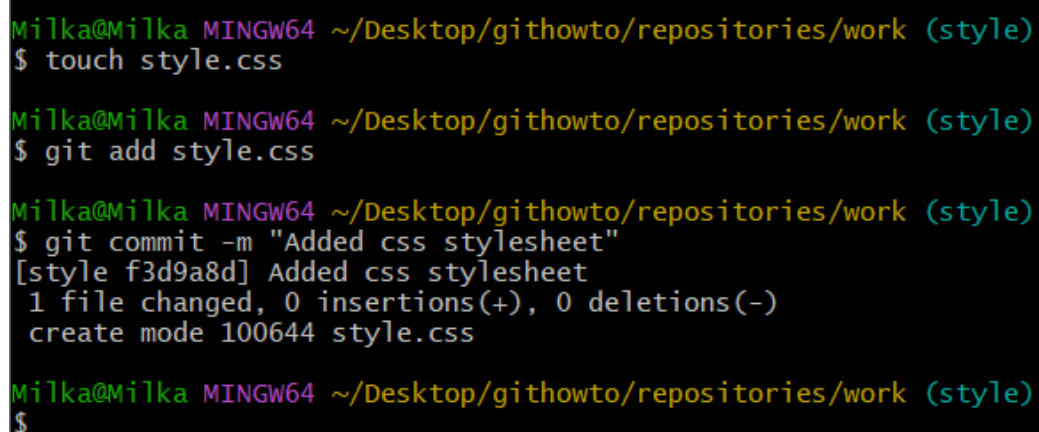
```

Рисунок 57 – Создание новой ветки style

Обратите внимание, что команда `git status` сообщает о том, что вы находитесь в ветке style.

Добавьте файл стилей style.css

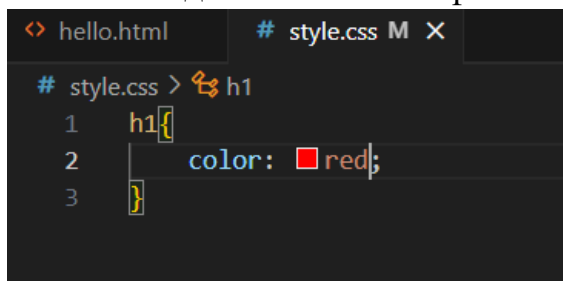
```
touch style.css  
  
h1 {  
    color: red;  
}  
  
git add style.css  
git commit -m "Added css stylesheet"
```



A terminal window with a black background and green text. The prompt is 'Milka@Milka MINGW64 ~/Desktop/ghithowto/repositories/work (style)'. The user enters '\$ touch style.css', '\$ git add style.css', and '\$ git commit -m "Added css stylesheet"'. The output shows the commit hash '[style f3d9a8d]', the message 'Added css stylesheet', and details: '1 file changed, 0 insertions(+), 0 deletions(-)' and 'create mode 100644 style.css'.

```
Milka@Milka MINGW64 ~/Desktop/ghithowto/repositories/work (style)  
$ touch style.css  
  
Milka@Milka MINGW64 ~/Desktop/ghithowto/repositories/work (style)  
$ git add style.css  
  
Milka@Milka MINGW64 ~/Desktop/ghithowto/repositories/work (style)  
$ git commit -m "Added css stylesheet"  
[style f3d9a8d] Added css stylesheet  
1 file changed, 0 insertions(+), 0 deletions(-)  
create mode 100644 style.css  
  
Milka@Milka MINGW64 ~/Desktop/ghithowto/repositories/work (style)  
$
```

Рисунок 58 – Создание и коммит файла style.css



A code editor window with a dark background. The title bar shows 'hello.html' and '# style.css M X'. The editor content shows the CSS file with line numbers 1, 2, and 3. Line 1 is 'h1{', line 2 is 'color: red;', and line 3 is '}'.

```
<> hello.html # style.css M X  
# style.css > h1  
1 h1{  
2   color: red;  
3 }
```

Рисунок 59 – Содержание файла style.css

Измените hello.html, чтобы он использовал style.css.

```
<!-- Author: Alexander Shvets (alex@ghithowto.com) -->  
<html>  
  <head>  
    <link type="text/css" rel="stylesheet" media="all" href="style.css" />  
  </head>  
  <body>  
    <h1>Hello, World!</h1>  
  </body>  
</html>  
  
git add hello.html  
git commit -m "Included stylesheet into hello.html"
```

```

2  ∨ <html>
3  ∨   <head>
4  |   <link type="text/css" rel="stylesheet" media="all" href="style.css">
5  |   </head>
6  ∨   <body>
7  |   <h1>Hello, World!</h1>
8  |   </body>
9  </html>

```

Рисунок 60 – Изменение файла hello.html

```

Milka@Milka MINGW64 ~/Desktop/ghowto/repositories/work (style)
$ git add hello.html

Milka@Milka MINGW64 ~/Desktop/ghowto/repositories/work (style)
$ git commit -m "Included stylesheet into hello.html"
[style b798b23] Included stylesheet into hello.html
1 file changed, 1 insertion(+)

Milka@Milka MINGW64 ~/Desktop/ghowto/repositories/work (style)
$

```

Рисунок 61 – Коммит изменений

Теперь у нас есть новая ветка под названием style с двумя новыми коммитами. Далее мы узнаем, как переключаться между ветками.

Задание 19. Переключение веток

Теперь в вашем проекте есть две ветки:

```
git log --all
```

```

Milka@Milka MINGW64 ~/Desktop/ghowto/repositories/work (style)
$ git log --all
b798b23 2024-11-21 | Included stylesheet into hello.html (HEAD -> style) [Milka]
f3d9a8d 2024-11-21 | Added css stylesheet [Milka]
fa4bfa0 2024-11-21 | Added copyright statement with email (main) [Milka]
70f6beb 2024-11-21 | Added HTML header (tag: v1) [Milka]
d5ac676 2024-11-21 | Added standart HTML page tags (tag: v1-beta) [Milka]
e58b204 2024-11-21 | Added h1 tag [Milka]
b7049be 2024-11-21 | Initial Commit [Milka]

Milka@Milka MINGW64 ~/Desktop/ghowto/repositories/work (style)
$

```

Рисунок 62 – Просмотр веток

Просто используйте команду `git switch` для переключения между ветками.

```
git switch main
cat hello.html
```

```

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (style)
$ git stash
Saved working directory and index state WIP on style: b798b23 Included stylesheet into hello.html

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (style)
$ git switch main
Switched to branch 'main'

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$ cat hello.html
<!--Author: Maslava Uliana (maslavauliana@gmail.com)-->
<html>
  <head>
  </head>
  <body>
    <h1>Hello, World!</h1>
  </body>
</html>
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$

```

Рисунок 63 – Переключение между ветками

```

2  <html>
3  |   <head>
4  |   </head>
5  |   <body>
6  |   |   <h1>Hello, World!</h1>
7  |   </body>
8  </html>

```

Рисунок 64 – Ветка main

Теперь мы находимся в ветке main. Как видите, в hello.html нет никаких следов style.css. Не волнуйтесь, эти изменения все еще есть в репозитории, но мы не можем увидеть их из ветки main. Вернемся к ветке style:

```

git switch style
cat hello.html

```

```

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$ git switch style
Switched to branch 'style'

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (style)
$ cat hello.html
<!--Author: Maslava Uliana (maslavauliana@gmail.com)-->
<html>
  <head>
    <link type="text/css" rel="stylesheet" media="all" href="style.css">
  </head>
  <body>
    <h1>Hello, World!</h1>
  </body>
</html>
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (style)
$

```

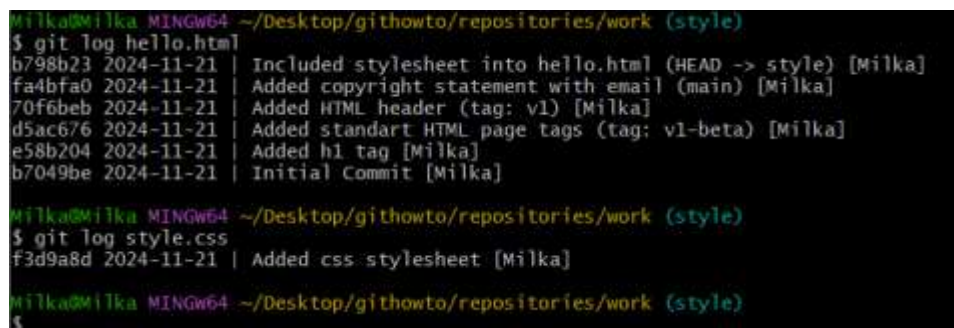
Рисунок 65 – Возвращение к ветке style

Мы вернулись к ветке `style`. Как видите, наши изменения, связанные с CSS, присутствуют.

Задание 20. Перемещение файлов

Git позволяет просматривать историю изменений конкретного файла. Давайте посмотрим историю изменений файла `hello.html` перед его переименованием.

```
git log hello.html
git log style.css
```



```
Milka@Milka MINGW64 ~/Desktop/ghowto/repositories/work (style)
$ git log hello.html
b798b23 2024-11-21 | Included stylesheet into hello.html (HEAD -> style) [Milka]
fa4bfa0 2024-11-21 | Added copyright statement with email (main) [Milka]
70f6beb 2024-11-21 | Added HTML header (tag: v1) [Milka]
d5ac676 2024-11-21 | Added standart HTML page tags (tag: v1-beta) [Milka]
e58b204 2024-11-21 | Added h1 tag [Milka]
b7049be 2024-11-21 | Initial Commit [Milka]

Milka@Milka MINGW64 ~/Desktop/ghowto/repositories/work (style)
$ git log style.css
f3d9a8d 2024-11-21 | Added css stylesheet [Milka]

Milka@Milka MINGW64 ~/Desktop/ghowto/repositories/work (style)
$
```

Рисунок 66 – Просмотр истории изменений в файлах `hello.html` и `style.css`

Команда `show` используется для просмотра изменений в конкретном коммите. Посмотрим изменения в файле `hello.html` в коммите, с тегом `v1` (можно использовать любую ссылку на коммит, например, метку `HEAD`, хеш коммита, имя ветки или тега и т.д.).

