Лабораторная работа №1

Работа с git

Акондзо Жордани Лади Гаэл

Содержание

# Цель работы

Приобретение практических навыков работы с ситемой управления версиями Git.

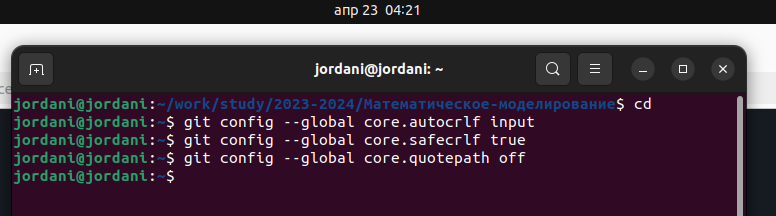
# Задание

Провести базовую настройку гит, создать проект и, используя его, изучить основные действия с репозиторием.

# Выполнение лабораторной работы

## Подготовка

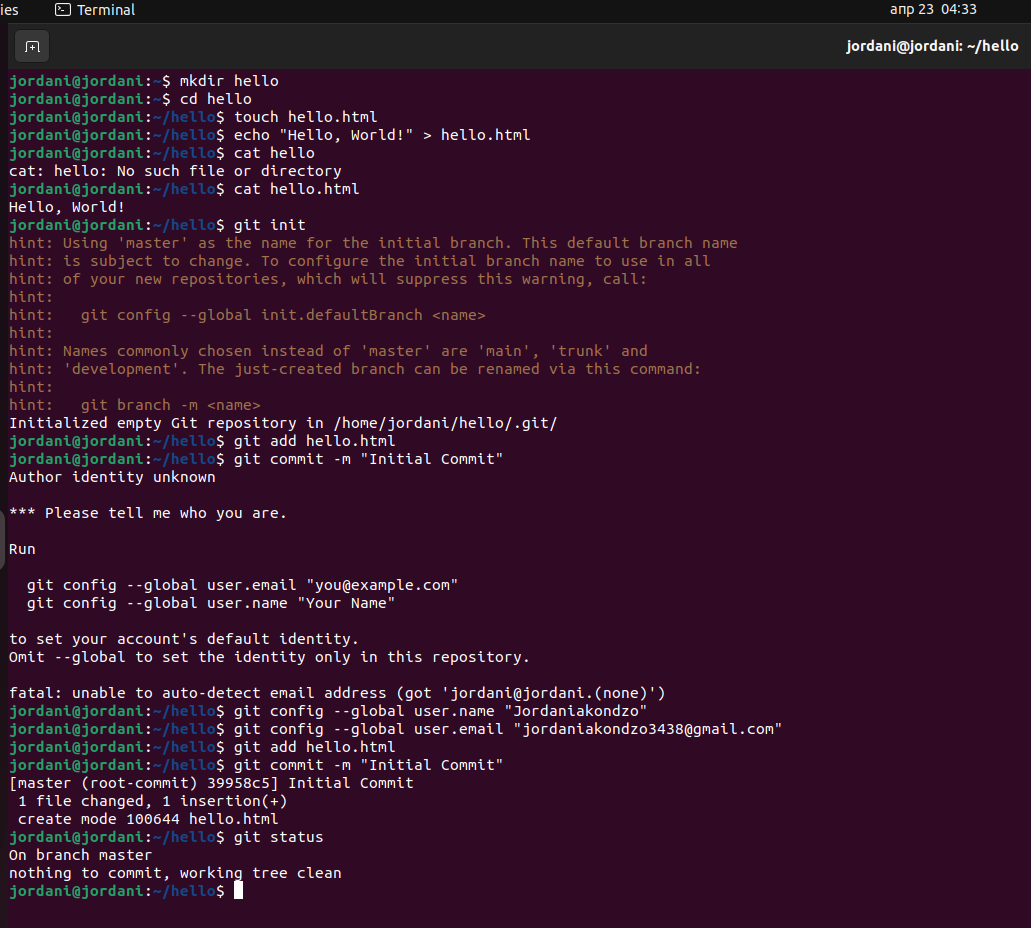
Сначала настроим core.autocrlf с параметрами true и input, чтобы сделать все переводы строк текстовых файлов в главном репозитории одинаковыми, а затем настроим отображение unicode(рис. @fig:001).



Настройка git

## Создание проекта

Создадим пустой каталог hello, а в нём файл с именем hello.html. Затем создадим git репозиторий из этого каталога, выполнив команду git init. Добавим файл в репозиторий и проверим статус, который сообщает, что коммитить нечего(рис. @fig:002).



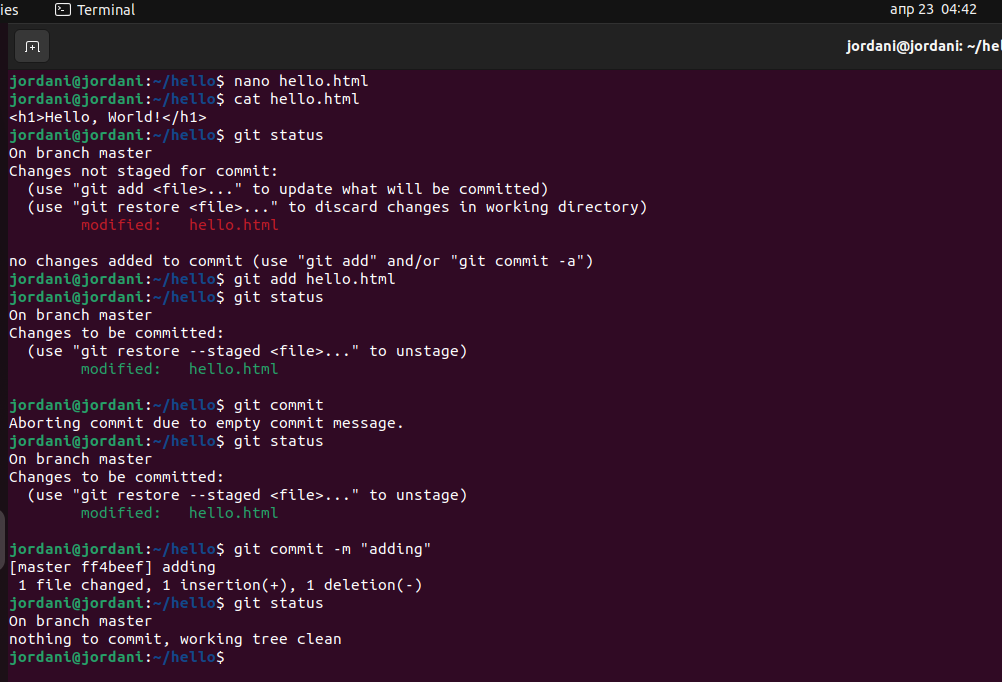
Создание репозитория

## Внесение изменений

Изменим содержимое файла hello.html на:

<h1>Hello, World!</h1>

Проверив состояние рабочего каталога увидим, что git знает, что файл hello.html был изменен, но при этом эти изменения еще не зафиксированы в репозитории. Теперь проиндексируем изменения и снова посмотрим статус, в нём указано, что изменения пока не записаны в резапозиторий. И наконец закоммитим изменения, внеся их в репозиторий и снова посмотрим статус, который теперь показывает, что все изменения внесены в репозиторий (рис. @fig:003).



