

# Цель работы

---

Вспомнить основные команды Git Bash для работы с системой контроля версий git. Выполнить тренировочные задания по работе с проектом(репозиторием) с помощью git.

# План работы

---

- Подготовка
- Создание проекта
- Внесение изменений
- Индексация изменений
- Отмена локальных изменений (до индексации)
- Отмена проиндексированных изменений (перед коммитом)
- Отмена коммитов
- Удаление коммитов из ветки
- Удаление тега oops
- Внесение изменений в коммиты
- Перемещение файлов
- Второй способ перемещения файлов
- Подробнее о структуре
- Git внутри: Каталог .git
- Работа непосредственно с объектами git
- Создание ветки
- Навигация по веткам
- Изменения в ветке main
- Коммит изменений README.md в ветку main

- Слияние
- Создание конфликта
- Разрешение конфликтов
- Сброс ветки style
- Сброс ветки main
- Перебазирование
- Слияние в ветку main
- Клонирование репозиториев
- Просмотр клонированного репозитория
- Что такое origin?
- Удаленные ветки
- Изменение оригинального репозитория
- Слияние извлеченных изменений
- Добавление ветки наблюдения
- Чистые репозитории
- Создание чистого репозитория
- Добавление удаленного репозитория
- Отправка изменений
- Извлечение общих изменений

## Выполнение лабораторной работы

---

### Подготовка к работе с git

Для начала работы необходимо выполнить команды `git config`, чтобы git узнал ваше имя и электронную почту. Однако, так как я ранее уже использовал git, то просто проверим это с помощью команды `git config user.name` и `git config user.email`, которая выводит информацию о имени и электронной почте

пользователя соответственно. В ней видно данные пользователя в последних двух строках. (рис. [-@fig:001])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/1
ab1
$ git config user.name
arseniy ilyinsky

dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/1ab1
$ git config user.email
ilinskiyar@gmail.com
```

{#fig:001 width=100%}

Также необходимо установить параметры окончаний строк:

- Настройка core.autocrlf с параметрами true и input делает все переводы строк текстовых файлов в главном репозитории одинаковыми. (рис. [-@fig:002]).
- Если core.safecrlf установлен в true или warm, git проверяет, если преобразование является обратимым для текущей настройки core.autocrlf. core.safecrlf true - отвержение необратимого преобразования lf<->crlf.

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/1ab1
$ git config --global core.autocrlf input

dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/1ab1
$ git config --global core.safecrlf true
```

{#fig:002 width=70%}

Так как по умолчанию, git будет печатать не-ASCII символы в именах файлов в виде восьмеричных последовательностей \nnn. Что бы избежать нечитаемых строк, установим соответствующий флаг. (рис. [-@fig:003])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/1ab1
$ git config --global core.quotepath off
```

{#fig:003 width=100%}

## Создание проекта

Создадим пустой каталог hello, затем войдем него и создадим файл с именем hello.html, содержащий приветствие "Hello world!". (рис. [-@fig:004])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/1ab1
$ mkdir hello

dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/1ab1
$ cd hello

dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/1ab1/hello
$ touch hello.html

dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/1ab1/hello
$ echo "Hello, World!" > hello.html
```

{#fig:004 width=70%}

Чтобы создать git репозиторий из этого каталога, выполним команду git init. (рис. [-@fig:005])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello
$ git init
Initialized empty Git repository in D:/ВУЗ - образование/Математическое моделирование/lab1/hello/.git/
{#fig:005 width=100%}
```

Добавим файл в репозиторий. (рис. [-@fig:006])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ git add hello.html

dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ git commit -m "Initial Commit"
[main (root-commit) 6ca1299] Initial Commit
 1 file changed, 1 insertion(+)
 create mode 100644 hello.html
```

{#fig:006 width=70%}

Используем команду git status, чтобы проверить текущее состояние репозитория. (рис. [-@fig:007])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ git status
On branch main
nothing to commit, working tree clean
```

{#fig:007 width=100%}

## Внесение изменений

Добавим кое-какие HTML-теги к нашему приветствию. Изменим содержимое файла hello.html на:

*Hello, World!*

---

И проверим состояние рабочего каталога. (рис. [-@fig:008])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (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")
```

{#fig:008 width=100%}

git знает, что файл hello.html был изменен, но при этом эти изменения еще не зафиксированы в репозитории.

## Индексация изменений

Теперь выполним команду git, чтобы проиндексировать изменения. (рис. [-@fig:009]).

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ git add hello.html
```

{#fig:009 width=70%}

А затем проверим состояние репозитория. рис. [-@fig:010])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   hello.html
```

{#fig:010 width=100%}

Изменения файла hello.html были проиндексированы. Это означает, что git теперь знает об изменении, но изменение пока не записано в репозиторий. Следующий коммит будет включать в себя проиндексированные изменения.

Сделаем коммит. (рис. [-@fig:011])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ git commit
```

{#fig:010 width=100%}

Открылся редактор. В первой строке введем комментарий: «Added h1 tag». (рис. [-@fig:012])

```
"Added h1 tag"
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch main
# Changes to be committed:
#   modified:   hello.html
```

{#fig:012

width=100%}

Сохраним файл и выйдем из редактора. (рис. [-@fig:013])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ git commit
[main 5f92f79] "Added h1 tag"
 1 file changed, 1 insertion(+), 1 deletion(-)
```

{#fig:013 width=100%}

Снова проверим состояние. (рис. [-@fig:014])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ git status
On branch main
nothing to commit, working tree clean
```

{#fig:014 width=100%}

Рабочий каталог чистый, можно продолжить работу.

Изменим страницу «Hello, World», чтобы она содержала стандартные теги и .

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

Теперь добавим это изменение в индекс git. (рис. [-@fig:015])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ git add hello.html
```

{#fig:015 width=100%}

Теперь добавим заголовки HTML (секцию ) к странице «Hello, World».

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

Проверим текущий статус. (рис. [-@fig:016])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (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
```

{#fig:016 width=100%}

hello.html указан дважды в состоянии. Первое изменение (добавление стандартных тегов) проиндексировано и готово к коммиту. Второе изменение (добавление заголовков HTML) является непроиндексированным. Если бы мы делали коммит сейчас, заголовки не были бы сохранены в репозиторий.

Произведем коммит проиндексированного изменения (значение по умолчанию). (рис. [-@fig:017])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ git commit -m "Added standard HTML page tags"
[main b38204d] Added standard HTML page tags
 1 file changed, 5 insertions(+), 1 deletion(-)
```

{#fig:017 width=100%}

Еще раз проверим состояние. (рис. [-@fig:018])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (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
```

{#fig:018 width=100%}

Состояние команды говорит о том, что hello.html имеет незафиксированные изменения, но уже не в буферной зоне.

Теперь добавим второе изменение в индекс, а затем проверим состояние. (рис. [-@fig:019])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ git add .

dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   hello.html
```

{#fig:019 width=100%}

В качестве файла для добавления, мы используем текущий каталог (.). Это краткий и удобный путь для добавления всех изменений в файлы текущего каталога и его подкаталоги. Но поскольку он добавляет все, не лишним будет проверить состояние перед запуском add, просто чтобы убедиться, что мы не добавили какой-то файл, который добавлять было не нужно.

Второе изменение было проиндексировано и готово к коммиту. Выполним коммит. (рис. [-@fig:020])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ git commit -m "Added HTML header"
[main b840713] Added HTML header
 1 file changed, 2 insertions(+)
```

{#fig:020 width=100%}

Получим список произведенных изменений: (рис. [-@fig:021])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ git log
commit b840713180dfb70179447e6ca7a8d2db8d77d3d0 (HEAD -> main)
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date:   Thu Feb 10 21:00:48 2022 +0300

  Added HTML header

commit b38204d2e0d18817d4ec693756bcc6378002ba9b
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date:   Thu Feb 10 20:59:57 2022 +0300

  Added standard HTML page tags

commit 5f92f7965ccc0c32eb20ed38af6c7200b1164320
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date:   Thu Feb 10 20:57:05 2022 +0300

  "Added h1 tag"

commit 6ca1299253649153cd6effea99efe0e15f25fda7
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date:   Thu Feb 10 20:54:39 2022 +0300

  Initial Commit
```

{#fig:021 width=100%}

Однострочный формат истории: (рис. [-@fig:022])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ git log --pretty=oneline
b840713180dfb70179447e6ca7a8d2db8d77d3d0 (HEAD -> main) Added HTML header
b38204d2e0d18817d4ec693756bcc6378002ba9b Added standard HTML page tags
5f92f7965ccc0c32eb20ed38af6c7200b1164320 "Added h1 tag"
6ca1299253649153cd6effea99efe0e15f25fda7 Initial Commit
```

{#fig:022 width=100%}

Получив хэши предыдущих версий, изучил данные лога и нашел хэш для первого коммита - 6ca1299...

Использовал этот хэш-код для перехода к изначальному снимку репозитория с помощью команды checkout. (рис. [-@fig:023])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ git checkout "6ca1299253649153cd6effea99efe0e15f25fda7"
Note: switching to '6ca1299253649153cd6effea99efe0e15f25fda7'.

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 6ca1299 Initial Commit

{#fig:023 width=100%}

И проверил содержимое файла hello.html. (рис. [-@fig:024])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello ((6ca1299...))
$ cat hello.html
Hello, World!
```

{#fig:024 width=100%}

Затем вернулся к последней версии ветки main. (рис. [-@fig:025])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello ((6ca1299...))
$ git checkout main
Previous HEAD position was 6ca1299 Initial Commit
Switched to branch 'main'
```

{#fig:025 width=100%}

И проверил содержимое файла hello.html. (рис. [-@fig:026])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ cat hello.html
<html>
  <head>
  </head>
  <body>
    <h1>Hello, World!</h1>
  </body>
</html>
```

{#fig:026 width=100%}

Назовем текущую версию страницы hello первой (v1). Для этого создадим тег с помощью команды git tag v1. Теперь текущая версия страницы называется v1. (рис. [-@fig:027])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ git tag v1
```

{#fig:027 width=100%}

Давайте создадим тег для версии, которая идет перед текущей версией и назовем ее v1-beta. В первую очередь нам надо переключиться на предыдущую версию. Вместо поиска до хэш, мы будем использовать ^, обозначающее «родитель v1». Вместо обозначения  $v1^$  можно использовать  $v1\sim 1$ . Это обозначение можно определить как «первую версию предшествующую v1». (рис. [-@fig:028])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ git checkout v1^
Note: switching to 'v1^'.
```