```
git show v1
```



```
Milka@Milka MINGW64 ~/Desktop/ghowto/repositories/work (style)
$ git show v1
70f6beb 2024-11-21 | Added HTML header (tag: v1) [Milka]

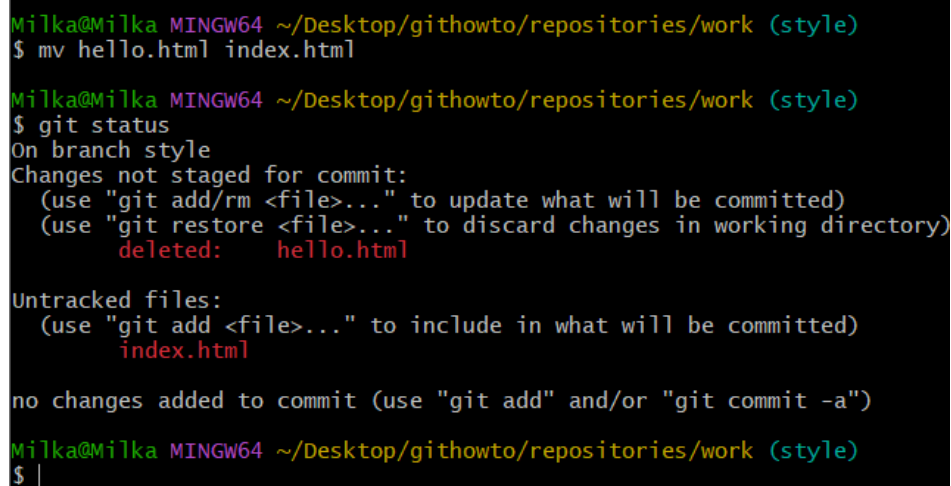
diff --git a/hello.html b/hello.html
index 710ce5e..1bbbd76 100644
--- a/hello.html
+++ b/hello.html
@@ -1,4 +1,6 @@
<html>
+ <head>
+ </head>
  <body>
    <h1>Hello, World</h1>
  </body>

Milka@Milka MINGW64 ~/Desktop/ghowto/repositories/work (style)
$
```

Рисунок 67 – Просмотр изменений в коммите

Давайте переименуем наш файл `hello.html` в `index.html` с помощью стандартной команды `mv` и посмотрим, что из этого получится.

```
mv hello.html index.html
git status
```



```
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (style)
$ mv hello.html index.html

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (style)
$ git status
On branch style
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    hello.html

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        index.html

no changes added to commit (use "git add" and/or "git commit -a")

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (style)
$ |
```

Рисунок 68 – Переименование файла hello.html в index.html

Git воспринимает наше изменение так, будто файл был удален и создан заново. Это тревожный звоночек. Нам нужно сообщить Git, что мы именно переименовали файл, а не удалили его и сразу создали новый. В простейшем случае Git сам поймёт, что файл был переименован, как только мы добавим его в индекс:

```
git add .
git status
```



```
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (style)
$ git add .

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (style)
$ git status
On branch style
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        renamed:   hello.html -> index.html

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (style)
$
```

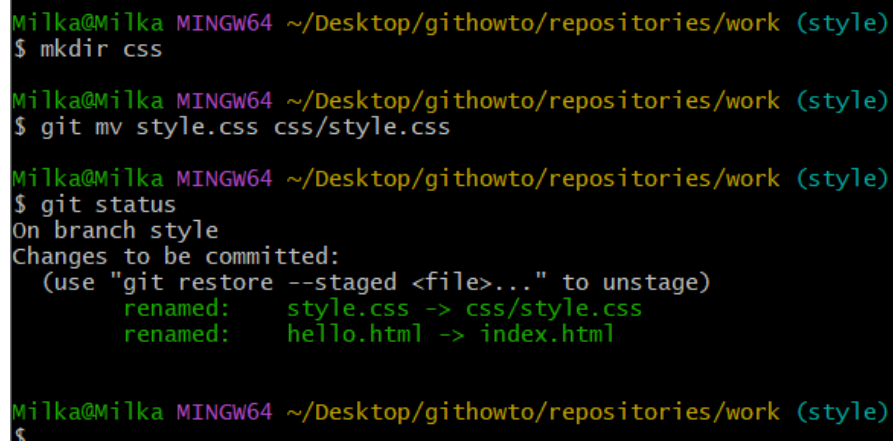
Рисунок 69 – Переименование файла в Git

В большинстве операционных систем переименование и перемещение файлов — это одно и то же. Итак, давайте переместим наш файл style.css в директорию css, но на этот раз сделаем это безопасно с помощью команды git mv. Эта команда гарантирует, что перемещение будет записано в истории Git как перемещение.

```
mkdir css
git mv style.css css/style.css
```



```
git status
```



```
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (style)
$ mkdir css

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (style)
$ git mv style.css css/style.css

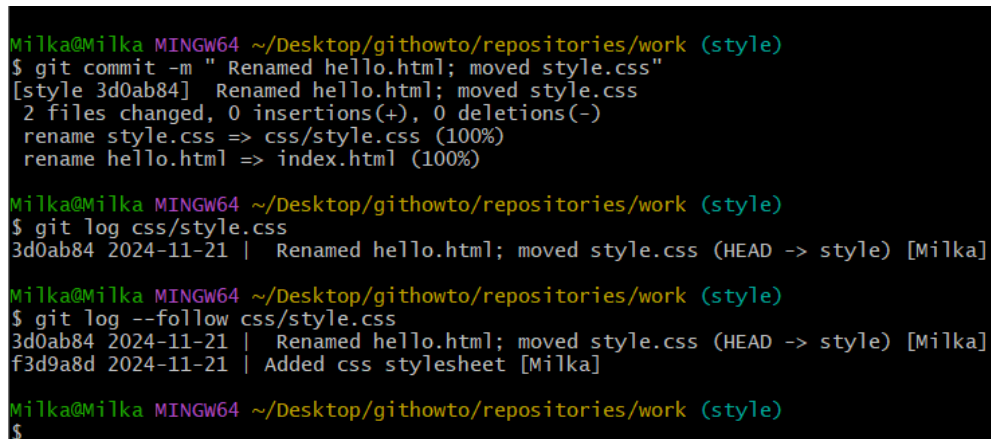
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (style)
$ git status
On branch style
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        renamed:    style.css -> css/style.css
        renamed:    hello.html -> index.html

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (style)
$
```

Рисунок 70 – Перемещение файла в Git

Давайте закомитим наши изменения и проверим историю изменений в файле `css/style.css`. Для просмотра истории файла до его перемещения нам потребуется добавить опцию `--follow`. Выполним оба варианта команды, чтобы понять разницу.

```
git commit -m "Renamed hello.html; moved style.css"
git log css/style.css
git log --follow css/style.css
```



```
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (style)
$ git commit -m "Renamed hello.html; moved style.css"
[style 3d0ab84] Renamed hello.html; moved style.css
2 files changed, 0 insertions(+), 0 deletions(-)
rename style.css => css/style.css (100%)
rename hello.html => index.html (100%)

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (style)
$ git log css/style.css
3d0ab84 2024-11-21 | Renamed hello.html; moved style.css (HEAD -> style) [Milka]

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (style)
$ git log --follow css/style.css
3d0ab84 2024-11-21 | Renamed hello.html; moved style.css (HEAD -> style) [Milka]
f3d9a8d 2024-11-21 | Added css stylesheet [Milka]

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (style)
$
```

Рисунок 71 – Коммит изменений

Задание 21. Изменения в ветке main

Как я уже говорил, Git позволяет работать с несколькими ветками одновременно. Это очень удобно при работе в команде, поскольку люди могут параллельно работать над разными функциями. Это также полезно при работе в одиночку: разрабатывая функции в отдельных ветках, вы можете исправлять ошибки и выпускать небольшие обновления, используя стабильный код в ветке `main`.

Создадим файл README. В нем будет рассказано о сути нашего проекта.

This is the Hello World example from the GitHowTo tutorial.

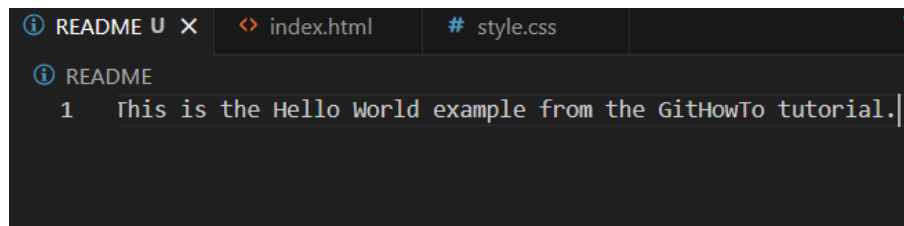


Рисунок 72 – Созданный файл README

Сделайте коммит файла README в ветку main. В настоящее время мы находимся в ветке style. Файл README не является частью этой ветки, поэтому перед коммитом мы должны переключиться на ветку main.

```
git switch main
git add README
git commit -m "Added README"
```

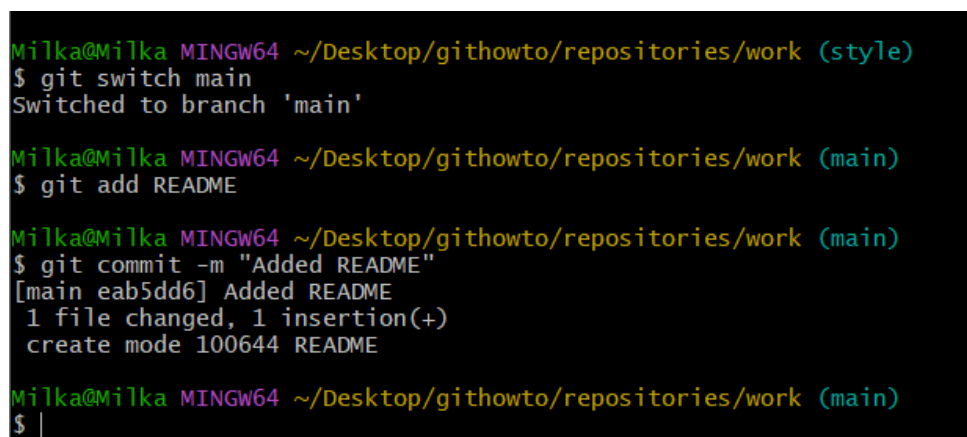


Рисунок 73 – Коммит файла README

Задание 22. Просмотр отличающихся веток

Теперь у нас есть две расходящиеся ветки в репозитории. Используйте следующую команду log для просмотра веток и их расхождения.