Внесение изменений в содержимое репозитория

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

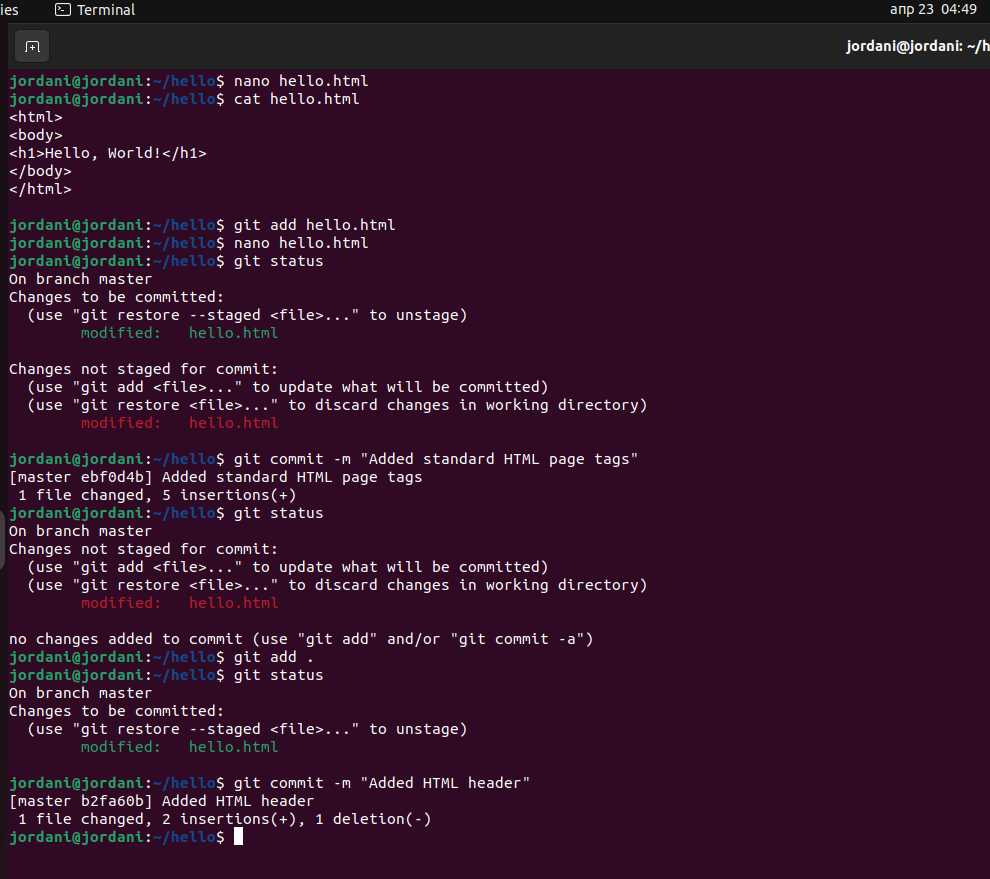
и

.

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

Теперь добавим это изменение в индекс git и добавим заголовки HTML (секцию

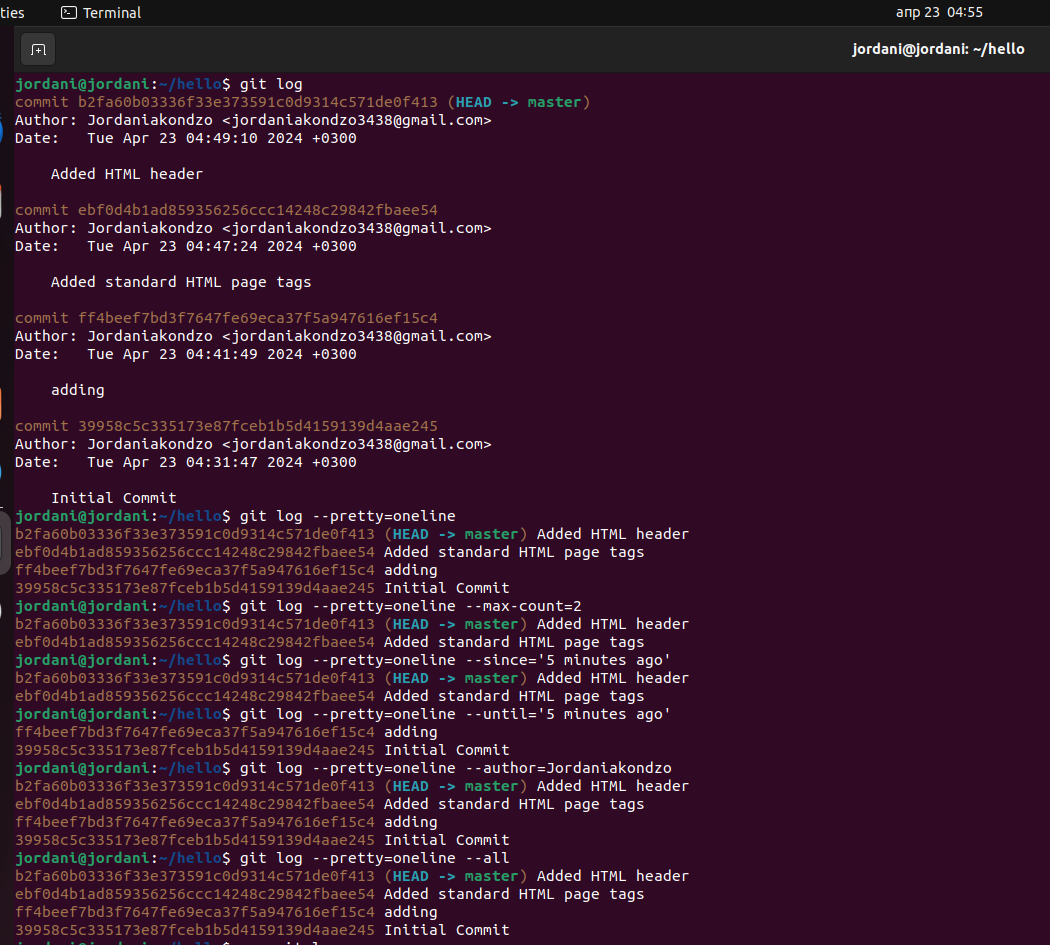
) к странице «Hello, World». Проверив текущий статус увидим, что hello.html указан дважды в состоянии. Первое изменение (добавление стандартных тегов) проиндексировано и готово к коммиту. Второе изменение (добавление заголовков HTML) является непроиндексированным. Произведем коммит проиндексированного изменения, затем проиндексируем оставшееся изменение, посмотрим статус и прокоммитим его(рис. @fig:004).



Внесение нескольких изменений в содержимое репозитория

## История

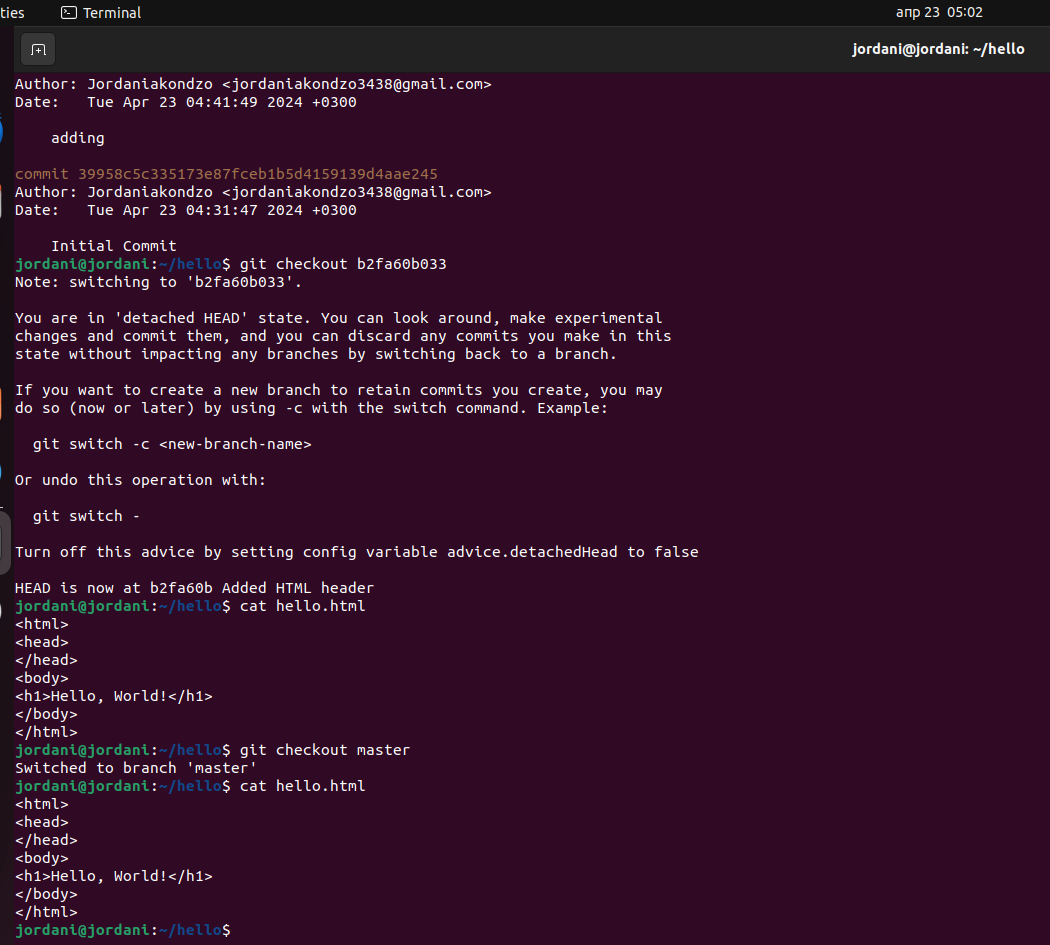
Посмотрим список произведённыз изменений в станлартном виде, затем в однострочном, а также с указанием времени и количества(рис. @fig:005).