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 b38204d Added standard HTML page tags

{#fig:028 width=100%}

Это версия с тегами и , но еще пока без . Давайте сделаем ее версией v1-beta с помощью команды git tag v1-beta. (рис. [-@fig:029])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello ((b38204d...))
$ git tag v1-beta
```

{#fig:029 width=100%}

Теперь попробуем попереключаться между двумя отмеченными версиями командой git checkout *имя соответствующего тега*. (рис. [-@fig:030])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello ((v1-beta))
$ git checkout v1
Previous HEAD position was b38204d Added standard HTML page tags
HEAD is now at b840713 Added HTML header
```

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello ((v1))
$ git checkout v1-beta
Previous HEAD position was b840713 Added HTML header
HEAD is now at b38204d Added standard HTML page tags
```

{#fig:030 width=100%}

Просмотрим доступные теги с помощью команды git tag.

Также теги можно посмотреть в логе. (рис. [-@fig:031])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/вуз - образование/Математическое моделирование/lab1/hello ((v1-beta))
$ git tag
v1
v1-beta
{#fig:031 width=100%}
```

## Отмена локальных изменений (до индексации)

Иногда случается, что вы изменили файл в рабочем каталоге, и хотите отменить последние коммиты. С этим справится команда git checkout.

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

```
<html>
  <head>
  </head>
  <body>
    <h1>Hello, World!</h1>
    <!-- This is a bad comment. We want to revert it. -->
  </body>
</html>
```

Сначала проверим состояние рабочего каталога. (рис. [-@fig:032])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/вуз - образование/Математическое моделирование/lab1/hello (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")
{#fig:032 width=100%}
```

Мы видим, что файл hello.html был изменен, но еще не проиндексирован.

Используем команду git checkout для переключения версии файла hello.html в репозитории. (рис. [-@fig:033], рис. [-@fig:034])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/вуз - образование/Математическое моделирование/lab1/hello (main)
$ git checkout hello.html
Updated 1 path from the index
{#fig:033 width=100%}
```

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/вуз - образование/Математическое моделирование/lab1/hello (main)
$ git status
On branch main
nothing to commit, working tree clean

dark2@LAPTOP-6ITRB0QA MINGW64 /d/вуз - образование/Математическое моделирование/lab1/hello (main)
$ cat hello.html
<html>
<head>
</head>
<body>
<h1>Hello, World!</h1>
</body>
</html>
```

{#fig:034 width=100%}

Команда git status показывает нам, что не было произведено никаких изменений, не зафиксированных в рабочем каталоге.

## Отмена проиндексированных изменений (перед коммитом)

Внесем 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>
```

Проиндексируем это изменение. (рис. [-@fig:035])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/вуз - образование/Математическое моделирование/lab1/hello (main)
$ git add hello.html
```

{#fig:035 width=100%}

Проверим состояние после нежелательного изменения. (рис. [-@fig:036])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/вуз - образование/Математическое моделирование/lab1/hello (main)
$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   hello.html
```

{#fig:036 width=100%}

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

К счастью, вывод состояния показывает нам именно то, что мы должны сделать для отмены индексации изменения.

Команда git reset сбрасывает буферную зону к HEAD. Это очищает буферную зону от изменений, которые мы только что проиндексировали. (рис. [-@fig:037])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/вуз - образование/Математическое моделирование/lab1/hello (main)
$ git reset HEAD hello.html
Unstaged changes after reset:
M     hello.html
```

{#fig:037 width=100%}

Команда git reset (по умолчанию) не изменяет рабочий каталог. Поэтому рабочий каталог все еще содержит нежелательный комментарий. Мы можем использовать команду git checkout, чтобы удалить нежелательные изменения в рабочем каталоге. (рис. [-@fig:038])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/вуз - образование/Математическое моделирование/lab1/hello (main)
$ git checkout hello.html
Updated 1 path from the index
```

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/вуз - образование/Математическое моделирование/lab1/hello (main)
$ git status
On branch main
nothing to commit, working tree clean
```

{#fig:038 width=100%}

Наш рабочий каталог опять чист.

## Отмена коммитов

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

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

Изменим файл hello.html на:

```
<html>
  <head>
  </head>
  <body>
    <h1>Hello, World!</h1>
    <!-- This is an unwanted but committed change -->
  </body>
</html>
```

Проиндексируем его и выполним коммит. (рис. [-@fig:039])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/вуз - образование/Математическое моделирование/lab1/hello (main)
$ git add hello.html
```

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/вуз - образование/Математическое моделирование/lab1/hello (main)
$ git commit -m "Oops, we didn't want this commit"
[main 19decdb] Oops, we didn't want this commit
 1 file changed, 5 insertions(+), 4 deletions(-)
```

{#fig:039 width=100%}

Чтобы отменить коммит, нам необходимо сделать коммит, который удаляет изменения, сохраненные нежелательным коммитом. (рис. [-@fig:040])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/вуз - образование/Математическое моделирование/lab1/hello (main)
$ git revert HEAD
[main 8de1d14] Revert "Oops, we didn't want this commit"
 1 file changed, 4 insertions(+), 5 deletions(-)
```

{#fig:040 width=100%}

Так как мы отменили самый последний произведенный коммит, мы смогли использовать HEAD в качестве аргумента для отмены. Мы можем отменить любой произвольной коммит в истории, указав его хэш-значение.

Проверка лога показывает нежелательные и отмененные коммиты в наш репозиторий. (рис. [-@fig:041])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/вуз - образование/Математическое моделирование/lab1/hello (main)
$ git log
commit 8de1d14525b4bc792ab395c3911c04a0567d344b (HEAD -> main)
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date:   Thu Feb 10 21:12:40 2022 +0300

    Revert "Oops, we didn't want this commit"

    This reverts commit 19decdbd1521a8e60e8e9f3b87451ae7da7c49b87.

commit 19decdbd1521a8e60e8e9f3b87451ae7da7c49b87
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date:   Thu Feb 10 21:12:19 2022 +0300

    Oops, we didn't want this commit

commit b840713180dfb70179447e6ca7a8d2db8d77d3d0 (tag: v1)
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date:   Thu Feb 10 21:00:48 2022 +0300

    Added HTML header

commit b38204d2e0d18817d4ec693756bcc6378002ba9b (tag: v1-beta)
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date:   Thu Feb 10 20:59:57 2022 +0300

    Added standard HTML page tags

commit 5f92f7965ccc0c32eb20ed38af6c7200b1164320
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date:   Thu Feb 10 20:57:05 2022 +0300

    "Added h1 tag"

commit 6ca1299253649153cd6effea99efe0e15f25fd7
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date:   Thu Feb 10 20:54:39 2022 +0300

    Initial Commit
```

{#fig:041 width=100%}

## Удаление коммитов из ветки

git revert является мощной командой, которая позволяет отменить любые коммиты в репозиторий. Однако, и оригинальный и «отмененный» коммиты видны в истории ветки (при использовании команды git log).

Часто мы делаем коммит, и сразу понимаем, что это была ошибка. Было бы неплохо иметь команду «возврата», которая позволила бы нам сделать вид, что неправильного коммита никогда и не было. Команда «возврата» даже предотвратила бы появление нежелательного коммита в истории git log.

При получении ссылки на коммит (т.е. хэш, ветка или имя тега), команда git reset:

- перепишет текущую ветку, чтобы она указывала на нужный коммит;
- опционально сбросит буферную зону для соответствия с указанным коммитом;
- опционально сбросит рабочий каталог для соответствия с указанным коммитом.

Давайте сделаем быструю проверку нашей истории коммитов. (рис. [-@fig:042])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/вуз - образование/Математическое моделирование/lab1/hello (main)
$ git log
commit 8de1d14525b4bc792ab395c3911c04a0567d344b (HEAD -> main)
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date:   Thu Feb 10 21:12:40 2022 +0300

    Revert "Oops, we didn't want this commit"

    This reverts commit 19decabd1521a8e60e8e9f3b87451ae7da7c49b87.

commit 19decabd1521a8e60e8e9f3b87451ae7da7c49b87
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date:   Thu Feb 10 21:12:19 2022 +0300

    Oops, we didn't want this commit

commit b840713180dfb70179447e6ca7a8d2db8d77d3d0 (tag: v1)
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date:   Thu Feb 10 21:00:48 2022 +0300

    Added HTML header

commit b38204d2e0d18817d4ec693756bcc6378002ba9b (tag: v1-beta)
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date:   Thu Feb 10 20:59:57 2022 +0300

    Added standard HTML page tags

commit 5f92f7965ccc0c32eb20ed38af6c7200b1164320
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date:   Thu Feb 10 20:57:05 2022 +0300

    "Added h1 tag"

commit 6ca1299253649153cd6effea99efe0e15f25fda7
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date:   Thu Feb 10 20:54:39 2022 +0300

Initial Commit
```

{#fig:042 width=100%}

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. (рис. [-@fig:043])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/вуз - образование/Математическое моделирование/lab1/hello (main)
$ git tag oops
```

{#fig:043 width=100%}

Глядя на историю лога, 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 (если она не имеет тега, мы можем использовать хеш-значение). (рис. [-@fig:044])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ git reset --hard v1
HEAD is now at b840713 Added HTML header

dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ git log
commit b840713180dfb70179447e6ca7a8d2db8d77d3d0 (HEAD -> main, tag: v1)
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date:   Thu Feb 10 21:00:48 2022 +0300

    Added HTML header

commit b38204d2e0d18817d4ec693756bcc6378002ba9b (tag: v1-beta)
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date:   Thu Feb 10 20:59:57 2022 +0300

    Added standard HTML page tags

commit 5f92f7965ccc0c32eb20ed38af6c7200b1164320
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date:   Thu Feb 10 20:57:05 2022 +0300

    "Added h1 tag"

commit 6ca1299253649153cd6effea99efe0e15f25fda7
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date:   Thu Feb 10 20:54:39 2022 +0300

    Initial Commit
```

{#fig:044 width=100%}

Наша ветка main теперь указывает на коммит v1, а коммитов Oops и Revert Oops в ветке уже нет.