```
git log --all --graph
```

```

Milka@Milka MINGW64 ~/Desktop/ghowto/repositories/work (main)
$ git log --all graph
fatal: ambiguous argument 'graph': unknown revision or path not in the working tree.
Use '--' to separate paths from revisions, like this:
'git <command> [<revision>...] -- [<file>...]'

Milka@Milka MINGW64 ~/Desktop/ghowto/repositories/work (main)
$ git log --all --graph --oneline
* eab5dd6 (HEAD -> main) Added README
  * 3d0ab84 (style) Renamed hello.html; moved style.css
  * 719ae54 (refs/stash) WIP on style: b798b23 Included stylesheet into hello.html
  |
  | * 50bd3f0 index on style: b798b23 Included stylesheet into hello.html
  |
  | * b798b23 Included stylesheet into hello.html
  | * f3d9a8d Added css stylesheet
  |
  * fa4bfa0 Added copyright statement with email
  * 70f6beb (tag: v1) Added HTML header
  * d5ac676 (tag: v1-beta) Added standart HTML page tags
  * e58b204 Added h1 tag
  * b7049be Initial Commit

```

Рисунок 74 – Просмотр текущих веток

Опция `--all` гарантирует, что мы видим все ветки, так как по умолчанию в логе показывается только текущая ветка.

Опция `--graph` добавляет простое дерево коммитов, представленное в виде простых текстовых линий. Мы видим обе ветки (`style` и `main`) причём ветка `main` помечена как `HEAD`, что означает, что она является текущей. Общим предком для обеих веток является ветка, в которую был внесен коммит «Added copyright statement with email».

Задание 23. Слияние

Слияние переносит изменения из двух веток в одну. Давайте вернемся к ветке `style` и сольем `main` со `style`.

```

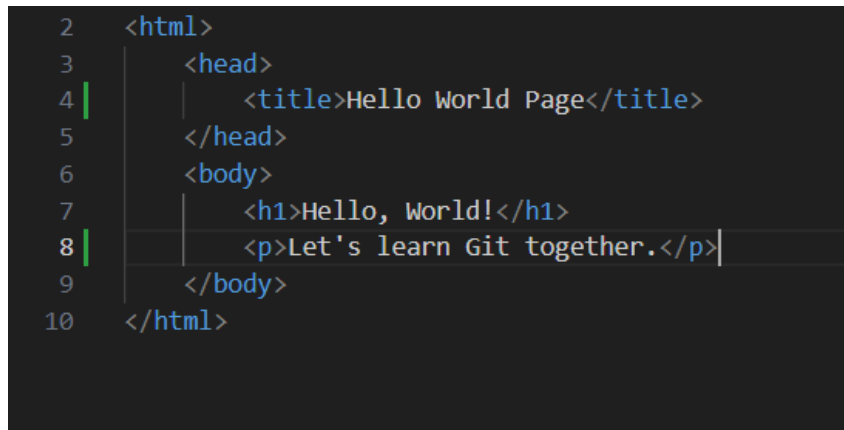
git switch style
git merge main
git log --all --graph

```



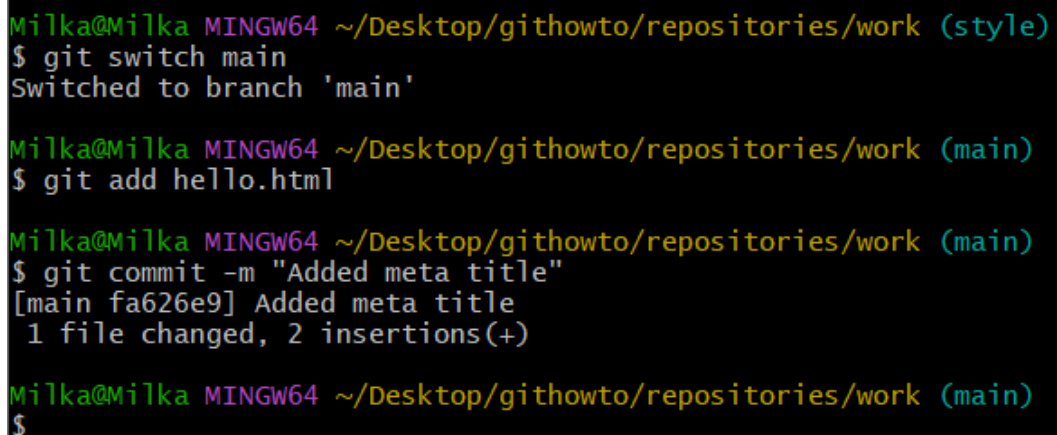
```
<html>
  <head>
    <title>Hello World Page</title>
  </head>
  <body>
    <h1>Hello, World!</h1>
    <p>Let's learn Git together.</p>
  </body>
</html>

git add hello.html
git commit -m "Added meta title"
```

A screenshot of a code editor with a dark background. The code is written in a light blue font. Line numbers 2 through 10 are visible on the left side of the editor. The code is an HTML document structure with a head section containing a title and a body section containing an h1 heading and a paragraph.

```
2  <html>
3    <head>
4      <title>Hello World Page</title>
5    </head>
6    <body>
7      <h1>Hello, World!</h1>
8      <p>Let's learn Git together.</p>
9    </body>
10  </html>
```

Рисунок 76 – Изменение файла hello.html

A screenshot of a terminal window with a black background and green and white text. The terminal shows a series of Git commands being executed in a directory. The output shows switching to the 'main' branch, adding the 'hello.html' file, and committing the changes with the message 'Added meta title'. The commit hash 'fa626e9' is shown, along with the summary '1 file changed, 2 insertions(+)'.

```
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (style)
$ git switch main
Switched to branch 'main'

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$ git add hello.html

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$ git commit -m "Added meta title"
[main fa626e9] Added meta title
1 file changed, 2 insertions(+)

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$
```

Рисунок 77 – Создание коммита

Просмотр веток:

```
git log --all --graph
```


Рисунок 79 – Возвращение в ветку style и слияние в нее всех изменений из ветки main

Похоже, что у нас возник конфликт. Ничего удивительного! Посмотрим, что скажет по этому поводу Git:

```
git status
```

```
Milka@Milka MINGW64 ~/Desktop/ghowto/repositories/work (style)
$ git merge main
Auto-merging index.html
CONFLICT (content): Merge conflict in index.html
Automatic merge failed; fix conflicts and then commit the result.

Milka@Milka MINGW64 ~/Desktop/ghowto/repositories/work (style|MERGING)
$ git status
On branch style
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)
        both modified:   index.html

no changes added to commit (use "git add" and/or "git commit -a")

Milka@Milka MINGW64 ~/Desktop/ghowto/repositories/work (style|MERGING)
$
```

Рисунок 80 – Просмотр состояния Git

Если открыть файл index.html, то можно увидеть:

```
2  <html>
3  <head>
4  <link type="text/css" rel="stylesheet" media="all" href="style.css">
5  <title>Hello World Page</title>
6  </head>
7  <body>
8  <h1>Hello, World!</h1>
9  <p>let's learn Git together.</p>
10 </body>
11 </html>
```

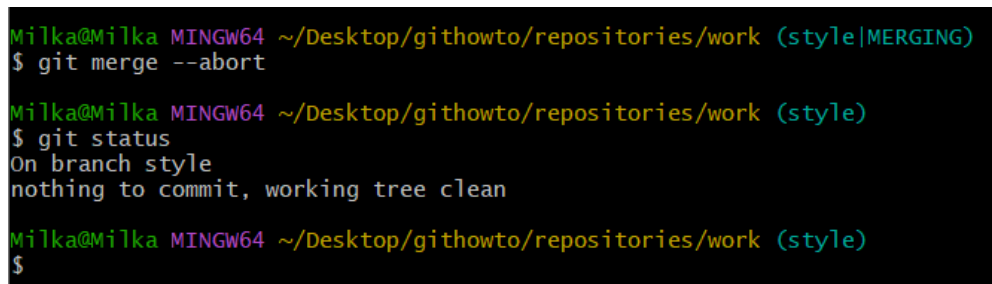
Рисунок 81 – Содержание файла index.html

Часть между <<<<<< >>>>>> является конфликтом. Верхняя часть соответствует ветке style, которая является текущей веткой (или HEAD) репозитория. Нижняя часть соответствует изменениям из ветки main. Git не может решить, какие изменения применить, поэтому он просит вас разрешить конфликт вручную. Вы можете оставить изменения из ветки style или из main, либо объединить их любым удобным способом. Вы также можете внести в файл любые другие изменения.

Кстати, вы заметили, что наше второе изменение, тег `<p>`, не является частью конфликта? Это потому, что Git сумел автоматически объединить ее.

Прежде чем мы приступим к разрешению нашего конфликта, хочу заметить, что сразу бросаться к разрешению конфликта не всегда оптимально. Конфликт может быть вызван изменениями, о которых вы не знаете. Или же изменения слишком велики, чтобы разрешить конфликт сразу. По этой причине Git позволяет прервать слияние и вернуться к предыдущему состоянию. Для этого можно воспользоваться командой `git merge --abort`, как это было предложено командой `status`, которую мы выполнили ранее.