Просмотр истории

## Получение старых версий

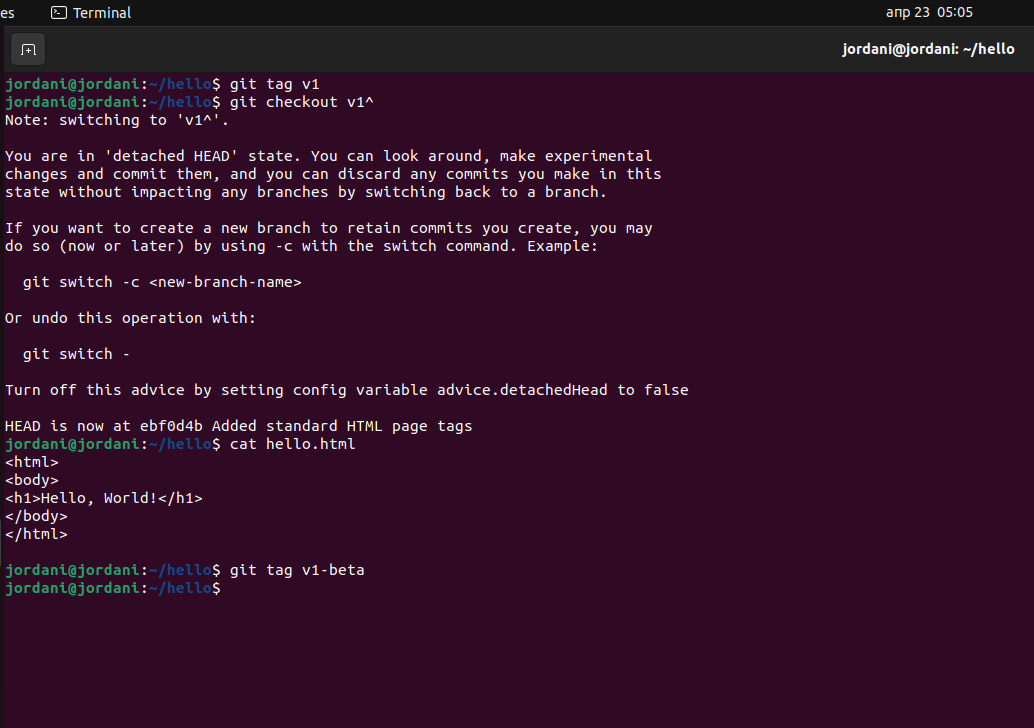
Изучим данные лога и найдем там хэш первого коммита, используя его вернемся к первой верссии и просмотрим файл hello.html, действительно, увидим первую версию. Затем вернемся к последней версии в ветке master и вновь посмотрим на файл(рис. @fig:006).



Просмотр разных версий репозитория

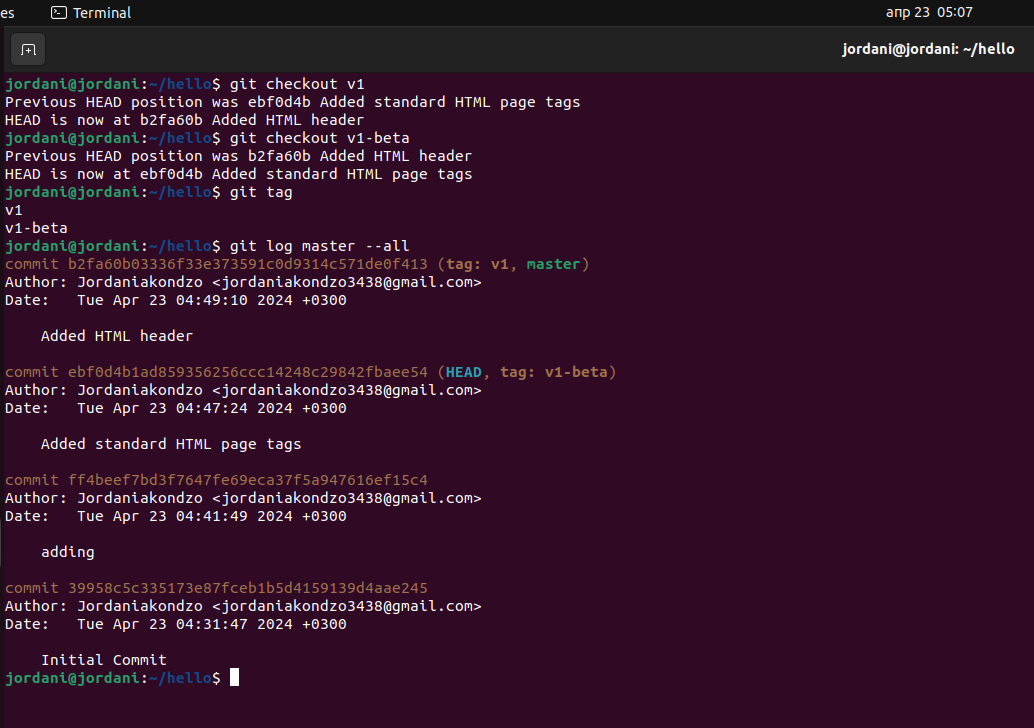
## Создание тегов версий

Назовем текущую версию страницы hello первой (v1). Создадим тег первой версии и используем его для того чтобы венуться к предыдущей, которой также присвоим тег(рис. @fig:007).



Создание тегов версий

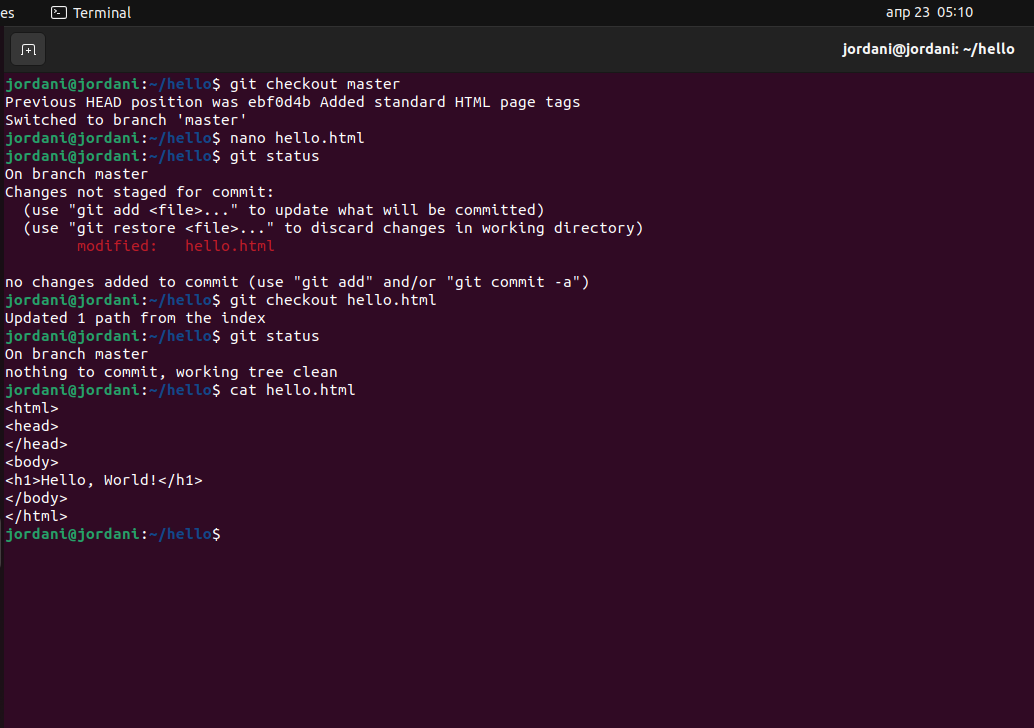
Переключимся по тегам между двумя отмеченными версиями. Просмотрим все доступные теги(их два) и посмотрим теки в логе(рис. @fig:008).