Параметр --hard указывает, что рабочий каталог должен быть обновлен в соответствии с новым head 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. (рис. [-@fig:045])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ git log --all
commit 8de1d14525b4bc792ab395c3911c04a0567d344b (tag: oops)
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date:   Thu Feb 10 21:12:40 2022 +0300

    Revert "Oops, we didn't want this commit"

    This reverts commit 19decabd1521a8e60e8e9f3b87451ae7da7c49b87.

commit 19decabd1521a8e60e8e9f3b87451ae7da7c49b87
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date:   Thu Feb 10 21:12:19 2022 +0300

    Oops, we didn't want this commit

commit b840713180dfb70179447e6ca7a8d2db8d77d3d0 (HEAD -> main, tag: v1)
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date:   Thu Feb 10 21:00:48 2022 +0300

    Added HTML header

commit b38204d2e0d18817d4ec693756bcc6378002ba9b (tag: v1-beta)
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date:   Thu Feb 10 20:59:57 2022 +0300

    Added standard HTML page tags

commit 5f92f7965ccc0c32eb20ed38af6c7200b1164320
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date:   Thu Feb 10 20:57:05 2022 +0300

    "Added h1 tag"

commit 6ca1299253649153cd6effea99efe0e15f25fda7
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date:   Thu Feb 10 20:54:39 2022 +0300

    Initial Commit
```

{#fig:045 width=100%}

Мы видим, что ошибочные коммиты не исчезли. Они все еще находятся в репозитории. Просто они отсутствуют в ветке main. Если бы мы не отметили их тегами, они по-прежнему находились бы в репозитории, но не было бы никакой возможности ссылаться на них, кроме как при помощи их хэш имен. Коммиты, на которые нет ссылок, остаются в репозитории до тех пор, пока не будет запущен сборщик мусора.

Сброс в локальных ветках, как правило, безопасен. Последствия любой «аварии» как правило, можно восстановить простым сбросом с помощью нужного коммита. Однако, если ветка «расшарена» на удаленных репозиториях, сброс может сбить с толку других пользователей ветки.

## Удаление тега oops

Тег oops свою функцию выполнил. Теперь удалим его и коммиты, на которые он ссылался, сборщиком мусора. (рис. [-@fig:046])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ git tag -d oops
Deleted tag 'oops' (was 8de1d14)

dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ git log --all
commit b840713180dfb70179447e6ca7a8d2db8d77d3d0 (HEAD -> main, tag: v1)
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date:   Thu Feb 10 21:00:48 2022 +0300

    Added HTML header

commit b38204d2e0d18817d4ec693756bcc6378002ba9b (tag: v1-beta)
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date:   Thu Feb 10 20:59:57 2022 +0300

    Added standard HTML page tags

commit 5f92f7965ccc0c32eb20ed38af6c7200b1164320
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date:   Thu Feb 10 20:57:05 2022 +0300

    "Added h1 tag"

commit 6ca1299253649153cd6effea99efe0e15f25fda7
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date:   Thu Feb 10 20:54:39 2022 +0300

    Initial Commit
```

{#fig:046 width=100%}

Тер «oops» больше не будет отображаться в репозитории.

## Внесение изменений в коммиты

Добавим в страницу комментарий автора (свою фамилию).

```
<!-- Author: Arsenij A. Ilinskij -->
<html>
  <head>
  </head>
  <body>
    <h1>Hello, World!</h1>
  </body>
</html>
```

Выполним коммит изменений. (рис. [-@fig:047])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ git add hello.html

dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ git commit -m "Add an author comment"
[main fba43eb] Add an author comment
 1 file changed, 1 insertion(+)
```

{#fig:047 width=100%}

Обновим страницу hello, включив в нее email.

```
<!-- Author: Arsenij A. Ilinskij ( ilinskyar@gmail.com ) -->
<html>
  <head>
  </head>
  <body>
    <h1>Hello, World!</h1>
  </body>
</html>
```

Мы действительно не хотим создавать отдельный коммит только ради электронной почты. Давайте изменим предыдущий коммит, включив в него адрес электронной почты. (рис. [-@fig:048])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ git add hello.html

dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ git commit --amend -m "Add an author/email comment"
[main 37294a8] Add an author/email comment
  Date: Thu Feb 10 21:29:58 2022 +0300
  1 file changed, 2 insertions(+), 1 deletion(-)
```

{#fig:048 width=100%}

Просмотрим историю изменений. (рис. [-@fig:049])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ git log
commit 37294a83a3b46fe829627e0378da1943d0528c03 (HEAD -> main)
Author: arseniy ilyinsky <ilinskyar@gmail.com>
Date:   Thu Feb 10 21:29:58 2022 +0300

  Add an author/email comment

commit b840713180dfb70179447e6ca7a8d2db8d77d3d0 (tag: v1)
Author: arseniy ilyinsky <ilinskyar@gmail.com>
Date:   Thu Feb 10 21:00:48 2022 +0300

  Added HTML header

commit b38204d2e0d18817d4ec693756bcc6378002ba9b (tag: v1-beta)
Author: arseniy ilyinsky <ilinskyar@gmail.com>
Date:   Thu Feb 10 20:59:57 2022 +0300

  Added standard HTML page tags

commit 5f92f7965ccc0c32eb20ed38af6c7200b1164320
Author: arseniy ilyinsky <ilinskyar@gmail.com>
Date:   Thu Feb 10 20:57:05 2022 +0300

  "Added h1 tag"

commit 6ca1299253649153cd6effea99efe0e15f25fda7
Author: arseniy ilyinsky <ilinskyar@gmail.com>
Date:   Thu Feb 10 20:54:39 2022 +0300

  Initial Commit
```

{#fig:049 width=100%}

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.

## Перемещение файлов

Сейчас мы собираемся создать структуру нашего репозитория. Давайте создадим каталог lib и перенесем файл hello.html в него. (рис. [-@fig:050])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ mkdir lib

dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ git mv hello.html lib

dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    renamed:   hello.html -> lib/hello.html
```

{#fig:050 width=100%}

Перемещая файлы с помощью git mv, мы информируем git о 2 вещах:

- Что файл hello.html был удален.
- Что файл lib/hello.html был создан.
- Оба эти факта сразу же проиндексированы и готовы к коммиту. Команда git status сообщает, что файл был перемещен.

## Второй способ перемещения файлов

Положительной чертой git является то, что вы можете забыть о версионном контроле до того момента, когда вы готовы приступить к коммиту кода. Что бы случилось, если бы мы использовали командную строку операционной системы для перемещения файлов вместо команды git?

Следующий набор команд идентичен нашим последним действиям. Работы здесь побольше, но результат тот же. Мы могли бы выполнить:

```
mkdir lib
mv hello.html lib
git add lib/hello.html
git rm hello.html
```

Давайте сделаем коммит этого перемещения: (рис. [-@fig:051])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ git commit -m "Moved hello.html to lib"
[main 58e4685] Moved hello.html to lib
1 file changed, 0 insertions(+), 0 deletions(-)
rename hello.html => lib/hello.html (100%)
```

{#fig:051 width=100%}

## Подробнее о структуре

Создадим (рис. [-@fig:052]) и добавим файл index.html в наш репозиторий.

```
<html>
<body>
```

```
<iframe src="lib/hello.html" width="200" height="200" />
</body>
</html>
```

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/вуз - образование/Математическое моделирование/lab1/hello (main)
$ touch index.html
```

{#fig:052 width=100%}

Проиндексируем и закоммитим изменения. (рис. [-@fig:053])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/вуз - образование/Математическое моделирование/lab1/hello (main)
$ git add index.html

dark2@LAPTOP-6ITRB0QA MINGW64 /d/вуз - образование/Математическое моделирование/lab1/hello (main)
$ git commit -m "Added index.html"
[main 6564b23] Added index.html
 1 file changed, 5 insertions(+)
 create mode 100644 index.html
```

{#fig:053 width=100%}

Теперь при открытии index.html, мы видим кусок страницы hello в маленьком окошке. (рис. [-@fig:054])



{#fig:054 width=100%}

## Git внутри: Каталог .git

Выполним

```
ls -C .git
```

Это каталог, в котором хранится вся информация git. (рис. [-@fig:055])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/вуз - образование/Математическое моделирование/lab1/hello (main)
$ ls -C .git
COMMIT_EDITMSG  ORIG_HEAD  description  index  logs/    packed-refs
HEAD           config     hooks/       info/   objects/  refs/
```

{#fig:055 width=100%}

Выполним

```
ls -C .git/objects
```

Мы видим набор каталогов, имена которых состоят из 2 символов. Имена каталогов являются первыми двумя буквами хэша sha1 объекта, хранящегося в git. (рис. [-@fig:056])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/вуз - образование/Математическое моделирование/lab1/hello (main)
$ ls -C .git/objects
05/ 19/ 36/ 47/ 52/ 58/ 61/ 64/ 6c/ 8a/ 99/ a4/ ae/ b8/ e2/ e8/ f7/ info/
14/ 31/ 37/ 4a/ 53/ 5f/ 62/ 65/ 72/ 8d/ 9e/ a9/ b3/ e1/ e7/ f2/ fb/ pack/
```

{#fig:056 width=100%}

Выполним

```
ls -C .git/objects/0a
```

Смотрим в один из каталогов с именем из 2 символов. Мы видим файлы с именами из 38 символов. Это файлы, содержащие объекты, хранящиеся в git. Они сжаты и закодированы, поэтому просмотр их содержимого нам мало чем поможет. (рис. [-@fig:057])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/вуз - образование/Математическое моделирование/lab1/hello (main)
$ ls -C .git/objects/47
b5d9ec954ec9b213762f4dcfc9c71a855dce4
```

{#fig:057 width=100%}

Выполним

```
cat .git/config
```

Это файл конфигурации, создающийся для каждого конкретного проекта. Записи в этом файле будут перезаписывать записи в файле .gitconfig вашего главного каталога, по крайней мере в рамках этого проекта. (рис. [-@fig:058])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/вуз - образование/Математическое моделирование/lab1/hello (main)
$ cat .git/config
[core]
  repositoryformatversion = 0
  filemode = false
  bare = false
  logallrefupdates = true
  symlinks = false
  ignorecase = true
```

{#fig:058 width=100%}

Выполним