```
git merge --abort
git status
```



```
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (style|MERGING)
$ git merge --abort

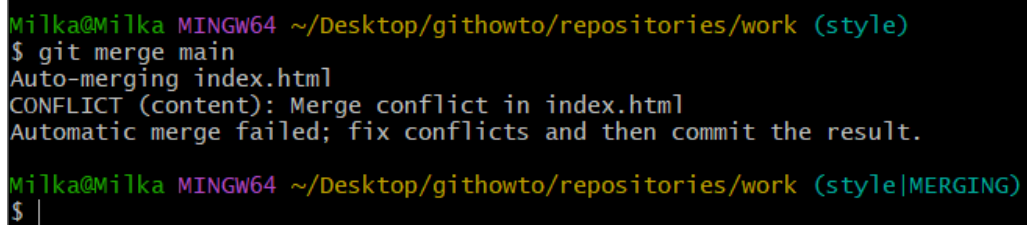
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (style)
$ git status
On branch style
nothing to commit, working tree clean

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (style)
$
```

Рисунок 82 – Отмена слияния и возврат к предыдущему состоянию

После небольшой медитации мы готовы к разрешению конфликта. Давайте снова запустим объединение.

```
git merge main
```



```
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (style)
$ git merge main
Auto-merging index.html
CONFLICT (content): Merge conflict in index.html
Automatic merge failed; fix conflicts and then commit the result.

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (style|MERGING)
$ |
```

Рисунок 83 – Конфликт веток

Чтобы разрешить конфликт, нужно отредактировать файл до состояния, которое нас устраивает, и затем закоммитить его как обычно. В нашем случае мы объединим изменения из обеих веток. Поэтому мы отредактируем файл до следующего состояния:

```
<!-- Author: Alexander Shvets (alex@githowto.com) -->
<html>
  <head>
    <title>Hello World Page</title>
    <link type="text/css" rel="stylesheet" media="all" href="style.css" />
  </head>
```

```

    <body>
      <h1>Hello, World!</h1>
      <p>Let's learn Git together.</p>
    </body>
  </html>

```

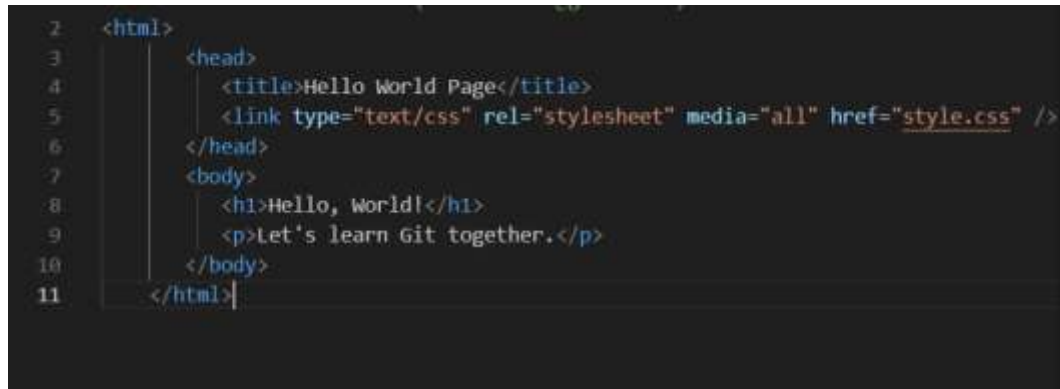


Рисунок 84 – Изменение файла index.html

Закоммитьте разрешенный конфликт:

```

git add index.html
git commit -m "Resolved merge conflict"
git status
git log --all --graph

```

```

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (style|MERGING)
$ git add index.html

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (style|MERGING)
$ git commit -m "Resolved merge conflict"
[style 2be5eb4] Resolved merge conflict

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (style)
$ git status
On branch style
nothing to commit, working tree clean

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (style)
$ git log --all --graph --oneline
* 2be5eb4 (HEAD -> style) Resolved merge conflict
|
| * fa626e9 (main) Added meta title
| * 4134aed Merge branch 'main' into style
|
| | * eab5dd6 Added README
| | * 3d0ab84 Renamed hello.html; moved style.css
| | * 719ae54 (refs/stash) WIP on style: b798b23 Included stylesheet into hello.html
| |
| | * 50bd3f0 index on style: b798b23 Included stylesheet into hello.html
| |
| | * b798b23 Included stylesheet into hello.html
| | * f3d9a8d Added css stylesheet
| |
| * fa4bfa0 Added copyright statement with email
| * 70f6beb (tag: v1) Added HTML header
| * d5ac676 (tag: v1-beta) Added standart HTML page tags
| * e58b204 Added h1 tag
| * b7049be Initial Commit

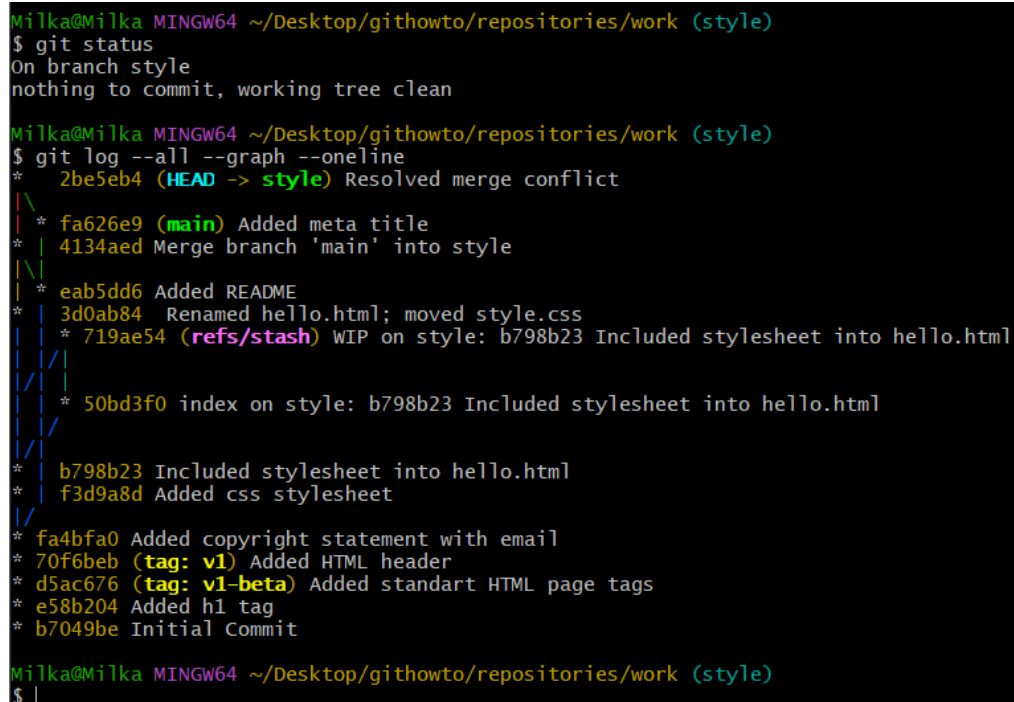
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (style)
$

```

Рисунок 85 – Разрешение конфликта

Давайте посмотрим на текущее состояние нашего хранилища и убедимся, что все в порядке:

```
git status
git log --all --graph
```



```
Milka@Milka MINGW64 ~/Desktop/ghowto/repositories/work (style)
$ git status
On branch style
nothing to commit, working tree clean

Milka@Milka MINGW64 ~/Desktop/ghowto/repositories/work (style)
$ git log --all --graph --oneline
* 2be5eb4 (HEAD -> style) Resolved merge conflict
|
| * fa626e9 (main) Added meta title
| * 4134aed Merge branch 'main' into style
|/
| * eab5dd6 Added README
| * 3d0ab84 Renamed hello.html; moved style.css
| * 719ae54 (refs/stash) WIP on style: b798b23 Included stylesheet into hello.html
|/
| * 50bd3f0 index on style: b798b23 Included stylesheet into hello.html
|/
| * b798b23 Included stylesheet into hello.html
| * f3d9a8d Added css stylesheet
|/
| * fa4bfa0 Added copyright statement with email
| * 70f6beb (tag: v1) Added HTML header
| * d5ac676 (tag: v1-beta) Added standart HTML page tags
| * e58b204 Added h1 tag
| * b7049be Initial Commit
$
```

Рисунок 86 – Проверка состояния Git

Задание 26. rebase против merge

Давайте рассмотрим различия между слиянием и перебазированием. Для того чтобы это сделать, нам нужно вернуться в репозиторий в момент до первого слияния, а затем повторить те же действия, но с использованием перебазирования вместо слияния.

Мы будем использовать команду `reset` для возврата веток к предыдущему состоянию.

Задание 27. Сброс ветки style

Давайте вернемся во времени на ветке `style` к точке перед тем, как мы слили ее с веткой `main`. Мы можем сбросить ветку к любому коммиту при помощи команды `reset`. По сути, это изменение указателя ветки на любую точку дерева коммитов.

В этом случае мы хотим вернуться в ветке `style` в точку перед слиянием с `main`. Нам необходимо найти последний коммит перед слиянием.

```
git switch style
git log --graph
```

```

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (style)
$ git log --graph
* 2be5eb4 2024-11-21 | Resolved merge conflict (HEAD -> style) [Milka]
|
| * fa626e9 2024-11-21 | Added meta title (main) [Milka]
| * 4134aed 2024-11-21 | Merge branch 'main' into style [Milka]
|/
| * eab5dd6 2024-11-21 | Added README [Milka]
| * 3d0ab84 2024-11-21 | Renamed hello.html; moved style.css [Milka]
| * b798b23 2024-11-21 | Included stylesheet into hello.html [Milka]
| * f3d9a8d 2024-11-21 | Added css stylesheet [Milka]
|/
* fa4bfa0 2024-11-21 | Added copyright statement with email [Milka]
* 70f6beb 2024-11-21 | Added HTML header (tag: v1) [Milka]
* d5ac676 2024-11-21 | Added standart HTML page tags (tag: v1-beta) [Milka]
* e58b204 2024-11-21 | Added h1 tag [Milka]
* b7049be 2024-11-21 | Initial Commit [Milka]

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (style)
$

```

Рисунок 87 – Возврат к ветке style

Это немного трудно читать, но, глядя на данные, мы видим, что коммит «Renamed hello.html; moved style.css» был последним на ветке style перед слиянием. Давайте сбросим ветку style к этому коммиту. Чтобы сослаться на этот коммит, мы либо используем его хеш, либо посчитаем, что этот коммит находится за 2 коммита до HEAD, то есть HEAD~2 в нотации Git.