Переключение по имени тега и просмотр доступных тегов

## Отмена локальных изменений (до индексации)

Убдеимся, что мы находимся на последнем коммите ветки master и внесем изменение в файл hello.html в виде нежелательного комментария. Затем проверим статус, увидим, что изменения ещё не проиндексированы. Используем команду git checkout для переключения версии файла hello.html в репозитории(рис. @fig:009).



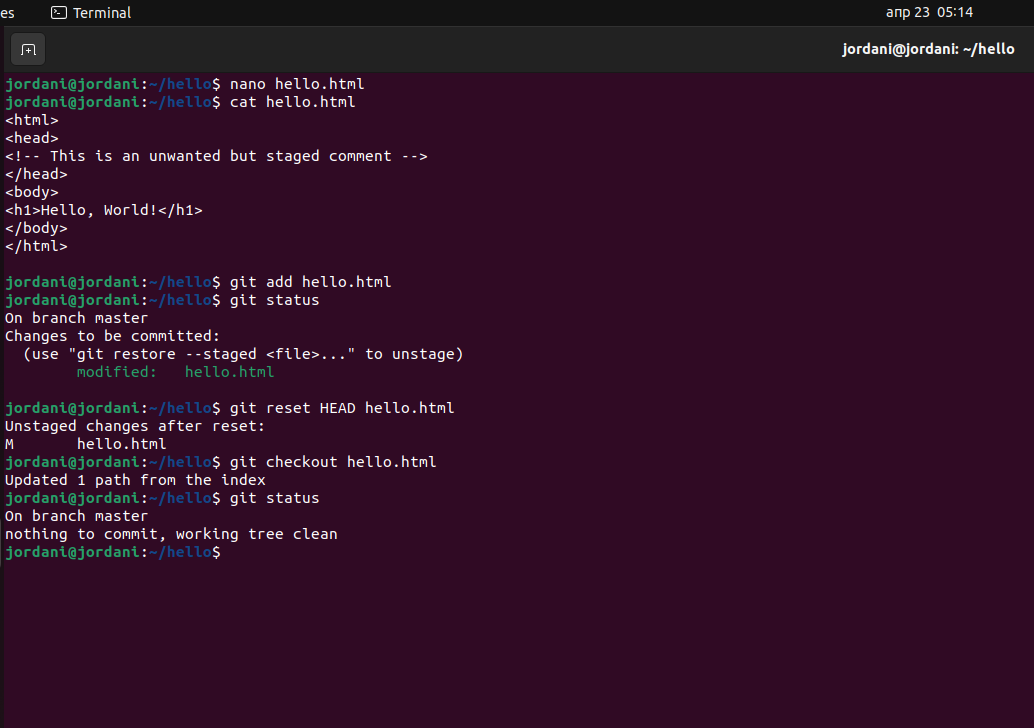
Отмена локальных изменений (до индексации)

## 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 и проверим состояние. Состояние показывает, что изменение было проиндексировано и готово к коммиту. Используем команду git reset, чтобы сбросить буферную зону к HEAD. Это очищает буферную зону от изменений, которые мы только что проиндексировали. И переключимся на последнюю версию коммита, посмотрев статус увидим, что наш каталог опять чист(рис. @fig:010).



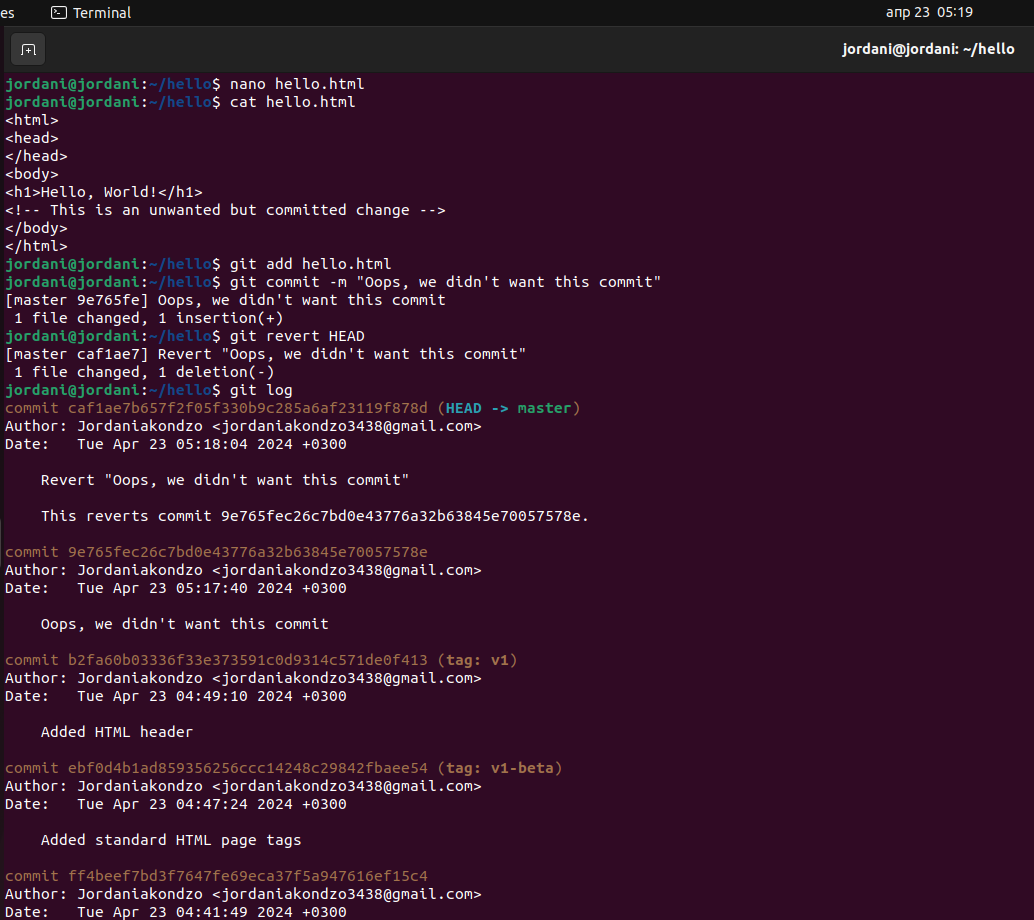
Отмена проиндексированных изменений (перед коммитом)

## Отмена коммитов

Изменим файл hello.html на следующий.

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

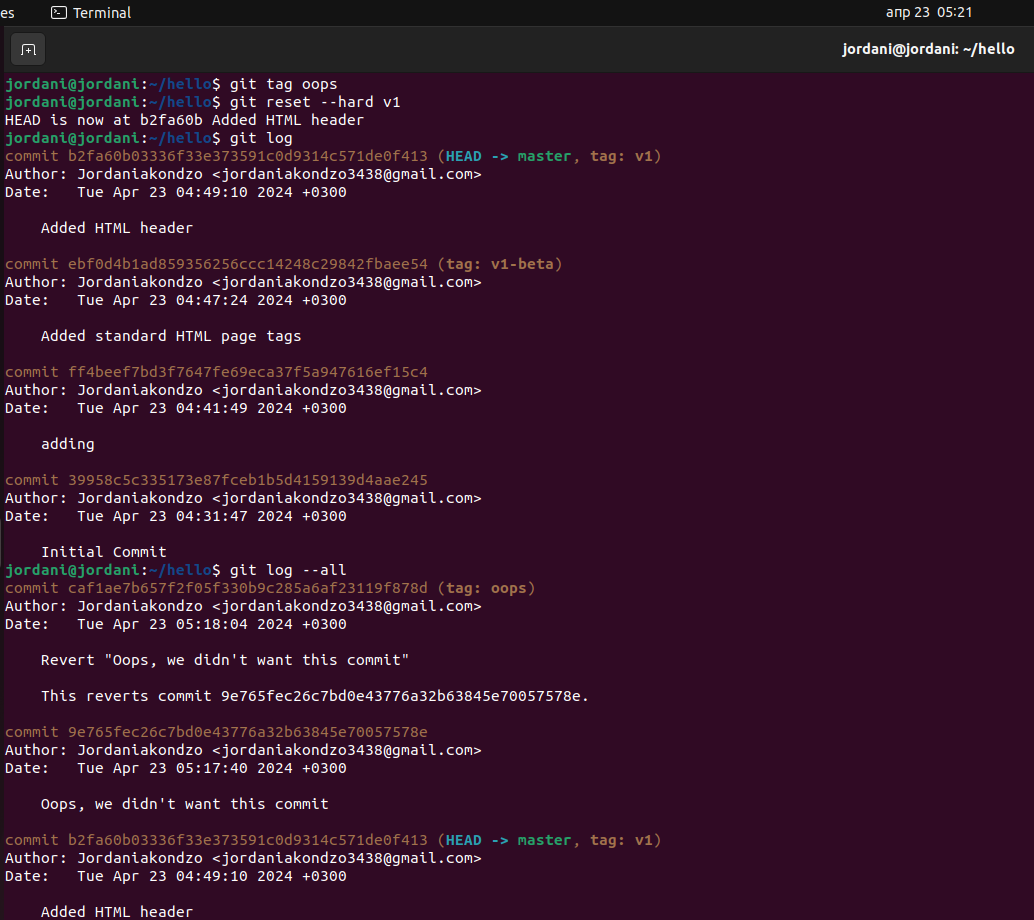
Проиндексируем изменения файла и прокоммитим их. Чтобы отменить коммит, нам необходимо сделать коммит, который удаляет изменения, сохраненные нежелательным коммитом. Перейдем в редактор, где изменим нежелательный коммит. Проверим лог. Проверка лога показывает нежелательные и отмененные коммиты в наш репозиторий(рис. @fig:011).