```
ls .git/refs
ls .git/refs/heads
ls .git/refs/tags
cat .git/refs/tags/v1
```

Вы должны узнавать файлы в подкаталоге тегов. Каждый файл соответствует тегу, ранее созданному с помощью команды git tag. Его содержание — это всего лишь хэш коммита, привязанный к тегу. Каталог heads практически аналогичен, но используется для веток, а не тегов. На данный момент у нас есть только одна ветка, так что все, что вы увидите в этом каталоге – это ветка main. (рис. [-@fig:059])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ ls .git/refs
heads/  tags/
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ ls .git/refs/heads
main
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ ls .git/refs/tags
v1  v1-beta
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ cat .git/refs/tags/v1
b840713180dfb70179447e6ca7a8d2db8d77d3d0
```

{#fig:059 width=100%}

Выполним

```
cat .git/HEAD
```

Файл HEAD содержит ссылку на текущую ветку, в данный момент это должна быть ветка main. (рис. [-@fig:060])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ cat .git/HEAD
ref: refs/heads/main
```

{#fig:060 width=100%}

## Работа непосредственно с объектами git

Выполним

```
git log --max-count=1
```

Эта команда показывает последний коммит в репозитории. (рис. [-@fig:061])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ git log --max-count=1
commit 6564b23673a7381583964e75191a2aa821e5b67a (HEAD -> main)
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date:   Thu Feb 10 21:54:06 2022 +0300

    Added index.html
```

{#fig:061 width=100%}

Выведем последний коммит с помощью SHA1 хэша с помощью команд (рис. [-@fig:062]):

```
git cat-file -t <hash>
git cat-file -p <hash>
```

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ git cat-file -t "6564b23673a7381583964e75191a2aa821e5b67a"
commit

dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ git cat-file -p "6564b23673a7381583964e75191a2aa821e5b67a"
tree 0546109a900e42a846a0b9525dccca230147d8823
parent 58e4685b1a9bd6b5ab41cf53f25161efd81d7a6f
author arseniy ilyinsky <ilinskiyar@gmail.com> 1644519246 +0300
committer arseniy ilyinsky <ilinskiyar@gmail.com> 1644519246 +0300

Added index.html
```

{#fig:062 width=100%}

Мы можем вывести дерево каталогов, ссылка на который идет в коммите. Это должно быть описание файлов (верхнего уровня) в нашем проекте (для конкретного коммита). Используем SHA1 хэш(0546109...) из строки «tree», из списка выше. Для этого выполним (рис. [-@fig:063]):

```
git cat-file -p 0546109
```

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ git cat-file -p "0546109a900e42a846a0b9525dccca230147d8823"
100644 blob 47b5d9ec954ec9b213762f4dcfc9c71a855dce4    index.html
040000 tree 3659d2b0e969cb218e4c5ab7c026cead4013f885   lib
```

{#fig:063 width=100%}

Выполним то же самое, только с хэшом каталога SHA1 хэш(3659d2b...) из строки «lib», из списка выше. (рис. [-@fig:064]):

```
git cat-file -p 3659d2b
```

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ git cat-file -p "3659d2b0e969cb218e4c5ab7c026cead4013f885"
100644 blob a4f60895c6da79bf495c50fad9d6e40a8de41b4f    hello.html
```

{#fig:064 width=100%}

Выполним то же самое, только с хэшом каталога SHA1 хэш(a4f6089...) из строки «hello», из списка выше. (рис. [-@fig:065]):-

```
git cat-file -p a4f6089
```

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/вуз - образование/Математическое моделирование/lab1/hello (main)
$ git cat-file -p "a4f60895c6da79bf495c50fad9d6e40a8de41b4f"
<!-- Author: Arsenij A. Ilinskij ( ilinskyar@gmail.com ) -->
<html>
<head>
</head>
<body>
  <h1>Hello, World!</h1>
</body>
</html>
```

{#fig:065 width=70%}

Дойдем до оригинального файла hello.html с самого первого коммита вручную по ссылкам SHA1 хэша. Для этого будемходить по хэшу parent, пока не дойдем до начального коммита. (рис. [-@fig:067], рис. [-@fig:068])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/вуз - образование/Математическое моделирование/1
ab1/hello (main)
$ git cat-file -p "58e4685b1a9bd6b5ab41cf53f25161efd81d7a6f"
tree e29c0c3fff1f993df8f5cc555f48face2880492e
parent 37294a83a3b46fe829627e0378da1943d0528c03
author arseniy ilyinsky <ilinskyar@gmail.com> 1644518075 +0300
committer arseniy ilyinsky <ilinskyar@gmail.com> 1644518075 +0300
```

Moved hello.html to lib

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/вуз - образование/Математическое моделирование/1
ab1/hello (main)
$ git cat-file -p "37294a83a3b46fe829627e0378da1943d0528c03"
tree 3659d2b0e969cb218e4c5ab7c026cead4013f885
parent b840713180dfb70179447e6ca7a8d2db8d77d3d0
author arseniy ilyinsky <ilinskyar@gmail.com> 1644517798 +0300
committer arseniy ilyinsky <ilinskyar@gmail.com> 1644517890 +0300
```

Add an author/email comment

{#fig:067 width=100%}

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/вуз - образование/Математическое моделирование/1
ab1/hello (main)
$ git cat-file -p "b840713180dfb70179447e6ca7a8d2db8d77d3d0"
tree 3113cc5b7683c93d829f2fc7187aae450bfedce9
parent b38204d2e0d18817d4ec693756bcc6378002ba9b
author arseniy ilyinsky <ilinskyar@gmail.com> 1644516048 +0300
committer arseniy ilyinsky <ilinskyar@gmail.com> 1644516048 +0300
```

Added HTML header

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/вуз - образование/Математическое моделирование/1
ab1/hello (main)
$ git cat-file -p "b38204d2e0d18817d4ec693756bcc6378002ba9b"
tree a93988585391b784dc988bee620f41c64968589d
parent 5f92f7965ccc0c32eb20ed38af6c7200b1164320
author arseniy ilyinsky <ilinskyar@gmail.com> 1644515997 +0300
committer arseniy ilyinsky <ilinskyar@gmail.com> 1644515997 +0300
```

Added standard HTML page tags

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/вуз - образование/Математическое моделирование/1
ab1/hello (main)
$ git cat-file -p "5f92f7965ccc0c32eb20ed38af6c7200b1164320"
tree 629377ed8eb96afede9fd6cb7b48b1e4a1d3dc56
parent 6ca1299253649153cd6effea99efe0e15f25fda7
author arseniy ilyinsky <ilinskyar@gmail.com> 1644515825 +0300
committer arseniy ilyinsky <ilinskyar@gmail.com> 1644515825 +0300
```

"Added h1 tag"

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/вуз - образование/Математическое моделирование/1
ab1/hello (main)
$ git cat-file -p "6ca1299253649153cd6effea99efe0e15f25fda7"
tree 523cd318fc3b74ad3852cd34a74720ad6f7dd422
author arseniy ilyinsky <ilinskyar@gmail.com> 1644515679 +0300
committer arseniy ilyinsky <ilinskyar@gmail.com> 1644515679 +0300
```

Initial Commit

{#fig:068 width=100%}

Оригинальный файл hello.html. (рис. [-@fig:069])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/вуз - образование/Математическое моделирование/lab1/hello (main)
$ git cat-file -p "523cd318fc3b74ad3852cd34a74720ad6f7dd422"
100644 blob 8ab686eafeb1f44702738c8b0f24f2567c36da6d hello.html

dark2@LAPTOP-6ITRB0QA MINGW64 /d/вуз - образование/Математическое моделирование/lab1/hello (main)
$ git cat-file -p "8ab686eafeb1f44702738c8b0f24f2567c36da6d"
Hello, World!
```

{#fig:069 width=100%}

## Создание ветки

Пора сделать наш hello world более выразительным. Так как это может занять некоторое время, лучше переместить эти изменения в отдельную ветку, чтобы изолировать их от изменений в ветке main.

Назовем нашу новую ветку «style». Выполним (рис. [-@fig:070]):

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ git checkout -b style
Switched to a new branch 'style'

dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (style)
$ git status
On branch style
nothing to commit, working tree clean
```

{#fig:070 width=100%}

Команда git status сообщает о том, что в данный момент мы находимся в ветке «style».

Добавим файл стилей style.css. Выполним (рис. [-@fig:071]):

```
h1 {
    color: red;
}
```

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (style)
$ touch lib/style.css
```

{#fig:071 width=100%}

Проиндексируем его и выполним коммит. (рис. [-@fig:072]):

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (style)
$ git add lib/style.css

dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (style)
$ git commit -m "Added css stylesheet"
[style 67c6e60] Added css stylesheet
 1 file changed, 3 insertions(+)
 create mode 100644 lib/style.css
```

{#fig:072 width=100%}

Теперь изменим основную страницу. Обновим файл hello.html, чтобы использовать стили style.css.

```
<!-- Author: Arsenij A. Ilinskij ( ilinskyar@gmail.com ) -->
<html>
    <head>
        <link type="text/css" rel="stylesheet"
              media="all" href="style.css" />
    </head>
    <body>
        <h1>Hello, World!</h1>
    </body>
</html>
```

Проиндексируем его и выполним коммит. (рис. [-@fig:073]):

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (style)
$ git add lib/hello.html

dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (style)
$ git commit -m "Hello uses style.css"
[style 5a9b91b] Hello uses style.css
 1 file changed, 1 insertion(+)
```

{#fig:073 width=100%}

Теперь обновим файл index.html, чтобы он тоже использовал style.css.