```
git reset --hard HEAD~2
```

Теперь проверим историю ветки style. В истории не должно быть коммитов слияния.

```
git log --graph
```

```

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (style)
$ git reset --hard HEAD~2
HEAD is now at 3d0ab84 Renamed hello.html; moved style.css

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (style)
$ git log --graph
* 3d0ab84 2024-11-21 | Renamed hello.html; moved style.css (HEAD -> style) [Milka]
* b798b23 2024-11-21 | Included stylesheet into hello.html [Milka]
* f3d9a8d 2024-11-21 | Added css stylesheet [Milka]
* fa4bfa0 2024-11-21 | Added copyright statement with email [Milka]
* 70f6beb 2024-11-21 | Added HTML header (tag: v1) [Milka]
* d5ac676 2024-11-21 | Added standart HTML page tags (tag: v1-beta) [Milka]
* e58b204 2024-11-21 | Added h1 tag [Milka]
* b7049be 2024-11-21 | Initial Commit [Milka]

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (style)
$

```

Рисунок 88 – Проверка ветки style

Задание 28. Перебазирование

Мы вернули ветку style к точке перед первым слиянием. При этом в ветке main есть два коммита, которых нет в ветке style: новый файл README и конфликтующее изменение в файле index.html. На этот раз мы перенесем эти изменения в ветку style с помощью команды rebase, а не merge.

Перебазируем ветку style на ветку main:

```
git switch style
git rebase main
git status
```

```
Milka@Milka MINGW64 ~/Desktop/ghowto/repositories/work (style)
$ git rebase main
Auto-merging hello.html
CONFLICT (content): Merge conflict in hello.html
error: could not apply b798b23... Included stylesheet into hello.html
hint: Resolve all conflicts manually, mark them as resolved with
hint: "git add/rm <conflicted_files>", then run "git rebase --continue".
hint: You can instead skip this commit: run "git rebase --skip".
hint: To abort and get back to the state before "git rebase", run "git rebase --abort".
hint: Disable this message with "git config advice.mergeConflict false"
Could not apply b798b23... Included stylesheet into hello.html

Milka@Milka MINGW64 ~/Desktop/ghowto/repositories/work (style|REBASE 2/3)
$ git satus
git: 'satus' is not a git command. See 'git --help'.

The most similar command is
    status

Milka@Milka MINGW64 ~/Desktop/ghowto/repositories/work (style|REBASE 2/3)
$
```

Рисунок 89 – Перебазирование ветки style на ветку main

Опять возник конфликт! Обратите внимание, что конфликт возник в файле hello.html, а не в файле index.html, как в прошлый раз. Это связано с тем, что rebase находился в процессе применения изменений style поверх ветки main. Файл hello.html в main еще не был переименован, поэтому он все еще имеет старое имя.

При слиянии возник бы «обратный» конфликт. При слиянии изменения ветки main были бы применены поверх ветки style. В ветке style файл переименован, поэтому конфликт возник бы в файле index.html.

```
2  <html>
3  <head>
4  <==== HEAD (текущее изменение)
5  <title>Hello World Page</title>
6  <====
7  <link type="text/css" rel="stylesheet" media="all" href="style.css">
8  >>>>> b798b23 (Included stylesheet into hello.html) (ожидаемое изменение)
9  </head>
10 <body>
11 <h1>hello, world!</h1>
12 <p>Let's learn Git together.</p>
13 </body>
14 </html>
```

Рисунок 90 – Конфликт в файле hello.html

Сам конфликт может быть разрешен тем же способом, что и предыдущий. Сначала мы изменим файл hello.html, чтобы он соответствовал нашим ожиданиям.

```
<!-- Author: Alexander Shvets (alex@ghowto.com) -->
<html>
  <head>
    <title>Hello World Page</title>
```

```

    <link type="text/css" rel="stylesheet" media="all" href="style.css" />
</head>
<body>
    <h1>Hello, World!</h1>
    <p>Let's learn Git together.</p>
</body>
</html>

```

Рисунок 91 – Изменение файла hello.html

Но после этого нам не нужно коммитить изменения. Мы можем просто добавить файл в индекс и продолжить процесс rebase. Вот почему я люблю rebase! Он позволяет мне устранять конфликты, не создавая кучу уродливых конфликтов слияния.

```

git add .
git rebase --continue

```

Рисунок 92 – Выполнение rebase

Здесь, скорее всего, Git снова откроет редактор, чтобы позволить нам изменить текст коммита. Мы можем оставить текст без изменений. После сохранения изменений Git завершит процесс rebase, и мы сможем выполнить следующие команды:

```

git status
git log --all --graph

```

```

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (style)
$ git status
On branch style
nothing to commit, working tree clean

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (style)
$ git log --all --graph --oneline
* 62ed61c (HEAD -> style) Renamed hello.html; moved style.css
* 618244c Included stylesheet into hello.html
* 55e9c72 Added css stylesheet
* fa626e9 (main) Added meta title
* eab5dd6 Added README
| * 719ae54 (refs/stash) WIP on style: b798b23 Included stylesheet into hello.html
| | \
| | * 50bd3f0 index on style: b798b23 Included stylesheet into hello.html
| | /
| * b798b23 Included stylesheet into hello.html
| * f3d9a8d Added css stylesheet
| /
* fa4bfa0 Added copyright statement with email
* 70f6beb (tag: v1) Added HTML header
* d5ac676 (tag: v1-beta) Added standart HTML page tags
* e58b204 Added h1 tag
* b7049be Initial Commit

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (style)
$

```

Рисунок 92 – Заверение rebase и просмотр состояния Git

Конечный результат перебазирувания очень похож на результат слияния. Ветка style в настоящее время содержит все свои изменения, а также все изменения ветки main. Однако, дерево коммитов значительно отличается. Дерево коммитов ветки style было переписано таким образом, что ветка main является частью истории коммитов. Это делает цепь коммитов линейной и гораздо более читабельной.

Используйте команду rebase:

- Когда вы получаете изменения из удаленного репозитория и хотите применить их к своей локальной ветке.
- Если вы хотите, чтобы история коммитов была линейной и легко читаемой.

Не используйте команду rebase:

- Если текущая ветка является публичной и общей. Переписывание таких веток будет мешать работе других членов команды.
- Если важна точная история ветки коммитов (поскольку команда rebase переписывает историю коммитов).

Учитывая приведенные выше рекомендации, я предпочитаю использовать команду rebase для краткосрочных, локальных веток и команду merge для веток в публичном репозитории.

Задание 29. Слияние в ветку main

Мы поддерживали соответствие ветки style с веткой main (с помощью rebase), теперь давайте сольем изменения style в ветку main.

```
git switch main
git merge style
```

```
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (style)
$ git switch main
Switched to branch 'main'

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$ git merge style
Updating fa626e9..62ed61c
Fast-forward
 css/style.css | 0
 hello.html    | 10 -----
 index.html    | 11 ++++++++
 3 files changed, 11 insertions(+), 10 deletions(-)
 create mode 100644 css/style.css
 delete mode 100644 hello.html
 create mode 100644 index.html

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$
```

Рисунок 93 – Слияние ветки style в main

Поскольку последний коммит в main предшествует последнему коммиту ветки style, Git может выполнить ускоренное слияние, просто переместив указатель ветки вперед, на тот же коммит, что и ветка style.

При ускоренном слиянии конфликты не возникают. Кроме того, при ускоренном слиянии не создается фиксация слияния.

Просмотрите логи:

```
git log --all --graph
```

```
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$ git log --all --graph
* 62ed61c 2024-11-21 | Renamed hello.html; moved style.css (HEAD -> main, style) [Milka]
* 618244c 2024-11-21 | Included stylesheet into hello.html [Milka]
* 55e9c72 2024-11-21 | Added css stylesheet [Milka]
* fa626e9 2024-11-21 | Added meta title [Milka]
* eab5dd6 2024-11-21 | Added README [Milka]
| * 719ae54 2024-11-21 | WIP on style: b798b23 Included stylesheet into hello.html (refs/stash) [Milka]
| |
| | * 50bd3f0 2024-11-21 | index on style: b798b23 Included stylesheet into hello.html [Milka]
| | |
| | * b798b23 2024-11-21 | Included stylesheet into hello.html [Milka]
| | * f3d9a8d 2024-11-21 | Added css stylesheet [Milka]
| |
| * fa4bfa0 2024-11-21 | Added copyright statement with email [Milka]
| * 70f6beb 2024-11-21 | Added HTML header (tag: v1) [Milka]
| * d5ac676 2024-11-21 | Added standart HTML page tags (tag: v1-beta) [Milka]
| * e58b204 2024-11-21 | Added h1 tag [Milka]
| * b7049be 2024-11-21 | Initial Commit [Milka]
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$
```

Рисунок 94 – Просмотр логов

Теперь ветки style и main идентичны.

Часть II: Несколько репозиториев

Задание 30. Клонирование репозиториев

Если вы работаете в команде, последующие 8 уроков довольно важны в понимании, т.к. вы почти всегда будете работать с клонированными репозиториями.

Перейдите в директорию repositories:

```
cd ..  
pwd  
ls
```



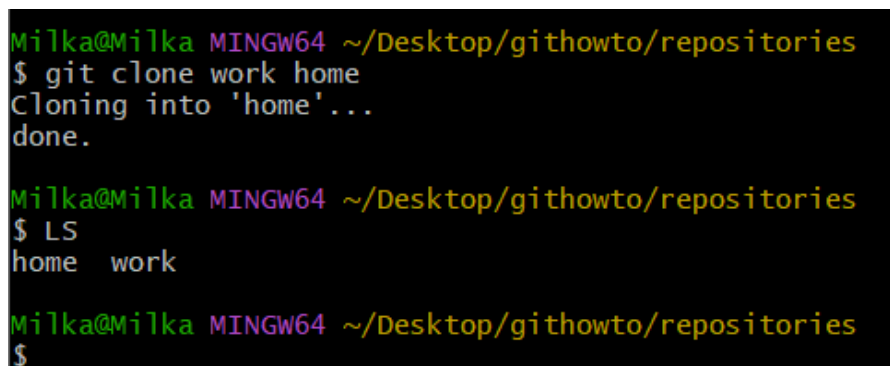
```
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)  
$ cd ..  
  