Отмена коммитов

## Удаление коммиттов из ветки

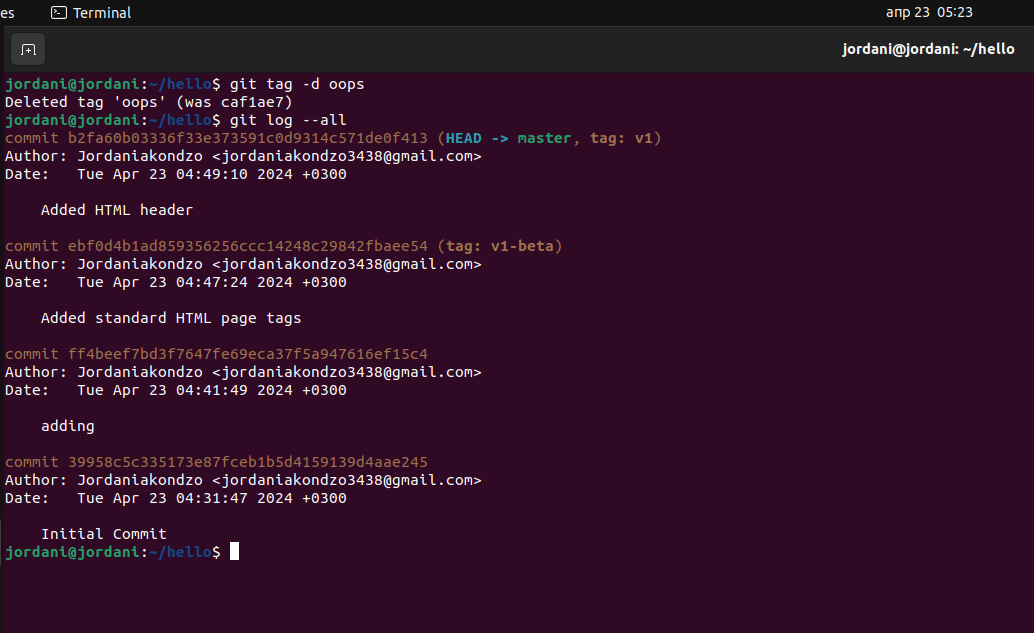
Удалим последние два коммита с помощью сброса, сначала отметим последний коммит тегом, чтобы его можно было потом найти. Используем команду git reset, чтобы вернуться к версии до этих коммитов. Теперь в логе их нет, но если посмотреть логи с опцией –all можно всё ещё их увидеть, но метка HEAD находится на нужной нам версии(рис. @fig:012).



Удаление коммиттов из ветки

## Удаление тега oops

Удалим тег oops и коммиты, на которые он ссылался, сборщиком мусора. Теперь этот тег не отображается в репозитории(рис. @fig:013).



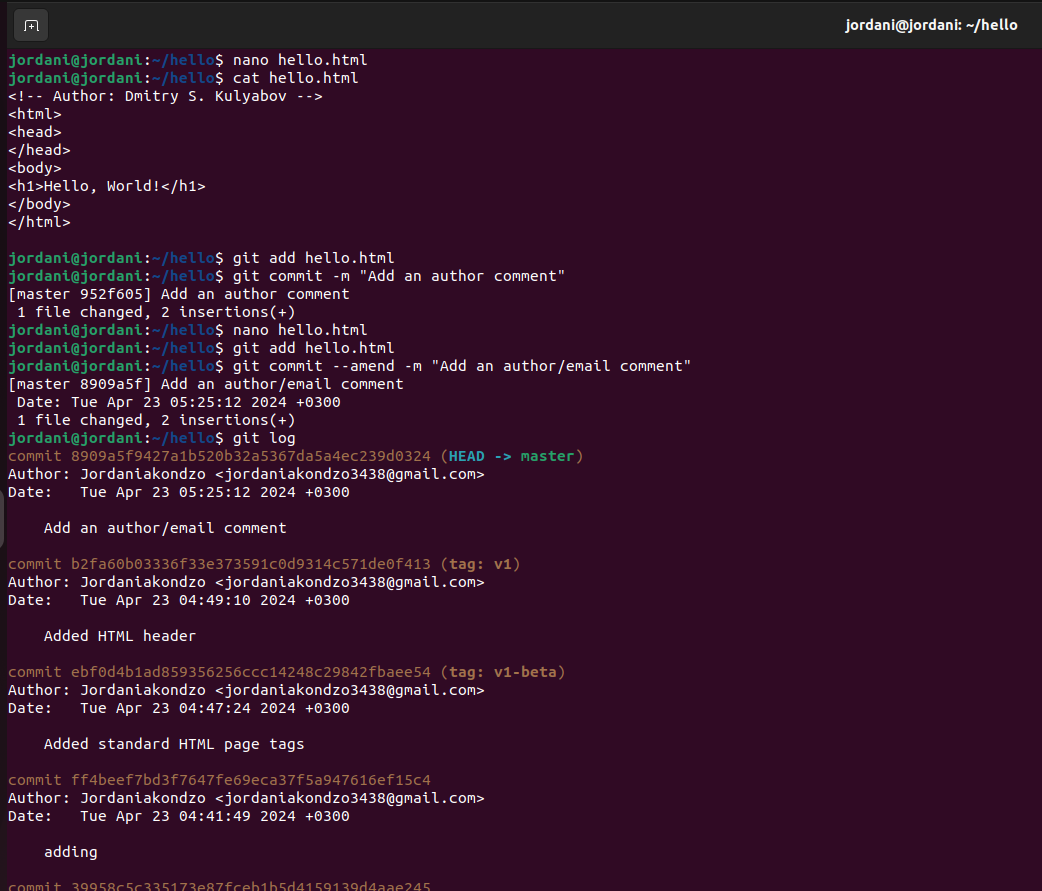
Удаление тега oops

## Изменение предыдущего коммита

Добавим в страницу комментарий автора.