```
<html>
  <head>
    <link type="text/css" rel="stylesheet"
      media="all" href="lib/style.css" />
  </head>
  <body>
    <iframe src="lib/hello.html" width="200" height="200" />
  </body>
</html>
```

Проиндексируем его и выполним коммит. (рис. [-@fig:074]):

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (style)
$ git add index.html

dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (style)
$ git commit -m "Updated index.html"
[style d9ac061] Updated index.html
 1 file changed, 4 insertions(+), 1 deletion(-)
```

{#fig:074 width=100%}

Теперь при открытии index.html, мы видим кусок страницы hello в маленьком окошке с красным цветом шрифта. (рис. [-@fig:075])



{#fig:075 width=100%}

## Навигация по веткам

Теперь в нашем проекте есть две ветки. Выполним (рис. [-@fig:076]):

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (style)
$ git log --all
commit d9ac061bc0b57a961b665abdc190aa57c4a6c968 (HEAD -> style)
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date:   Thu Feb 10 22:14:28 2022 +0300

    Updated index.html

commit 5a9b91b6324b09d1f3b3cdb2ab5cd704c4944836
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date:   Thu Feb 10 22:12:51 2022 +0300

    Hello uses style.css

commit 67c6e603ca0ebcd858d4167ae7ba1949785d52be
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date:   Thu Feb 10 22:09:07 2022 +0300

    Added css stylesheet

commit 6564b23673a7381583964e75191a2aa821e5b67a (main)
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date:   Thu Feb 10 21:54:06 2022 +0300

    Added index.html

commit 58e4685b1a9bd6b5ab41cf53f25161efd81d7a6f
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date:   Thu Feb 10 21:34:35 2022 +0300

    Moved hello.html to lib

commit 37294a83a3b46fe829627e0378da1943d0528c03
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date:   Thu Feb 10 21:29:58 2022 +0300

    Add an author/email comment

commit b840713180dfb70179447e6ca7a8d2db8d77d3d0 (tag: v1)
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date:   Thu Feb 10 21:00:48 2022 +0300
    I
    Added HTML header

commit b38204d2e0d18817d4ec693756bcc6378002ba9b (tag: v1-beta)
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date:   Thu Feb 10 20:59:57 2022 +0300

    Added standard HTML page tags
```

{#fig:076 width=100%}

Используя команду git checkout переключимся на ветку main. (рис. [-@fig:077])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (style)
$ git checkout main
Switched to branch 'main'

dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ cat lib/hello.html
<!-- Author: Arsenij A. Ilinskij ( ilinskiyar@gmail.com ) -->
<html>
<head>
</head>
<body>
<h1>Hello, World!</h1>
</body>
</html>
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ git checkout style
Switched to branch 'style'
```

{#fig:077 width=100%}

Сейчас мы находимся на ветке main. Это заметно по тому, что файл hello.html не использует стили style.css.

Теперь вернемся к ветке style. Выполним (рис. [-@fig:078]):

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/вуз - образование/Математическое моделирование/lab1/hello (main)
$ git checkout style
Switched to branch 'style'

dark2@LAPTOP-6ITRB0QA MINGW64 /d/вуз - образование/Математическое моделирование/lab1/hello (style)
$ cat lib/hello.html
<!-- Author: Arsenij A. Ilinskij ( ilinskiyar@gmail.com ) -->
<html>
  <head>
    <link type="text/css" rel="stylesheet" media="all" href="style.css" />
  </head>
  <body>
    <h1>Hello, world!</h1>
  </body>
</html>
```

{#fig:078 width=100%}

Содержимое lib/hello.html подтверждает, что мы вернулись на ветку style.

## Изменения в ветке main

Допустим, пока мы меняли ветку style, кто-то решил обновить ветку main. Они добавили файл README.md.

Создадим файл README в ветке main. Выполним (рис. [-@fig:079]):

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/вуз - образование/Математическое моделирование/lab1/hello (style)
$ git checkout main
Switched to branch 'main'

dark2@LAPTOP-6ITRB0QA MINGW64 /d/вуз - образование/Математическое моделирование/lab1/hello (main)
$ touch README.md
```

{#fig:079 width=100%}

Добавим в него содержимое *This is the Hello World example from the git tutorial.* (рис. [-@fig:080]):

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/вуз - образование/Математическое моделирование/lab1/hello (main)
$ echo "This is the Hello World example from the git tutorial." > README.md
```

{#fig:080 width=100%}

## Сделаем коммит изменений README.md в ветку main

Проиндексируем файл README.md и выполним коммит. (рис. [-@fig:081]):

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/вуз - образование/Математическое моделирование/lab1/hello (main)
$ git add README.md

dark2@LAPTOP-6ITRB0QA MINGW64 /d/вуз - образование/Математическое моделирование/lab1/hello (main)
$ git commit -m "Added README"
[main 81e271b] Added README
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
```

{#fig:081 width=100%}

Просмотрим отличающиеся ветки. Теперь у нас в репозитории есть две отличающиеся ветки.

Просмотрим ветки и их различия, выполнив (рис. [-@fig:082]):

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ git log --graph --all
* commit 81e271b4706c3f692f48604696d42d1b41c78de8 (HEAD -> main)
| Author: arseniy ilyinsky <ilinskiyar@gmail.com>
| Date:   Thu Feb 10 22:33:32 2022 +0300
|
|     Added README
|
* commit d9ac061bc0b57a961b665abdc190aa57c4a6c968 (style)
| Author: arseniy ilyinsky <ilinskiyar@gmail.com>
| Date:   Thu Feb 10 22:14:28 2022 +0300
|
|     Updated index.html
|
* commit 5a9b91b6324b09d1f3b3cdb2ab5cd704c4944836
| Author: arseniy ilyinsky <ilinskiyar@gmail.com>
| Date:   Thu Feb 10 22:12:51 2022 +0300
|
|     Hello uses style.css
|
* commit 67c6e603ca0ebcd858d4167ae7ba1949785d52be
| Author: arseniy ilyinsky <ilinskiyar@gmail.com>
| Date:   Thu Feb 10 22:09:07 2022 +0300
|
|     Added css stylesheet
|
* commit 6564b23673a7381583964e75191a2aa821e5b67a
| Author: arseniy ilyinsky <ilinskiyar@gmail.com>
| Date:   Thu Feb 10 21:54:06 2022 +0300
|
|     Added index.html
|
* commit 58e4685b1a9bd6b5ab41cf53f25161efd81d7a6f
| Author: arseniy ilyinsky <ilinskiyar@gmail.com>
| Date:   Thu Feb 10 21:34:35 2022 +0300
|
|     Moved hello.html to lib
|
* commit 37294a83a3b46fe829627e0378da1943d0528c03
| Author: arseniy ilyinsky <ilinskiyar@gmail.com>
| Date:   Thu Feb 10 21:29:58 2022 +0300
|
|     Add an author/email comment
```

{#fig:082 width=100%}

Добавление опции `--graph` в `git log` вызывает построение дерева коммитов с помощью простых ASCII символов. Мы видим обе ветки (`style` и `main`), и то, что ветка `main` является текущей HEAD. Общим предшественником обеих веток является коммит «Added index.html». Опция `--all` гарантированно означает, что мы видим все ветки. По умолчанию показывается только текущая ветка.

## Слияние

Слияние переносит изменения из двух веток в одну. Вернемся к ветке `style` и сольем `main` с `style`, выполнив (рис. [-@fig:083], рис. [-@fig:084]):

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ git checkout style
Switched to branch 'style'

dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (style)
$ git merge main
Merge made by the 'recursive' strategy.
 README.md | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
```

{#fig:083 width=100%}

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (style)
$ git log --graph --all
* commit f997fa754cd934455fc00f6d2fba2b48f3980a60 (HEAD -> style)
| \ Merge: d9ac061 81e271b
| Author: arseniy ilyinsky <ilinskiyar@gmail.com>
| Date: Thu Feb 10 22:35:39 2022 +0300
|
|     Merge branch 'main' into style
|
* commit 81e271b4706c3f692f48604696d42d1b41c78de8 (main)
| Author: arseniy ilyinsky <ilinskiyar@gmail.com>
| Date: Thu Feb 10 22:33:32 2022 +0300
|
|     Added README
|
* commit d9ac061bc0b57a961b665abdc190aa57c4a6c968
| Author: arseniy ilyinsky <ilinskiyar@gmail.com>
| Date: Thu Feb 10 22:14:28 2022 +0300
|
|     Updated index.html
|
* commit 5a9b91b6324b09d1f3b3cdb2ab5cd704c4944836
| Author: arseniy ilyinsky <ilinskiyar@gmail.com>
| Date: Thu Feb 10 22:12:51 2022 +0300
|
|     Hello uses style.css
|
* commit 67c6e603ca0ebcd858d4167ae7ba1949785d52be
| Author: arseniy ilyinsky <ilinskiyar@gmail.com>
| Date: Thu Feb 10 22:09:07 2022 +0300
|
|     Added css stylesheet
|
* commit 6564b23673a7381583964e75191a2aa821e5b67a
| Author: arseniy ilyinsky <ilinskiyar@gmail.com>
| Date: Thu Feb 10 21:54:06 2022 +0300
|
|     Added index.html
|
* commit 58e4685b1a9bd6b5ab41cf53f25161efd81d7a6f
| Author: arseniy ilyinsky <ilinskiyar@gmail.com>
| Date: Thu Feb 10 21:34:35 2022 +0300
|
|     Moved hello.html to lib
```

{#fig:084 width=100%}

Путем периодического слияния ветки main с веткой style мы можем переносить из main любые изменения и поддерживать совместимость изменений style с изменениями в основной ветке.

## Создание конфликта

Но что если изменения в ветке main конфликтуют с изменениями в style?

Вернемся в ветку main. (рис. [-@fig:085])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (style)
$ git checkout main
Switched to branch 'main'
```

{#fig:085 width=100%}

И внесем следующие изменения в файл hello.html. (рис. [-@fig:086])

```
<!-- Author: Arsenij A. Ilinskij ( ilinskiyar@gmail.com ) -->
<html>
  <head>
    <!-- no style -->
```