Milka@Milka MINGW64 ~/Desktop/githowto/repositories  
$ pwd  
/c/Users/USER1/Desktop/githowto/repositories  
  
Milka@Milka MINGW64 ~/Desktop/githowto/repositories  
$ ls  
work  
  
Milka@Milka MINGW64 ~/Desktop/githowto/repositories  
$
```

Рисунок 95 – Переход в директорию repositories

В этот момент вы должны находиться в директории repositories. Здесь должен быть единственный репозиторий под названием work.

Давайте создадим клон репозитория.

```
git clone work home  
ls
```



```
Milka@Milka MINGW64 ~/Desktop/githowto/repositories  
$ git clone work home  
Cloning into 'home'...  
done.  
  
Milka@Milka MINGW64 ~/Desktop/githowto/repositories  
$ ls  
home work  
  
Milka@Milka MINGW64 ~/Desktop/githowto/repositories  
$
```

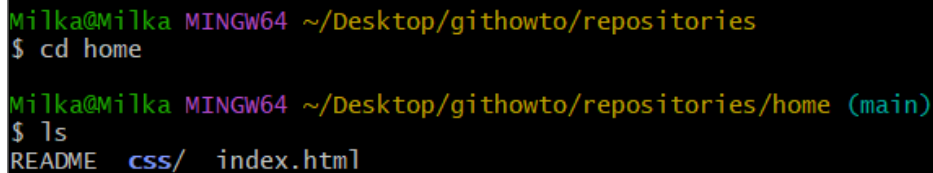
Рисунок 96 – Создание клона репозитория work

В вашем списке репозиториев теперь должно быть два репозитория: оригинальный репозиторий work и клонированный репозиторий home.

Задание 31. Просмотр клонированного репозитория

Давайте взглянем на клонированный репозиторий

```
cd home  
ls
```



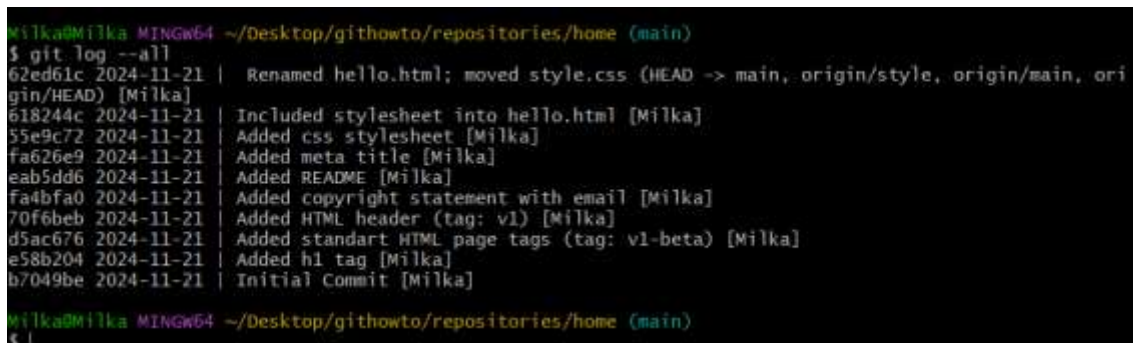
```
Milka@Milka MINGW64 ~/Desktop/githowto/repositories  
$ cd home  
  
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/home (main)  
$ ls  
README  css/  index.html
```

Рисунок 97 – Просмотр клонированного репозитория

Вы увидите список всех файлов на верхнем уровне оригинального репозитория README, index.html и css).

Просмотрите историю репозитория:

```
git log --all
```



```
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/home (main)  
$ git log --all  
62ed61c 2024-11-21 | Renamed hello.html; moved style.css (HEAD -> main, origin/style, origin/main, ori  
gin/HEAD) [Milka]  
618244c 2024-11-21 | Included stylesheet into hello.html [Milka]  
55e9c72 2024-11-21 | Added css stylesheet [Milka]  
fa626e9 2024-11-21 | Added meta title [Milka]  
eab5dd6 2024-11-21 | Added README [Milka]  
fa4bfa0 2024-11-21 | Added copyright statement with email [Milka]  
70f6beb 2024-11-21 | Added HTML header (tag: v1) [Milka]  
d5ac676 2024-11-21 | Added standart HTML page tags (tag: v1-beta) [Milka]  
e58b204 2024-11-21 | Added h1 tag [Milka]  
b7049be 2024-11-21 | Initial Commit [Milka]  
  
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/home (main)  
$
```

Рисунок 98 – Просмотр истории клонированного репозитория

Вы увидите список всех коммитов в новый репозиторий, и он должен совпадать с историей коммитов в оригинальном репозитории. Единственная разница должна быть в названиях веток.

Вы увидите ветку main (HEAD) в списке истории. Вы также увидите ветки со странными именами (origin/main, origin/style и origin/HEAD). Мы поговорим о них чуть позже.

Задание 32. Что такое origin?

Цель – узнать об именах удаленных репозиториях.

```
git remote
```



```
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/home (main)
$ git remote
origin
```

Рисунок 99 – Просмотр удаленного репозитория

Мы видим, что клонированный репозиторий знает об имени по умолчанию удаленного репозитория. Давайте посмотрим, можем ли мы получить более подробную информацию об имени по умолчанию:

```
git remote show origin
```

```
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/home (main)
$ git remote show origin
* remote origin
Fetch URL: C:/Users/USER1/Desktop/githowto/repositories/work
Push URL: C:/Users/USER1/Desktop/githowto/repositories/work
HEAD branch: main
Remote branches:
  main tracked
  style tracked
Local branch configured for 'git pull':
  main merges with remote main
Local ref configured for 'git push':
  main pushes to main (up to date)
```

Рисунок 100 – Просмотр информации о репозитории

Мы видим, что имя по умолчанию (origin) удаленного репозитория — изначально work. Удаленные репозитории обычно размещаются на отдельной машине, возможно, централизованном сервере. Однако, как мы видим здесь, они могут с тем же успехом указывать на репозиторий на той же машине. Нет ничего особенного в имени origin, однако существует традиция использовать origin в качестве имени первичного централизованного репозитория (если таковой имеется).

Задание 33. Удаленные ветки

Давайте посмотрим на ветки, доступные в нашем клонированном репозитории.

```
git branch
```

```
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/home (main)
$ git branch
* main

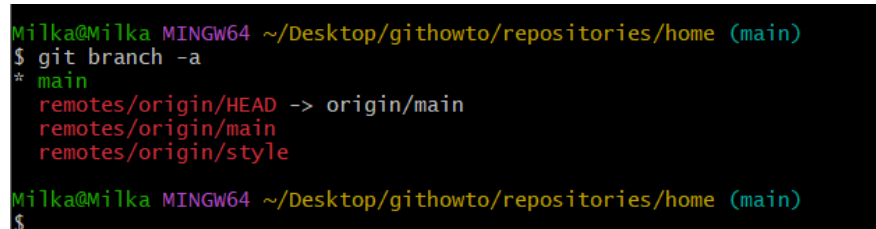
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/home (main)
$ |
```

Рисунок 101 – Просмотр доступных веток

Как мы видим, в списке только ветка main. Где ветка style? Команда `git branch` выводит только список локальных веток по умолчанию.

Для того чтобы увидеть все ветки, попробуйте следующую команду:

```
git branch -a
```



```
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/home (main)
$ git branch -a
* main
remotes/origin/HEAD -> origin/main
remotes/origin/main
remotes/origin/style
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/home (main)
$
```

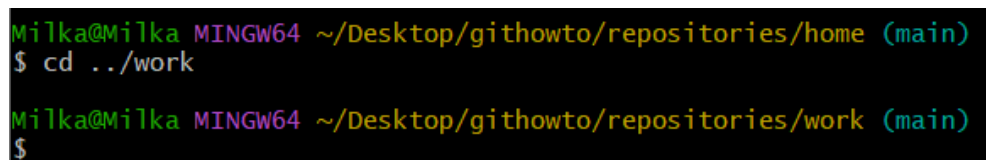
Рисунок 102 – Просмотр всех веток

Git выводит все коммиты в оригинальный репозиторий, но ветки в удаленном репозитории не рассматриваются как локальные. Если мы хотим иметь собственную ветку style, мы должны сами ее создать. Через минуту вы увидите, как это делается.

Задание 34. Изменение оригинального репозитория

Внесите изменения в оригинальный репозиторий work:

```
cd ../work
```

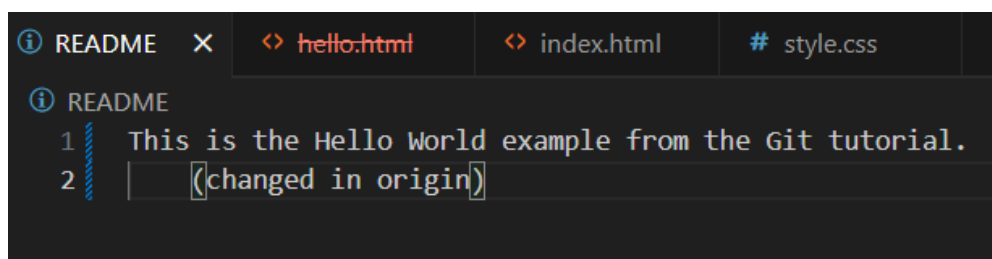


```
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/home (main)
$ cd ../work
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$
```

Рисунок 103 – Переход в репозиторий work

Внесите следующие изменения в файл README:

```
This is the Hello World example from the Git tutorial.
(changed in origin)
```

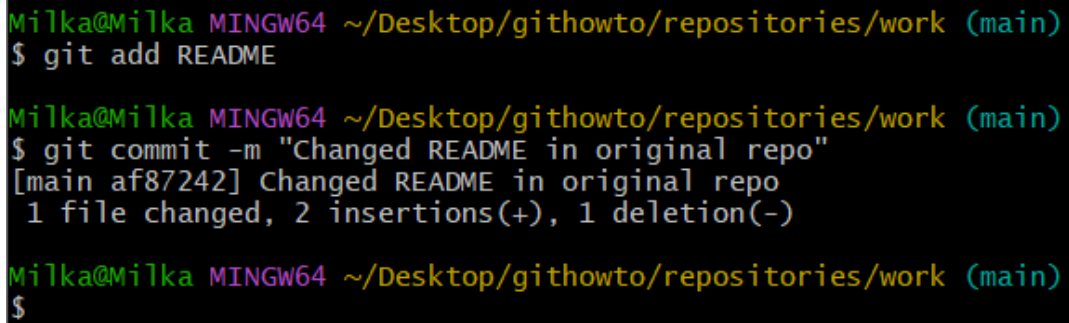


```
README x hello.html index.html style.css
1 This is the Hello World example from the Git tutorial.
2 (changed in origin)
```

Рисунок 104 – Изменение файла README

Теперь добавьте это изменение и сделайте коммит:

```
git add README
git commit -m "Changed README in original repo"
```



```
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$ git add README

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$ git commit -m "Changed README in original repo"
[main af87242] Changed README in original repo
1 file changed, 2 insertions(+), 1 deletion(-)

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$
```

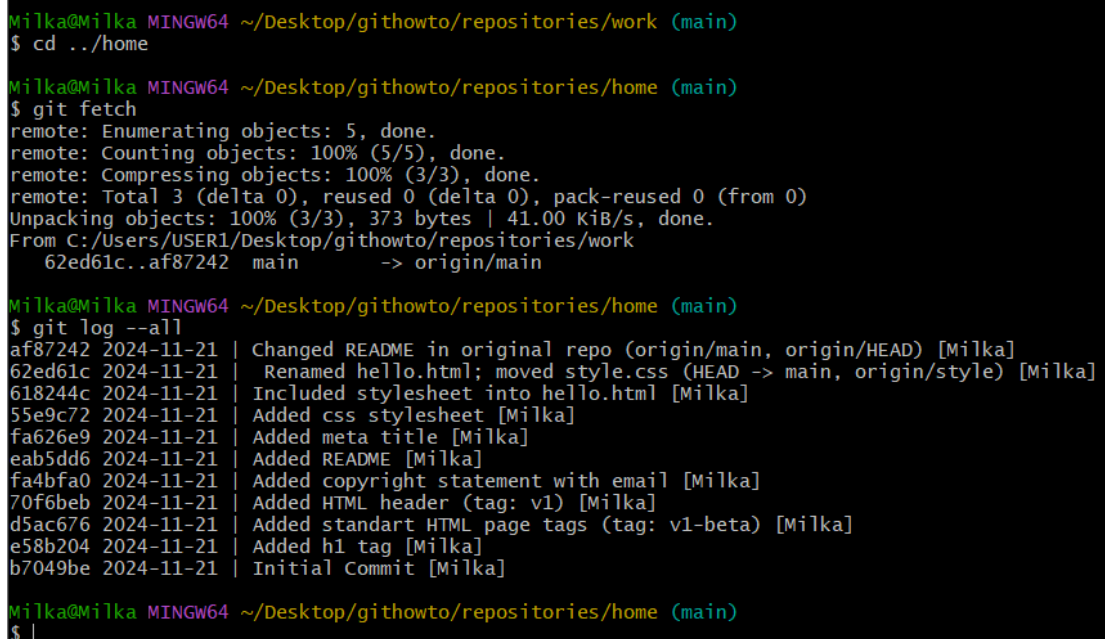
Рисунок 105 – Добавление изменения в Git и коммит

Теперь в оригинальной репозитории есть более поздние изменения, которых нет в клонированной версии. Далее мы подтянем и сольем эти изменения в клонированный репозиторий.

Задание 35. Подтягивание изменений

Цель – научиться подтягивать изменения из удаленного репозитория.

```
cd ../home
git fetch
git log --all
```



```
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$ cd ../home

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/home (main)
$ git fetch
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 373 bytes | 41.00 KiB/s, done.
From C:/Users/USER1/Desktop/githowto/repositories/work
 62ed61c..af87242  main       -> origin/main

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/home (main)
$ git log --all
af87242 2024-11-21 | Changed README in original repo (origin/main, origin/HEAD) [Milka]
62ed61c 2024-11-21 | Renamed hello.html; moved style.css (HEAD -> main, origin/style) [Milka]
618244c 2024-11-21 | Included stylesheet into hello.html [Milka]
55e9c72 2024-11-21 | Added css stylesheet [Milka]
fa626e9 2024-11-21 | Added meta title [Milka]
eab5dd6 2024-11-21 | Added README [Milka]
fa4bfa0 2024-11-21 | Added copyright statement with email [Milka]
70f6beb 2024-11-21 | Added HTML header (tag: v1) [Milka]
d5ac676 2024-11-21 | Added standart HTML page tags (tag: v1-beta) [Milka]
e58b204 2024-11-21 | Added h1 tag [Milka]
b7049be 2024-11-21 | Initial Commit [Milka]

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/home (main)
$
```

Рисунок 106 – Подтягивание изменений из удаленного репозитория

На данный момент в репозитории есть все коммиты из оригинального репозитория, но они не интегрированы в локальные ветки клонированного репозитория.

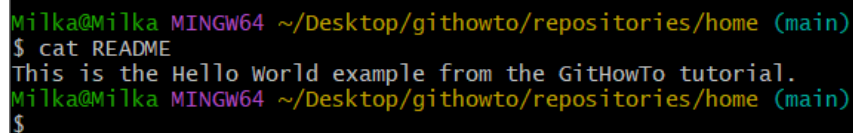
В истории выше найдите коммит «Changed README in original repo». Обратите внимание, что коммит включает в себя коммиты origin/main и origin/HEAD.

Теперь давайте посмотрим на коммит «Renamed hello.html; moved style.css». Вы увидите, что локальная ветка main указывает на этот коммит, а не на новый коммит, который мы только что подтянули.

Выводом является то, что команда git fetch будет подтягивать новые коммиты из удаленного репозитория, но не будет сливать их с вашими наработками в локальных ветках.

Мы можем продемонстрировать, что клонированный файл README не изменился.

```
cat README
```



```
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/home (main)
$ cat README
This is the Hello World example from the GitHowTo tutorial.
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/home (main)
$
```

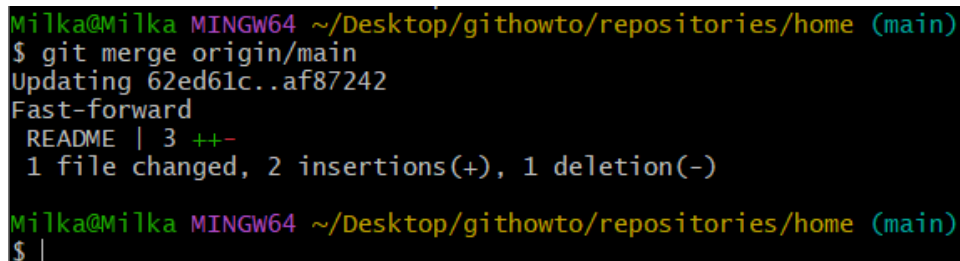
Рисунок 107 – Просмотр README

Как видите, никаких изменений.

Задание 36. Слияние подтянутых изменений

Слейте подтянутые изменения в локальную ветку main.

```
git merge origin/main
```



```
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/home (main)
$ git merge origin/main
Updating 62ed61c..af87242
Fast-forward
 README | 3 ++-
 1 file changed, 2 insertions(+), 1 deletion(-)
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/home (main)
$ |
```

Рисунок 108 – Слияние подтянутых изменений в локальную ветку main

Еще раз проверьте файл README. Сейчас мы должны увидеть изменения.

```
cat README
```

```

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/home (main)
$ cat README
This is the Hello World example from the Git tutorial.
(changed in origin)
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/home (main)
$ |

```

Рисунок 109 – Проверка файла README

Вот и изменения. Хотя команда `git fetch` не сливает изменения, мы можем вручную слить изменения из удаленного репозитория.

Команда `fetch` позволяет контролировать то, что именно подтягивается и сливается в ваши локальные ветки, но для удобства существует также команда `pull`, которая подтягивает и сливает изменения из удаленной ветки в текущую одним вызовом.

```
git pull
```

...эквивалентна следующим двум шагам:

```
git fetch
git merge origin/main
```

Задание 37. Добавление ветки наблюдения

Ветки, которые начинаются с `remotes/origin` являются ветками оригинального репозитория. Обратите внимание, что у вас больше нет ветки под названием `style`, но система контроля версий знает, что в оригинальном репозитории ветка `style` была.

Добавьте локальную ветку, которая отслеживает удаленную ветку:

```
git branch --track style origin/style
git branch -a
git log --max-count=2
```

```

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/home (main)
$ git branch --track style origin/style
branch 'style' set up to track 'origin/style'.