<!-- Author: Dmitry S. Kulyabov -->  
<html>  
 <head>  
 </head>  
 <body>  
 <h1>Hello, World!</h1>  
 </body>  
</html>

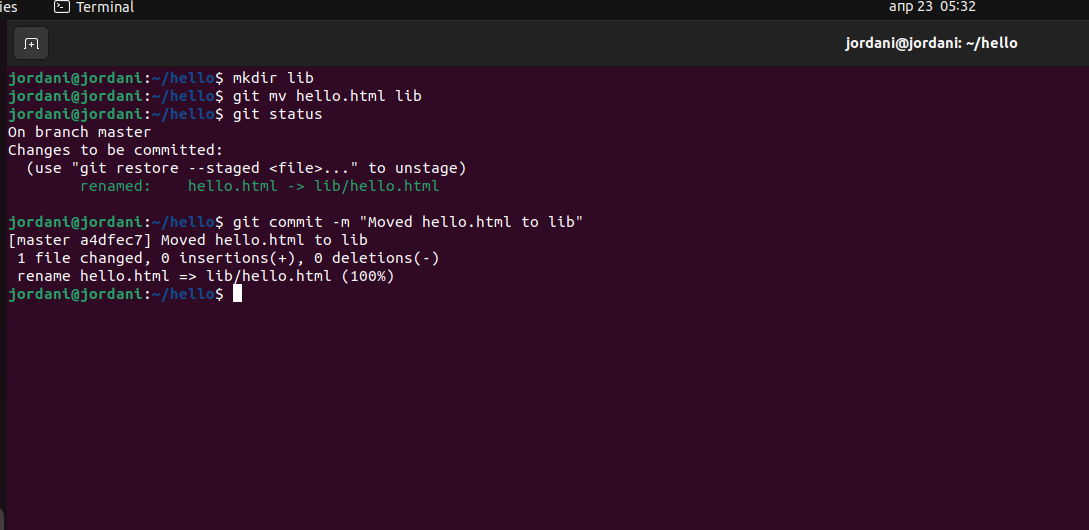
Затем добавим их в репозиторий. Теперь мы хотим добавить в комментарий автора почту, обноим страницу hello, включив в неё почту. Чтобы у нас остался один коммит, а не два, изменим последний с помощьб опции –amend, теперь в логах отображается последняя версия коммита(рис. @fig:014).



Изменение предыдущего коммита

## Перемещение файлов

Переместим наш файл в каталог lib. Для этого создадим его и используем команду git mv, сделаем коммит этого пермещения(рис. @fig:015).



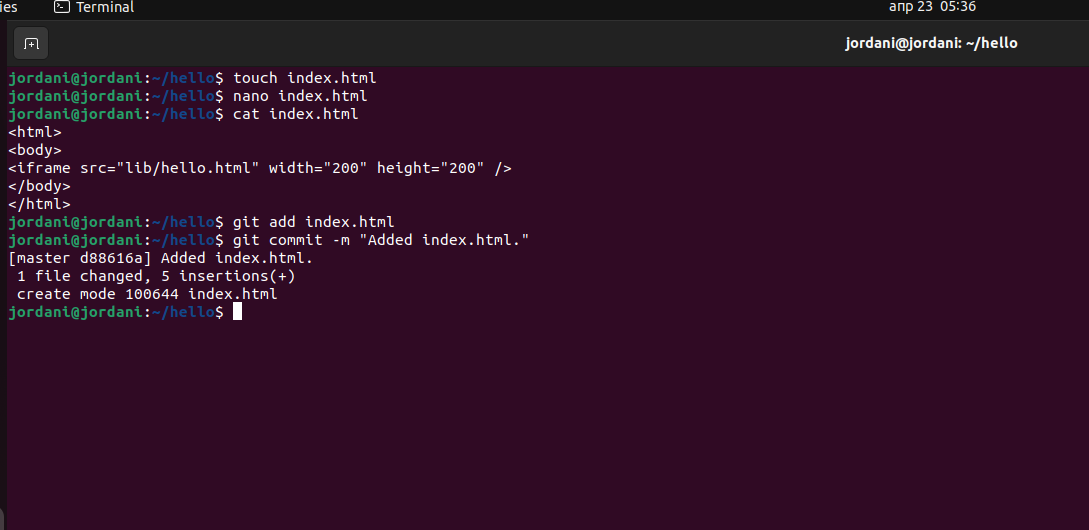
Перемещение файлов

## Подробнее о структуре

Добавим файл index.html в наш репозиторий

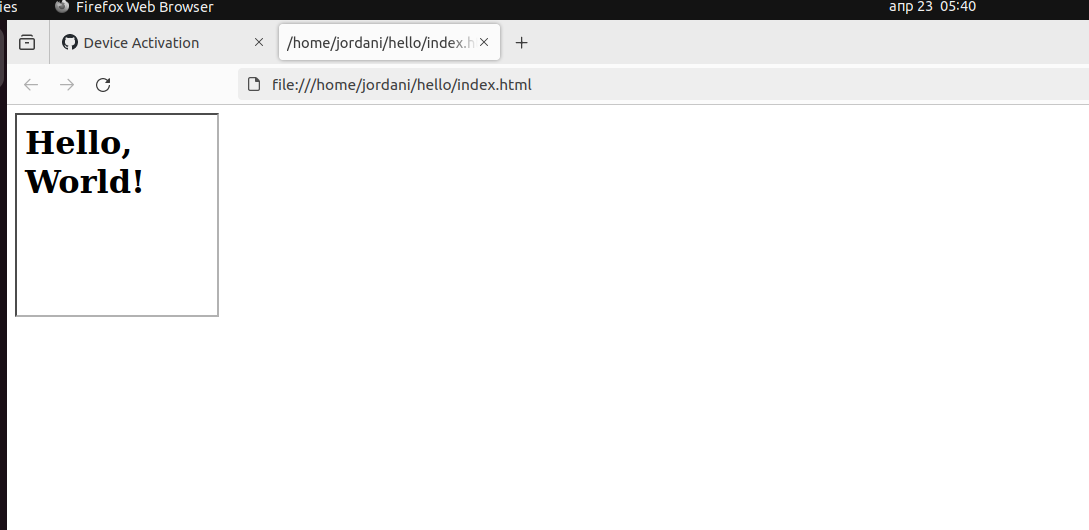
<html>  
 <body>  
 <iframe src="lib/hello.html" width="200" height="200" />  
 </body>  
</html>

Добавим файл и сделаем коммит(рис. @fig:016).



Добавление нового файла в репозиторий

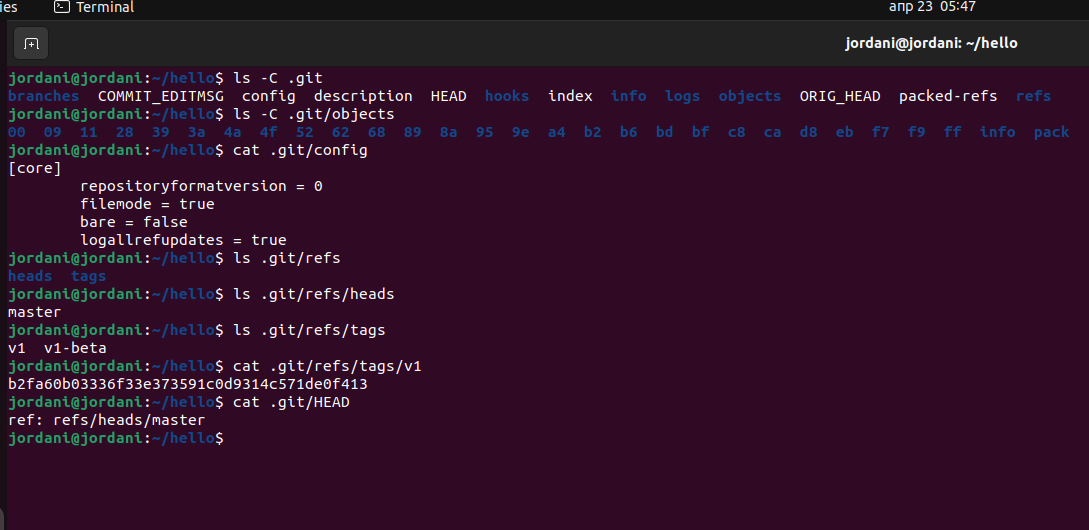
Теперь при открытии index.html, увидим кусок страницы hello в маленьком окошке(рис. @fig:017).