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

Проиндексируем его и выполним коммит. (рис. [-@fig:086])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ git add lib/hello.html

dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ git commit -m 'Life is great'
[main 7c8c6b1] Life is great
 1 file changed, 3 insertions(+), 2 deletions(-)
```

{#fig:086 width=100%}

Теперь просмотрим ветки, выполнив (рис. [-@fig:087]):

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (style)
$ git log --graph --all
* commit 7c8c6b16ad1b124206cf46aaaf92ffbbae96f311f (HEAD -> main)
  Author: arseniy ilyinsky <ilinskiyar@gmail.com>
  Date:   Thu Feb 10 22:39:27 2022 +0300

    Life is great

* commit f997fa754cd934455fc00f6d2fba2b48f3980a60 (style)
  Merge: d9ac061 81e271b
  Author: arseniy ilyinsky <ilinskiyar@gmail.com>
  Date:   Thu Feb 10 22:35:39 2022 +0300

    Merge branch 'main' into style

* commit 81e271b4706c3f692f48604696d42d1b41c78de8
  Author: arseniy ilyinsky <ilinskiyar@gmail.com>
  Date:   Thu Feb 10 22:33:32 2022 +0300

    Added README

* commit d9ac061bc0b57a961b665abdc190aa57c4a6c968
  Author: arseniy ilyinsky <ilinskiyar@gmail.com>
  Date:   Thu Feb 10 22:14:28 2022 +0300

    Updated index.html

* commit 5a9b91b6324b09d1f3b3cdb2ab5cd704c4944836
  Author: arseniy ilyinsky <ilinskiyar@gmail.com>
  Date:   Thu Feb 10 22:12:51 2022 +0300

    Hello uses style.css

* commit 67c6e603ca0ebcd858d4167ae7ba1949785d52be
  Author: arseniy ilyinsky <ilinskiyar@gmail.com>
  Date:   Thu Feb 10 22:09:07 2022 +0300

    Added css stylesheet
```

{#fig:087 width=100%}

После коммита «Added README» ветка main была объединена с веткой style, но в настоящее время в main есть дополнительный коммит, который не был слит с style. Последнее изменение в main конфликтует с некоторыми изменениями в style.

## Разрешение конфликтов

Теперь вернемся к ветке style и попытаемся объединить ее с новой веткой main, выполнив (рис. [-@fig:088]):

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ git checkout style
Switched to branch 'style'

dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (style)
$ git merge main
Auto-merging lib/hello.html
CONFLICT (content): Merge conflict in lib/hello.html
Automatic merge failed; fix conflicts and then commit the result.
```

{#fig:088 width=100%}

Теперь, если мы откроем файл lib/hello.html, то мы должны увидеть (рис. [-@fig:089]):

```
<!-- Author: Arsenij A. Ilinskij ( ilinskyar@gmail.com ) -->
<html>
  <head>
    <<<<< HEAD
      <link type="text/css" rel="stylesheet" media="all" href="style.css" />
    =====
      <!-- no style -->
    >>>>> main
    </head>
    <body>
      <h1>Hello, World! Life is great!</h1>
    </body>
  </html>
```

{#fig:089 width=100%}

Можно увидеть, что:

- Первый раздел — версия текущей ветки (style).
- Второй раздел — версия ветки main.

Для разрешения конфликта, нужно вручную решить проблему, а именно внести изменения в lib/hello.html. (рис. [-@fig:090])

```
<!-- Author: Arsenij A. Ilinskij ( ilinskyar@gmail.com ) -->
<html>
  <head>
    <link type="text/css" rel="stylesheet" media="all" href="style.css" />
  </head>
  <body>
    <h1>Hello, World! Life is great!</h1>
  </body>
</html>
```

{#fig:090 width=100%}

Сделаем коммит решения конфликта. (рис. [-@fig:091])

```
dark2@LAPTOP-6ITRBOQA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (style|MERGING)
$ git add lib/hello.html

dark2@LAPTOP-6ITRBOQA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (style|MERGING)
$ git commit -m "Merged main fixed conflict."
[style 340f7f7] Merged main fixed conflict.
```

{#fig:091 width=100%}

## Сброс ветки style

Вернемся на ветку style к точке перед тем, как мы слили ее с веткой main. Мы можем сбросить ветку к любому коммиту. По сути, это изменение указателя ветки на любую точку дерева коммитов. В этом случае мы хотим вернуться в ветку style в точку перед слиянием с main. Нам необходимо найти последний коммит перед слиянием, для этого выполним (рис. [-@fig:092]):

```
dark2@LAPTOP-6ITRBOQA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (style)
$ git checkout style
Already on 'style'

dark2@LAPTOP-6ITRBOQA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (style)
$ git log --graph
* commit 340f7f77a6381f1ee2b41582dc56819e629821c (HEAD -> style)
|\ Merge: f997fa7 7c8c6b1
| Author: arseniy ilyinsky <ilinskiiyar@gmail.com>
| Date:   Thu Feb 10 22:42:10 2022 +0300
|
|     Merged main fixed conflict.

* commit 7c8c6b16ad1b124206cf46aaf92ffbbae96f311f (main)
| Author: arseniy ilyinsky <ilinskiiyar@gmail.com>
| Date:   Thu Feb 10 22:39:27 2022 +0300
|
|     Life is great

* commit f997fa754cd934455fc00f6d2fba2b48f3980a60
| Merge: d9ac061 81e271b
| Author: arseniy ilyinsky <ilinskiiyar@gmail.com>
| Date:   Thu Feb 10 22:35:39 2022 +0300
|
|     Merge branch 'main' into style

* commit 81e271b4706c3f692f48604696d42d1b41c78de8
| Author: arseniy ilyinsky <ilinskiiyar@gmail.com>
| Date:   Thu Feb 10 22:33:32 2022 +0300
|
|     Added README

* commit d9ac061bc0b57a961b665abdc190aa57c4a6c968
| Author: arseniy ilyinsky <ilinskiiyar@gmail.com>
| Date:   Thu Feb 10 22:14:28 2022 +0300
|
|     Updated index.html

* commit 5a9b91b6324b09d1f3b3cdb2ab5cd704c4944836
| Author: arseniy ilyinsky <ilinskiiyar@gmail.com>
| Date:   Thu Feb 10 22:12:51 2022 +0300
|
|     Hello uses style.css
```

{#fig:092 width=100%}

Мы видим, что коммит «Updated index.html» ( хэш(d9ac061...) ) был последним на ветке style перед слиянием, поэтому сбросим ветку style к этому коммиту. (рис. [-@fig:093])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (style)
$ git reset --hard "d9ac061bc0b57a961b665abdc190aa57c4a6c968"
HEAD is now at d9ac061 Updated index.html
```

{#fig:093 width=100%}

Теперь проверим ветку. Найдем лог ветки style. (рис. [-@fig:094])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (style)
$ git log --graph --all
* commit 7c8c6b16ad1b124206cf46aaaf92ffbbae96f311f (main)
  Author: arseniy ilyinsky <ilinskiyar@gmail.com>
  Date:   Thu Feb 10 22:39:27 2022 +0300

    Life is great

* commit 81e271b4706c3f692f48604696d42d1b41c78de8
  Author: arseniy ilyinsky <ilinskiyar@gmail.com>
  Date:   Thu Feb 10 22:33:32 2022 +0300

    Added README

* commit d9ac061bc0b57a961b665abdc190aa57c4a6c968 (HEAD -> style)
  Author: arseniy ilyinsky <ilinskiyar@gmail.com>
  Date:   Thu Feb 10 22:14:28 2022 +0300

    Updated index.html

* commit 5a9b91b6324b09d1f3b3cdb2ab5cd704c4944836
  Author: arseniy ilyinsky <ilinskiyar@gmail.com>
  Date:   Thu Feb 10 22:12:51 2022 +0300
    Hello uses style.css
    [ ... ]

* commit 67c6e603ca0ebcd858d4167ae7ba1949785d52be
  Author: arseniy ilyinsky <ilinskiyar@gmail.com>
  Date:   Thu Feb 10 22:09:07 2022 +0300

    Added css stylesheet

* commit 6564b23673a7381583964e75191a2aa821e5b67a
  Author: arseniy ilyinsky <ilinskiyar@gmail.com>
  Date:   Thu Feb 10 21:54:06 2022 +0300

    Added index.html

* commit 58e4685b1a9bd6b5ab41cf53f25161efd81d7a6f
  Author: arseniy ilyinsky <ilinskiyar@gmail.com>
  Date:   Thu Feb 10 21:34:35 2022 +0300
    Moved hello.html to lib
```

{#fig:094 width=100%}

Можно заметить, что у нас в истории больше нет коммитов слияний.

## Сброс ветки main

Добавив интерактивный режим в ветку main, мы внесли изменения, конфликтующие с изменениями в ветке style. Давайте вернемся в ветку main в точку перед внесением конфликтующих изменений. Для этого выполним (рис. [-@fig:095]):

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (style)
$ git checkout main
Switched to branch 'main'

dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ git log --graph
* commit 7c8c6b16ad1b124206cf46aaaf92ffbbae96f311f (HEAD -> main)
| Author: arseniy ilyinsky <ilinskayar@gmail.com>
| Date:   Thu Feb 10 22:39:27 2022 +0300
|
|     Life is great
|
* commit 81e271b4706c3f692f48604696d42d1b41c78de8
| Author: arseniy ilyinsky <ilinskayar@gmail.com>
| Date:   Thu Feb 10 22:33:32 2022 +0300
|
|     Added README
|
* commit 6564b23673a7381583964e75191a2aa821e5b67a
| Author: arseniy ilyinsky <ilinskayar@gmail.com>
| Date:   Thu Feb 10 21:54:06 2022 +0300
|
|     Added index.html
|
* commit 58e4685b1a9bd6b5ab41cf53f25161efd81d7a6f
| Author: arseniy ilyinsky <ilinskayar@gmail.com>
| Date:   Thu Feb 10 21:34:35 2022 +0300
|
|     Moved hello.html to lib
|
* commit 37294a83a3b46fe829627e0378da1943d0528c03
| Author: arseniy ilyinsky <ilinskayar@gmail.com>
| Date:   Thu Feb 10 21:29:58 2022 +0300
|
|     Add an author/email comment
|
* commit b840713180dfb70179447e6ca7a8d2db8d77d3d0 (tag: v1)
| Author: arseniy ilyinsky <ilinskayar@gmail.com>
| Date:   Thu Feb 10 21:00:48 2022 +0300
|
|     Added HTML header
```