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/home (main)
$ git branch -a
* main
  style
remotes/origin/HEAD -> origin/main
remotes/origin/main
remotes/origin/style

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/home (main)
$ git log --max-count=2
af87242 2024-11-21 | Changed README in original repo (HEAD -> main, origin/main, origin/HEAD) [Milka]
62ed61c 2024-11-21 | Renamed hello.html; moved style.css (origin/style, style) [Milka]

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/home (main)
$

```

Рисунок 110 – Добавление локальной ветки для отслеживания удаленной

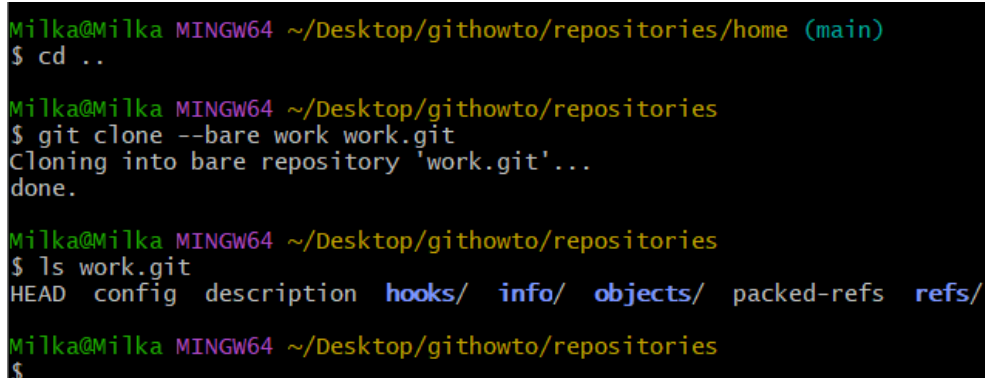
Теперь мы можем видеть ветку style в списке веток и логе.

Задание 38. Чистые репозитории

Чистый репозиторий — это репозиторий, не имеющий рабочей директории. Он содержит только директорию .git, в которой Git хранит все свои внутренние данные. Основное предназначение таких репозиториях — быть центральным хранилищем, в которое разработчики могут отправлять и из которого могут получать данные. Поэтому в них нет смысла создавать рабочие файлы, они будут только впустую занимать место на диске. Чистые репозитории также используются в сервисах Git-хостинга таких, как GitHub и GitLab. В следующих уроках мы узнаем, как создать чистый репозиторий и как отправлять в него изменения.

Создайте чистый репозиторий:

```
cd ..
git clone --bare work work.git
ls work.git
```



```
Milka@Milka MINGW64 ~/Desktop/ghowto/repositories/home (main)
$ cd ..

Milka@Milka MINGW64 ~/Desktop/ghowto/repositories
$ git clone --bare work work.git
Cloning into bare repository 'work.git'...
done.

Milka@Milka MINGW64 ~/Desktop/ghowto/repositories
$ ls work.git
HEAD  config  description  hooks/  info/  objects/  packed-refs  refs/

Milka@Milka MINGW64 ~/Desktop/ghowto/repositories
$
```

Рисунок 111 – Создание чистого репозитория

Принято считать, что репозитории, заканчивающиеся на .git, являются чистыми репозиториями. Мы видим, что в репозитории work.git нет рабочей директории. По сути, это просто директория .git из обычного репозитория.

Задание 39. Добавление удаленного репозитория

Давайте добавим репозиторий work.git к нашему оригинальному репозиторию.

```
cd work
git remote add shared ../work.git
```

```
Milka@Milka MINGW64 ~/Desktop/ghowto/repositories
$ cd work

Milka@Milka MINGW64 ~/Desktop/ghowto/repositories/work (main)
$ git remote add shared ../work.git
```

Рисунок 112 – Добавление удаленного репозитория

Задание 40. Отправка изменений

Поскольку чистые репозитории обычно располагаются на каком-либо удаленном сервере, вы не сможете туда просто зайти, дабы подтянуть изменения. Поэтому нам необходимо как-нибудь передать наши изменения в репозиторий.

Начнем с создания изменения, которое нужно передать в репозиторий. Отредактируйте README и закоммитьте его:

```
This is the Hello World example from the Git tutorial.
(changed in the origin and pushed to shared)

git switch main
git add README
git commit -m "Added shared comment to readme"
```

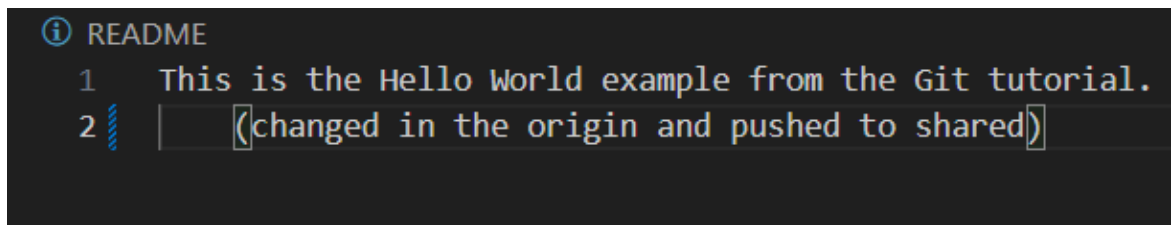


Рисунок 113 – Редактирование README

```
Milka@Milka MINGW64 ~/Desktop/ghowto/repositories/work (main)
$ git add README

Milka@Milka MINGW64 ~/Desktop/ghowto/repositories/work (main)
$ git commit -m "Added shared comment to readme"
[main 08bb955] Added shared comment to readme
1 file changed, 1 insertion(+), 1 deletion(-)

Milka@Milka MINGW64 ~/Desktop/ghowto/repositories/work (main)
$
```

Рисунок 114 – Коммит изменений

Теперь отправьте изменения в общий репозиторий.

```
git push shared main
```

```
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$ git push shared main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 405 bytes | 405.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To ../work.git
   af87242..08bb955  main -> main

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$
```

Рисунок 115 – Отправка изменений в общий репозиторий

Общим называется репозиторий, получающий отправленные нами изменения. Помните, мы добавили его в качестве удаленного репозитория в предыдущем уроке?

Задание 41. Подтягивание общих изменений

Быстро переключитесь в репозиторий home и подтяните изменения, только что отправленные в общий репозиторий.

```
cd ../home
```

Продолжите с...

```
git remote add shared ../work.git
git branch --track shared main
git pull shared main
cat README
```



```

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/work (main)
$ cd ../home

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/home (main)
$ git remote add shared ../work.git

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/home (main)
$ git branch --track shared main
branch 'shared' set up to track 'main'.

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/home (main)
$ git pull shared main
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 385 bytes | 32.00 KiB/s, done.
From ../work
 * branch          main      -> FETCH_HEAD
 * [new branch]    main      -> shared/main
Updating af87242..08bb955
Fast-forward
 README | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/home (main)
$ cat README
This is the Hello World example from the Git tutorial.
(changed in the origin and pushed to shared)
Milka@Milka MINGW64 ~/Desktop/githowto/repositories/home (main)

```

Рисунок 116 – Подтягивание общих изменений

Задание 42. Размещение ваших Git-репозитория

Хотите создать свой собственный GitHub? Существует множество способов совместного использования репозитория Git по сети. Здесь приведен простой и быстрый (но ненадежный и опасный) способ.

Запуск Git-сервера:

```

# (From the "repositories" directory)
git daemon --verbose --export-all --base-path=.

```

```

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/home (main)
$ # (From the "repositories" directory)

Milka@Milka MINGW64 ~/Desktop/githowto/repositories/home (main)
$ git daemon --verbose --export-all --base-path=.
[13832] Ready to rumble

```

Рисунок 117 – Запуск Git-сервера

Теперь в отдельном окне терминала перейдите в вашу директорию repositories:

```

# (From the "repositories" directory)
git clone git://localhost/work.git network_work
cd network_work
ls

```

```

milka@Milka MINGW64 ~
$ # (From the "repositories" directory)

milka@Milka MINGW64 ~
$ git clone git://localhost/work.git network_work
Cloning into 'network_work'...
fatal: read error: Invalid argument

milka@Milka MINGW64 ~
$ cd network_work
bash: cd: network_work: No such file or directory

milka@Milka MINGW64 ~
$ ls
AppData/
Application Data@
Contacts/
Cookies@
Desktop/
Documents/
Downloads/
Favorites/
Links/
Local Settings'@
Music/
MyApplicationNEW/
NTUSER.DAT
NTUSER.DAT{229b5caa-836e-11ee-8d56-ec79b28ac579}.TxR.0.regtrans-ms
NTUSER.DAT{229b5caa-836e-11ee-8d56-ec79b28ac579}.TxR.1.regtrans-ms
NTUSER.DAT{229b5caa-836e-11ee-8d56-ec79b28ac579}.TxR.2.regtrans-ms
NTUSER.DAT{229b5caa-836e-11ee-8d56-ec79b28ac579}.TxR.blf
NTUSER.DAT{229b5cab-836e-11ee-8d56-ec79b28ac579}.TM.blf
NTUSER.DAT{229b5cab-836e-11ee-8d56-ec79b28ac579}.TMContainer000000000000000000
1.regtrans-ms
NTUSER.DAT{229b5cab-836e-11ee-8d56-ec79b28ac579}.TMContainer000000000000000000
2.regtrans-ms
NetHood@
OneDrive/
Pictures/
PrintHood@
Recent@
'Saved Games'/
Searches/
SendTo@
Videos/
'WPS Cloud Files'/
ansel/
ntuser.dat.LOG1
ntuser.dat.LOG2
ntuser.ini
Мои документы'@
Шаблоны@
главное меню'@

```

Рисунок 118 – Открытие Git-сервера

Вы увидите копию проекта work.

Если вы хотите разрешить отправку изменений (push) в репозиторий Git Daemon, добавьте метку `--enable=receive-pack` к команде `git daemon`. Будьте осторожны, этот сервер не производит аутентификацию, поэтому любой сможет отправлять изменения в ваш репозиторий.

На этом этапе вам открываются безграничные возможности. Смелее! Возьмите в аренду сервер, купите доменное имя, разместите на этом сервере свои репозитории и наслаждайтесь своим личным GitHub!

Если серьезно, то вы можете самостоятельно разместить свой личный сервер GitLab. Этот продукт бесплатный и с открытым исходным кодом.

Спасибо за использование GitHowTo! Надеюсь, вам было интересно.

Выводы: Git — мощная и сложная распределенная система контроля версий, которая работает с изменениями, а не файлами. Git позволяет создавать репозитории с ветками, коммитить и отслеживать изменения в файлах, а также позволяет откатываться к любой интересующей версии проекта благодаря истории изменений.