Добавление нового файла в репозиторий

## Git внутри: Каталог .git

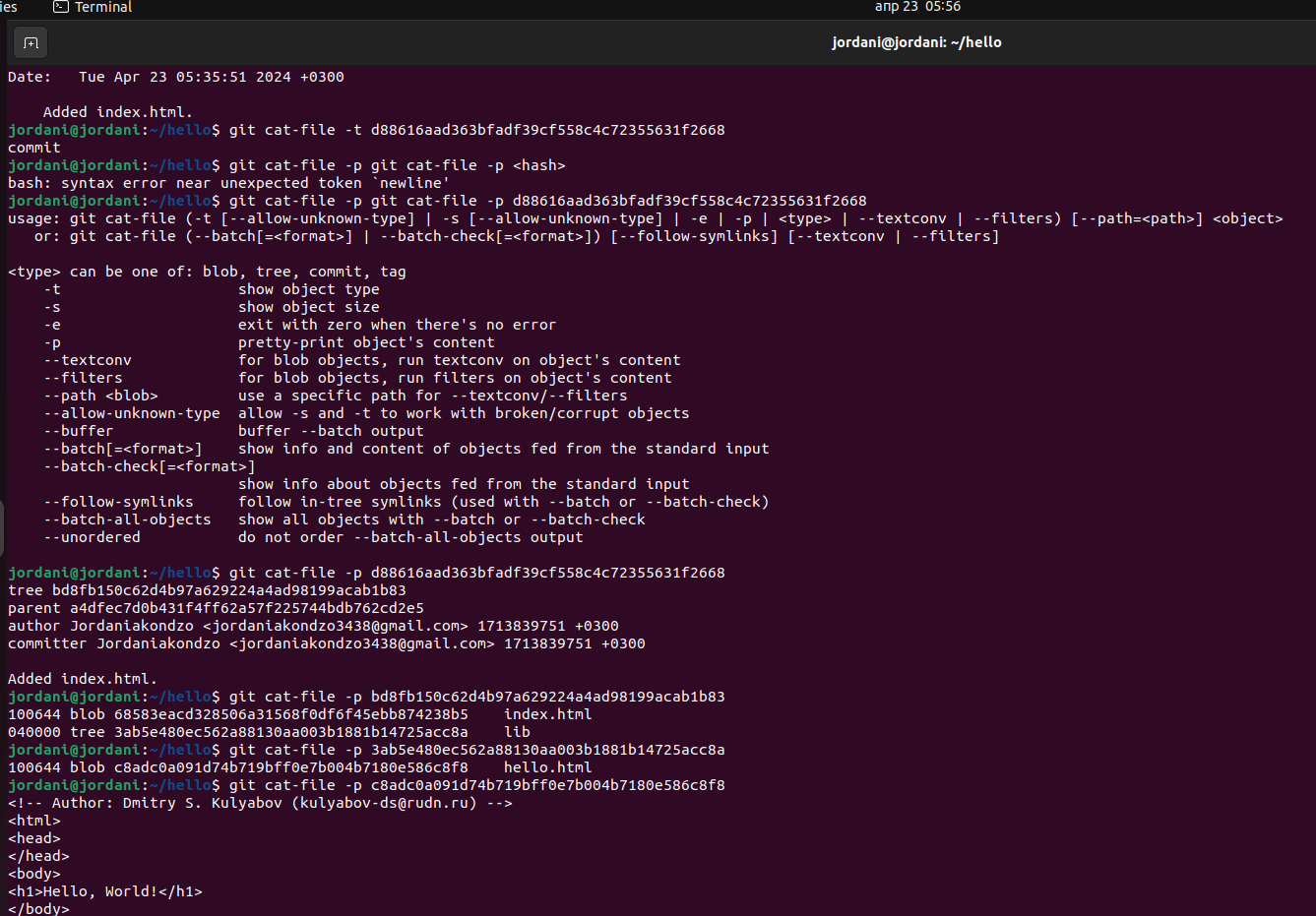
Просмотрим каталог, в котором хранится вся информация git. Затем посмотрим набор каталогов, имена которых состоят из 2 символов. Имена каталогов являются первыми двумя буквами хэша sha1 объекта, хранящегося в git. Посмотрим в один из каталогов с именем из 2 букв. Увидим файлы с именами из 38 символов. Это файлы, содержащие объекты, хранящиеся в git. Посмотрим файл конфигурации, создающийся для каждого конкретного проекта. Затем посмотрим подкаталоги .git/refs/heads и .git/refs/tags, а также содержимое файла v1, в нём хранится хэш коммита, привязанный к тегу. Также посмотрим содержимоей файла HEAD, который содержит ссылку на текущую ветку, в данный момент это ветка master(рис. @fig:018).



Каталог .git

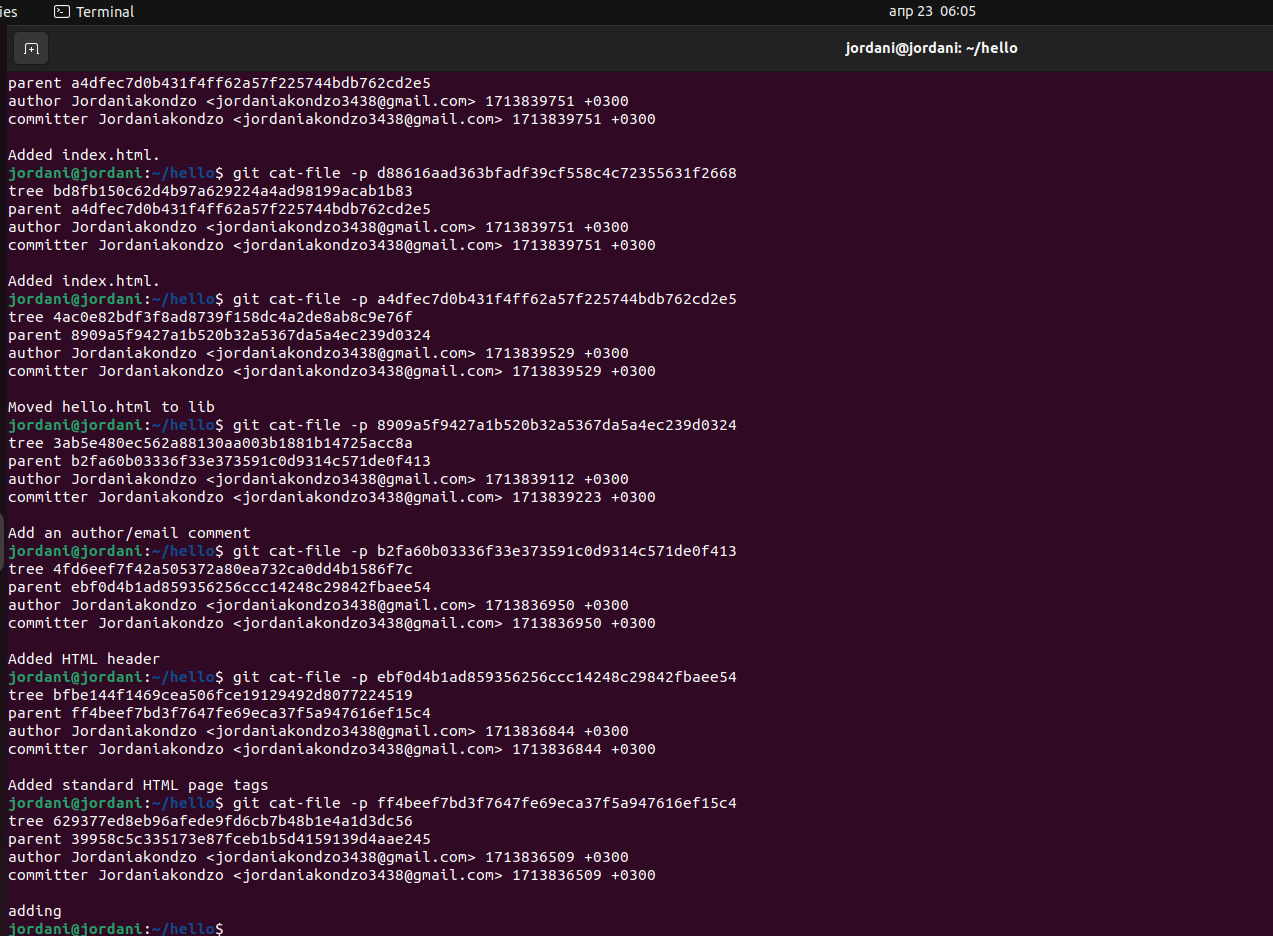
## Работа непосредственно с объектами git

Найдем последний коммит и выедем его с помощью SHA1 хэша. Затем посмотрим дерево каталогов, ссылка на который идёт в последнем коммите, вывдем каталог lib и файл hello.html(рис. @fig:019).