{#fig:095 width=100%}

Коммит «Added README» идет непосредственно перед коммитом конфликтующего интерактивного режима, поэтому мы сбросим ветку main к коммиту «Added README» ( хэш(81e271b..) ). (рис. [-@fig:096])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ git reset --hard "81e271b4706c3f692f48604696d42d1b41c78de8"
HEAD is now at 81e271b Added README

dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ git log --graph --all
* commit 81e271b4706c3f692f48604696d42d1b41c78de8 (HEAD -> main)
| Author: arseniy ilyinsky <ilinskiyar@gmail.com>
| Date:   Thu Feb 10 22:33:32 2022 +0300
|
|     Added README
|
* commit d9ac061bc0b57a961b665abdc190aa57c4a6c968 (style)
| Author: arseniy ilyinsky <ilinskiyar@gmail.com>
| Date:   Thu Feb 10 22:14:28 2022 +0300
|
|     Updated index.html
|
* commit 5a9b91b6324b09d1f3b3cdb2ab5cd704c4944836
| Author: arseniy ilyinsky <ilinskiyar@gmail.com>
| Date:   Thu Feb 10 22:12:51 2022 +0300
|
|     Hello uses style.css
|
* commit 67c6e603ca0ebcd858d4167ae7ba1949785d52be
| Author: arseniy ilyinsky <ilinskiyar@gmail.com>
| Date:   Thu Feb 10 22:09:07 2022 +0300
|
|     Added css stylesheet
|
* commit 6564b23673a7381583964e75191a2aa821e5b67a
| Author: arseniy ilyinsky <ilinskiyar@gmail.com>
| Date:   Thu Feb 10 21:54:06 2022 +0300
|
|     Added index.html
|
* commit 58e4685b1a9bd6b5ab41cf53f25161efd81d7a6f
| Author: arseniy ilyinsky <ilinskiyar@gmail.com>
| Date:   Thu Feb 10 21:34:35 2022 +0300
|
|     Moved hello.html to lib
```

{#fig:096 width=100%}

Посмотрев лог, можно прийти к выводу что, как будто репозиторий был перемотан назад во времени к точке до какого-либо слияния.

## Перебазирование

Рассмотрим различия между слиянием и перебазированием. Для того, чтобы это сделать, нам нужно вернуться в репозиторий в момент до первого слияния ( что мы и сделали в предыдущих 2ух пунктах ), а затем повторить те же действия, но с использованием перебазирования вместо слияния.

Используем команду `rebase` вместо команды `merge`. Мы вернулись в точку до первого слияния и хотим перенести изменения из ветки `main` в нашу ветку `style`. На этот раз для переноса изменений из ветки `main` мы будем использовать команду `git rebase` вместо слияния. (рис. [-@fig:097])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ git checkout style
Switched to branch 'style'

dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (style)
$ git rebase main
Successfully rebased and updated refs/heads/style.

dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (style)
$ git log --graph
* commit 1f21f230c542cdb289a80add22eb5dd3e018e266 (HEAD -> style)
| Author: arseniy ilyinsky <ilinskiyar@gmail.com>
| Date:   Thu Feb 10 22:14:28 2022 +0300
|
|     Updated index.html
|
* commit 3a2106acda3bd8bd4f17add116a8a1db0d8c6f98
| Author: arseniy ilyinsky <ilinskiyar@gmail.com>
| Date:   Thu Feb 10 22:12:51 2022 +0300
|
|     Hello uses style.css
|
* commit 252443ed3e1893ac880a7f1db809e0e594df8f34
| Author: arseniy ilyinsky <ilinskiyar@gmail.com>
| Date:   Thu Feb 10 22:09:07 2022 +0300
|
|     Added css stylesheet
|
* commit 81e271b4706c3f692f48604696d42d1b41c78de8 (main)
| Author: arseniy ilyinsky <ilinskiyar@gmail.com>
| Date:   Thu Feb 10 22:33:32 2022 +0300
|
|     Added README
|
* commit 6564b23673a7381583964e75191a2aa821e5b67a
| Author: arseniy ilyinsky <ilinskiyar@gmail.com>
| Date:   Thu Feb 10 21:54:06 2022 +0300
|
|     Added index.html
|
* commit 58e4685b1a9bd6b5ab41cf53f25161efd81d7a6f
| Author: arseniy ilyinsky <ilinskiyar@gmail.com>
| Date:   Thu Feb 10 21:34:35 2022 +0300
|
|     Moved hello.html to lib
|
* commit 37294a83a3b46fe829627e0378da1943d0528c03
| Author: arseniy ilyinsky <ilinskiyar@gmail.com>
| Date:   Thu Feb 10 21:29:58 2022 +0300
|
|     Add an author/email comment
```

{#fig:097 width=100%}

Конечный результат перебазирования 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.

## Слияние в ветку `main`

Мы поддерживали соответствие ветки `style` с веткой `main` (с помощью `rebase`), теперь давайте сольем изменения `style` в ветку `main`, выполнив (рис. [-@fig:098]):

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (style)
$ git checkout main
Switched to branch 'main'

dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ git merge style
Updating 81e271b..1f21f23
Fast-forward
 index.html    | 5 ++++-
 lib/hello.html | 1 +
 lib/style.css  | 3 ===
 3 files changed, 8 insertions(+), 1 deletion(-)
 create mode 100644 lib/style.css
```

{#fig:098 width=100%}

Поскольку последний коммит ветки main прямо предшествует последнему коммиту ветки style, git может выполнить ускоренное слияние-перемотку. При быстрой перемотке конфликтов быть не может.

Посмотрим логи, выполнив (рис. [-@fig:099]):

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ git log
commit 1f21f230c542cdb289a80add22eb5dd3e018e266 (HEAD -> main, style)
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date:   Thu Feb 10 22:14:28 2022 +0300

    Updated index.html

commit 3a2106acda3bd8bd4f17add116a8a1db0d8c6f98
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date:   Thu Feb 10 22:12:51 2022 +0300

    Hello uses style.css

commit 252443ed3e1893ac880a7f1db809e0e594df8f34
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date:   Thu Feb 10 22:09:07 2022 +0300

    Added css stylesheet

commit 81e271b4706c3f692f48604696d42d1b41c78de8
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date:   Thu Feb 10 22:33:32 2022 +0300

    Added README

commit 6564b23673a7381583964e75191a2aa821e5b67a
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date:   Thu Feb 10 21:54:06 2022 +0300

    Added index.html

commit 58e4685b1a9bd6b5ab41cf53f25161efd81d7a6f
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date:   Thu Feb 10 21:34:35 2022 +0300

    Moved hello.html to lib

commit 37294a83a3b46fe829627e0378da1943d0528c03
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date:   Thu Feb 10 21:29:58 2022 +0300

    Add an author/email comment
```

{#fig:099 width=100%}

Теперь ветки style и main идентичны.

## Клонирование репозиториев

Перейдем в рабочий каталог и сделаем клон нашего репозитория hello.

Для этого перейдем в «рабочий» каталог. (рис. [-@fig:100])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/вуз - образование/Математическое моделирование/lab1/hello (main)
$ cd ..

dark2@LAPTOP-6ITRB0QA MINGW64 /d/вуз - образование/Математическое моделирование/lab1
$ pwd
/d/вуз - образование/Математическое моделирование/lab1

dark2@LAPTOP-6ITRB0QA MINGW64 /d/вуз - образование/Математическое моделирование/lab1
$ ls
hello/
```

{#fig:100 width=100%}

Теперь создадим клон репозитория. (рис. [-@fig:101])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/вуз - образование/Математическое моделирование/lab1
$ git clone hello cloned_hello
Cloning into 'cloned_hello'...
done.
```

{#fig:101 width=100%}

В вашем «рабочем» каталоге теперь должно быть два репозитория: оригинальный репозиторий «hello» и клонированный репозиторий «cloned\_hello». (рис. [-@fig:102])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/вуз - образование/Математическое моделирование/lab1
$ ls
cloned_hello/ hello/
```

{#fig:102 width=100%}

## Просмотр клонированного репозитория

Давайте взглянем на клонированный репозиторий, выполнив (рис. [-@fig:103]):

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/вуз - образование/Математическое моделирование/lab1
$ cd cloned_hello

dark2@LAPTOP-6ITRB0QA MINGW64 /d/вуз - образование/Математическое моделирование/lab1/cloned_hello (main)
$ ls
README.md index.html lib/
```

{#fig:103 width=100%}

Мы увидим список всех файлов на верхнем уровне оригинального репозитория README.md, index.html и lib.

Давайте просмотрим историю репозитория, выполнив (рис. [-@fig:104]):

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/cloned_hello (main)
$ git log --all
commit 1f21f230c542cdb289a80add22eb5dd3e018e266 (HEAD -> main, origin/style, origin/main, origin/HEAD)
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date:   Thu Feb 10 22:14:28 2022 +0300

    Updated index.html

commit 3a2106acda3bd8bd4f17add116a8a1db0d8c6f98
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date:   Thu Feb 10 22:12:51 2022 +0300

    Hello uses style.css

commit 252443ed3e1893ac880a7f1db809e0e594df8f34
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date:   Thu Feb 10 22:09:07 2022 +0300

    Added css stylesheet

commit 81e271b4706c3f692f48604696d42d1b41c78de8
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date:   Thu Feb 10 22:33:32 2022 +0300

    Added README

commit 6564b23673a7381583964e75191a2aa821e5b67a
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date:   Thu Feb 10 21:54:06 2022 +0300

    Added index.html

commit 58e4685b1a9bd6b5ab41cf53f25161efd81d7a6f
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date:   Thu Feb 10 21:34:35 2022 +0300

    Moved hello.html to lib
```

{#fig:104 width=100%}

Мы видим список всех коммитов в новый репозитории, и более / менее совпадает с историей коммитов в оригинальном репозитории.

Однако ветки main (HEAD) мы можем увидеть ветки со странными именами (origin/main, origin/style и origin/HEAD).

## Что такое origin?

Выполнив (рис. [-@fig:104]):

```
git remote
```

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/cloned_hello (main)
$ git remote
origin
```

{#fig:105 width=100%}

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, выполнив (рис. [-@fig:105]):

```
git remote show origin
```