Работа непосредственно с объектами git

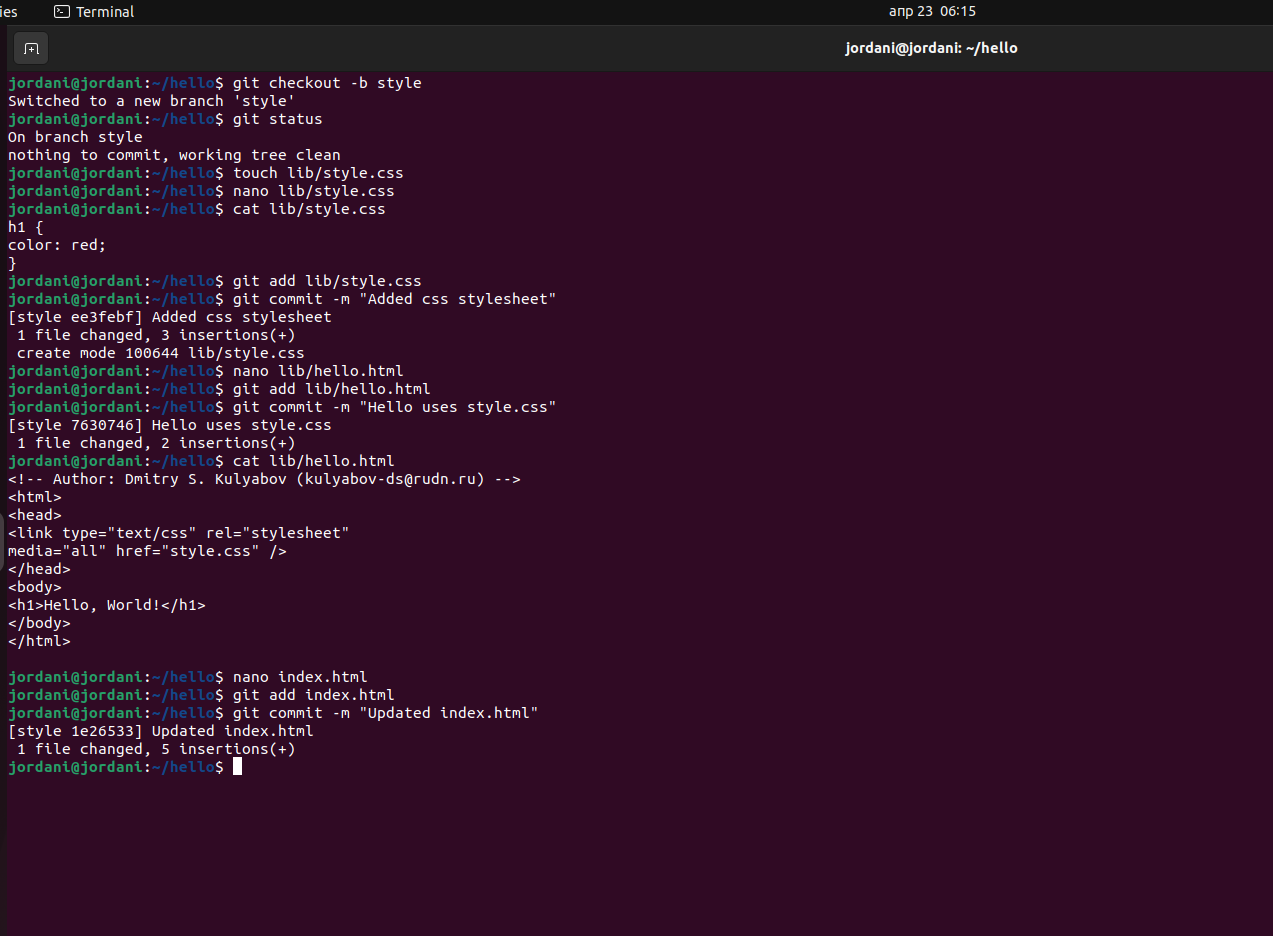
Исследуем git репозиторий вручную самостоятельно. Используя хэш родительского коммита последовательно дойдем до первой версии файла hello.html и посмотрим его(рис. @fig:020).



Поиск оригинального файла hello.html

## Создание ветки

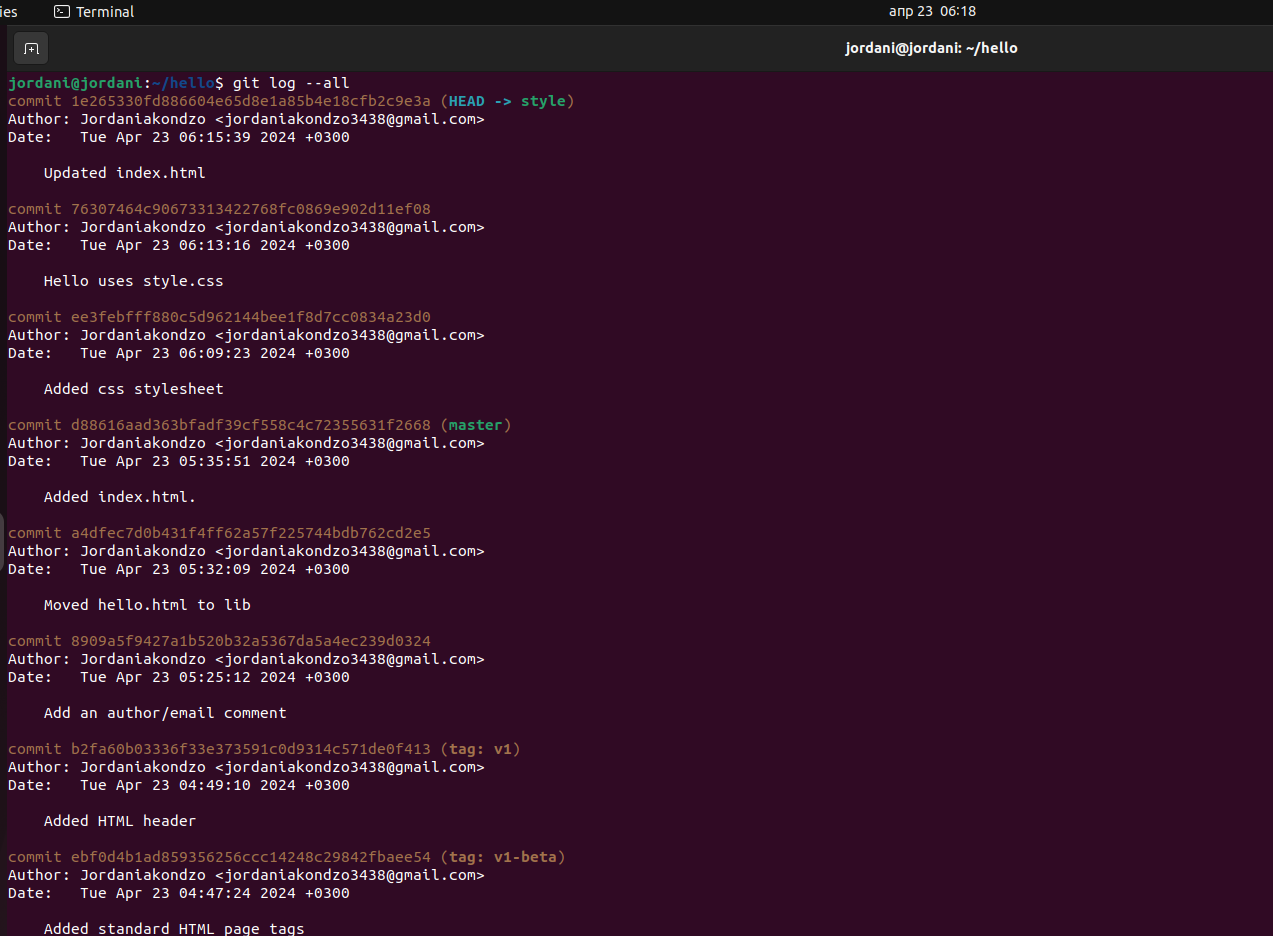
Создадим новую ветку «style» и перейдем в неё. Добавим туда файл стилей style.css и добавим его в репозиторий. Обновим файл hello.html, чтобы использовать стили style.css и index.html, также обавим их в репозиторий(рис. @fig:021).



Создание ветки

## Навигация по веткам

Посмотрим все логи(рис. @fig:022).



Просмотр логов новой ветки

Переключимся обратно на основную ветку и просмотрим содержимое файла ib/hello.html, заметим, что он не использует стили, также просмотрим содержимое этого файла в новой ветке

# Выводы

В результаты выполнение лабораторной работы были приобретены практические навыки работы с ситемой управления версиями Git.

# Список литературы