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/cloned_hello (main)
$ git remote show origin
* remote origin
  Fetch URL: D:/ВУЗ - образование/Математическое моделирование/lab1/hello
  Push URL: D:/ВУЗ - образование/Математическое моделирование/lab1/hello
  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)
```

{#fig:106 width=100%}

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.

## Удаленные ветки

Давайте посмотрим на ветки, доступные в нашем клонированном репозитории. (рис. [-@fig:107])

```
git branch
```

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/cloned_hello (main)
$ git branch
* main
```

{#fig:107 width=100%}

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, попробуем следующую команду (рис. [-@fig:108]):

```
git branch -a
```

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/cloned_hello (main)
$ git branch -a
* main
  remotes/origin/HEAD -> origin/main
  remotes/origin/main
  remotes/origin/style
```

{#fig:108 width=100%}

Git выводит все коммиты в оригинальный репозиторий, но ветки в удаленном репозитории не рассматриваются как локальные.

## Изменение оригинального репозитория

Внесем некоторые изменения в оригинальный репозиторий hello, чтобы затем попытаться извлечь и слить изменения из удаленной ветки в текущую.

Для этого перейдем в оригинальный репозиторий hello. (рис. [-@fig:109])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/cloned_hello (main)
$ cd ../hello
```

{#fig:109 width=100%}

Внесем следующие изменения в файла README.md. (рис. [-@fig:110])

```
This is the Hello World example from the git tutorial.
```

Теперь добавим это изменение и сделаем коммит. (рис. [-@fig:111])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ git add README.md

dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ git commit -m "Changed README is original repo"
[main 1ac5bda] Changed README is original repo
 1 file changed, 1 insertion(+), 1 deletion(-)
```

{#fig:111 width=100%}

Таким образом, в оригинальном репозитории есть более поздние изменения, которых нет в клонированной версии.

Теперь научимся извлекать эти изменения из удаленного репозитория, выполнив (рис. [-@fig:112], рис. [-@fig:113]):

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/hello (main)
$ cd ../cloned_hello

dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/cloned_hello (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 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 338 bytes | 1024 bytes/s, done.
From D:/ВУЗ - образование/Математическое моделирование/lab1/hello
  1f21f23..1ac5bda  main      -> origin/main
```

{#fig:112 width=100%}

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/cloned_hello (main)
$ git log --all
commit 1ac5bda84b806f6e93bcfd2e9e0c7bfcc971bc4a (origin/main, origin/HEAD)
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date: Thu Feb 10 23:01:41 2022 +0300

    Changed README is original repo

commit 1f21f230c542cdb289a80add22eb5dd3e018e266 (HEAD -> main, origin/style)
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date: Thu Feb 10 22:14:28 2022 +0300

    Updated index.html

commit 3a2106acda3bd8bd4f17add116a8a1db0d8c6f98
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date: Thu Feb 10 22:12:51 2022 +0300

    Hello uses style.css

commit 252443ed3e1893ac880a7f1db809e0e594df8f34
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date: Thu Feb 10 22:09:07 2022 +0300

    Added css stylesheet

commit 81e271b4706c3f692f48604696d42d1b41c78de8
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date: Thu Feb 10 22:33:32 2022 +0300

    Added README

commit 6564b23673a7381583964e75191a2aa821e5b67a
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date: Thu Feb 10 21:54:06 2022 +0300

    Added index.html
```

{#fig:113 width=100%}

Сейчас мы находимся в репозитории cloned\_hello. На данный момент в репозитории есть все коммиты из оригинального репозитория, но они не интегрированы в локальные ветки клонированного репозитория.

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

Теперь давайте посмотрим на коммит «Updated index.html». Можно увидеть, что локальная ветка main указывает на этот коммит, а не на новый коммит, который мы только что извлекли.

Теперь мы можем продемонстрировать, что клонированный файл README.md не изменился, выполнив (рис. [-@fig:114]):

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/cloned_hello (main)
$ cat README.md
This is the Hello World example from the git tutorial.
```

{#fig:114 width=100%}

## Слияние извлеченных изменений

Сольем извлеченные изменения в локальную ветку main, выполнив (рис. [-@fig:115]):

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/cloned_hello (main)
$ git merge origin/main
Updating 1f21f23..1ac5bda
Fast-forward
 README.md | 2 ++
 1 file changed, 1 insertion(+), 1 deletion(-)
```

{#fig:115 width=100%}

Сейчас мы должны увидеть изменения, для этого выполним (рис. [-@fig:116]):

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/cloned_hello (main)
$ cat README.md
This is the Hello World example from the git tutorial.
```

{#fig:116 width=100%}

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

Теперь давайте рассмотрим объединение fetch и merge в одну команду. (рис. [-@fig:117])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/cloned_hello (main)
$ git pull
Already up to date.
```

{#fig:117 width=100%}

Это эквивалентно двум следующим шагам:

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

## Добавление ветки наблюдения

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

Добавим локальную ветку, которая отслеживает удаленную ветку, для этого выполним следующие команды (рис. [-@fig:118]):

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/cloned_hello (main)
$ git branch --track style origin/style
Branch 'style' set up to track remote branch 'style' from 'origin'.

dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/cloned_hello (main)
$ git branch -a
* main
  style
    remotes/origin/HEAD -> origin/main
    remotes/origin/main
    remotes/origin/style

dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/cloned_hello (main)
$ git log --max-count=2
commit lac5bda84b806f6e93bcd2e9e0c7bfee971bc4a (HEAD -> main, origin/main, origin/HEAD)
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date:   Thu Feb 10 23:01:41 2022 +0300

    Changed README is original repo

commit 1f21f230c542cdb289a80add22eb5dd3e018e266 (origin/style, style)
Author: arseniy ilyinsky <ilinskiyar@gmail.com>
Date:   Thu Feb 10 22:14:28 2022 +0300

    Updated index.html
```

{#fig:118 width=100%}

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

## Чистые репозитории

Чистые репозитории (без рабочих каталогов) обычно используются для расшаривания. Обычный git-репозиторий подразумевает, что вы будете использовать его как рабочую директорию, поэтому вместе с файлами проекта в актуальной версии, git хранит все служебные, «чисто-репозиториевые» файлы в поддиректории `.git`. В удаленных репозиториях нет смысла хранить рабочие файлы на диске (как это делается в рабочих копиях), а все что им действительно нужно — это дельты изменений и другие бинарные данные репозитория. Вот это и есть «чистый репозиторий».

## Создайте чистый репозиторий

Создадим чистый репозиторий. (рис. [-@fig:119])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/cloned_hello (main)
$ cd ..

dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1
$ git clone --bare hello hello.git
Cloning into bare repository 'hello.git'...
done.

dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1
$ ls hello.git
HEAD config description hooks/ info/ objects/ packed-refs refs/
```

{#fig:119 width=100%}

Сейчас мы находимся в рабочем каталоге. Как правило, репозитории, оканчивающиеся на `.git` являются чистыми репозиториями. Мы видим, что в репозитории `hello.git` нет рабочего каталога. По сути, это есть не что иное, как каталог `.git` нечистого репозитория.

## Добавление удаленного репозитория

Давайте добавим репозиторий `hello.git` к нашему оригинальному репозиторию. (рис. [-@fig:120])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/вуз - образование/Математическое моделирование/lab1
$ cd hello

dark2@LAPTOP-6ITRB0QA MINGW64 /d/вуз - образование/Математическое моделирование/lab1/hello (main)
$ git remote add shared ../hello.git
```

{#fig:120 width=100%}

## Отправка изменений

Так как чистые репозитории, как правило, расшариваются на каком-нибудь сетевом сервере, нам необходимо отправить наши изменения в другие репозитории. Начнем с создания изменения для отправки.

Отредактируем файл README.md.

```
This is the Hello World example from the git tutorial.
(Changed in the original and pushed to shared)
```

Теперь добавим это изменение и сделаем коммит. (рис. [-@fig:121])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/вуз - образование/Математическое моделирование/lab1/hello (main)
$ git add README.md

dark2@LAPTOP-6ITRB0QA MINGW64 /d/вуз - образование/Математическое моделирование/lab1/hello (main)
$ git commit -m "Added shared comment to readme"
[main 2b6fc37] Added shared comment to readme
 1 file changed, 2 insertions(+), 1 deletion(-)
```

{#fig:121 width=100%}

Теперь отправим эти изменения в общий репозиторий. (рис. [-@fig:122])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/вуз - образование/Математическое моделирование/lab1/hello (main)
$ git push shared main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 412 bytes | 412.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To ../hello.git
  1ac5bda..2b6fc37  main -> main
```

{#fig:122 width=100%}

Общим называется репозиторий, получающий отправленные нами изменения.

## Извлечение общих изменений

Теперь научимся извлекать изменения из общего репозитория.

Для этого переключимся в клонированный репозиторий. (рис. [-@fig:123])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/вуз - образование/Математическое моделирование/lab1/hello (main)
$ cd ../cloned_hello
```

{#fig:123 width=100%}

Далее извлечем изменения, только что отправленные в общий репозиторий. (рис. [-@fig:124])

```
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/cloned_hello (main)
$ git remote add shared ../hello.git

dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/cloned_hello (main)
$ git branch --track shared main
Branch 'shared' set up to track local branch 'main'.

dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/cloned_hello (main)
$ cat README.md
This is the Hello World example from the git tutorial.
dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/cloned_hello (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
Unpacking objects: 100% (3/3), 392 bytes | 0 bytes/s, done.
From ../hello
 * branch           main      -> FETCH_HEAD
 * [new branch]     main      -> shared/main
Updating 1ac5bda..2b6fc37
Fast-forward
 README.md | 3 +-+
 1 file changed, 2 insertions(+), 1 deletion(-)

dark2@LAPTOP-6ITRB0QA MINGW64 /d/ВУЗ - образование/Математическое моделирование/lab1/cloned_hello (main)
$ cat README.md
This is the Hello World example from the git tutorial.
(Changed in the original and pushed to shared)
```

{#fig:124 width=100%}

Как мы можем заметить, изменения были успешно извлечены.

## Выводы

---

Научился создавать репозитории в github, использовать Git Bash, загружать файлы и папки на GitHub с помощью Git Bash.

## Список литературы

---

- Кулабов Д. С. *Лабораторная работа №1*\*: git.pdf\*
- Кулабов Д. С. *Лабораторная работа №1*\*: markdown.pdf\*