Лабораторная работа N1

Работа с git

Туем Гислен

Содержание

Список иллюстраций

Список таблиц

# 1 Цель работы

Приобретить практические навыки работы с Git.

# 2 Теоретическое введение

Git-это мощный инструмент, созданный 7 апреля 2005 года для управления ядром lunix.

# 3 Выполнение лабораторной работы

## 3.1 Подготовка

### 3.1.1 Утановка имени и электронной почты

git config --global user.name "Your Name"  
 git config --global user.email "your\_email@whatever.com"

Если git уже установлен, можете переходить к разделу оконча- ния строк.

### 3.1.2 Параметры установки окончаний строк и Установка отображения unicode.(рис. fig21).

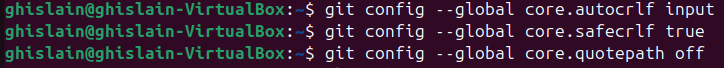


Рис. 1: fig1

Настройка core.autocrlf с параметрами true и input делает все переводы  
строк текстовых файлов в главном репозитории одинаковы.  
core.autocrlf true - git автоматически конвертирует CRLF->LF при комми-  
те и обратно LF->CRLF при выгрузке кода из репозитория на файловую систему  
(используют в Windows). core.autocrlf input - конвертация CRLF в LF только  
при коммитах (используют в Mac/Linux).  
  
 git config --global core.autocrlf input  
 git config --global core.safecrlf true  
 git config --global core.quotepath off

## 3.2 Cоздание проекта

### 3.2.1 Cоздайте страницу «Hello, World» (рис. 2).

mkdir hello  
cd hello  
touch hello.html  
echo "Hello, World!" > hello.html

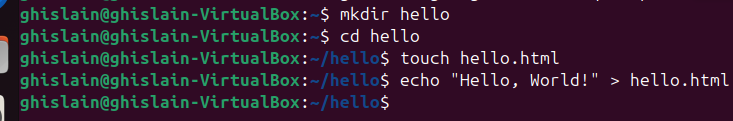


Рис. 2: fig3

### 3.2.2 Cоздание репозитория, добавление файла в репозиторий и Проверка состояние репозитория(рис. 3).

git init  
 git add hello.html  
 git commit -m "Initial Commit"  
 git status

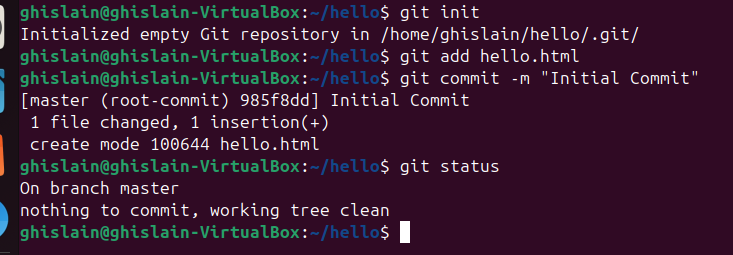


Рис. 3: fig4

## 3.3 Внесение изменений

### 3.3.1 Измените страницу «Hello, World» и Проверьте состояние рабочего каталога.(рис. 4)

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

<h1>Hello, World!</h1>  
 git status

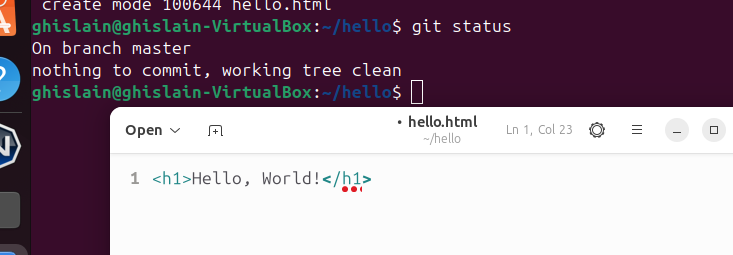


Рис. 4: fig1

## 3.4 Индексация изменений

Теперь выполните команду git, чтобы проиндексировать изменения. Проверьте состояние.(рис. 5).

git add hello.html  
 git status

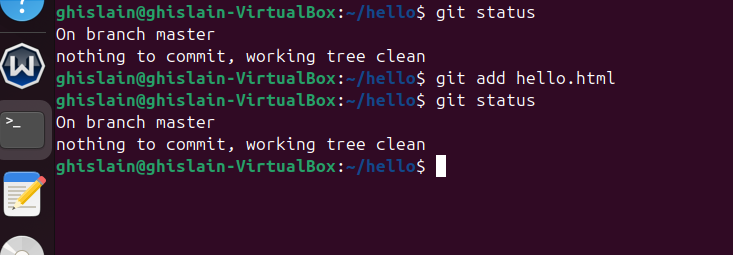


Рис. 5: fig5

### 3.4.1 Коммит изменений

Сделайте коммит и проверьте состояние.(рис. 6).

git commit

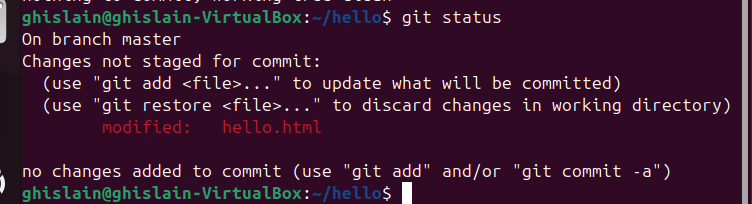


Рис. 6: fig7

В первой строке введите комментарий: «Added h1 tag». Сохраните файл и вый- дите из редактора.(рис. 7).

git status



Рис. 7: fig6

### 3.4.2 Добавьте стандартные теги страницы

Измените страницу «Hello, World», чтобы она содержала стандартные теги.(рис. 8)

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

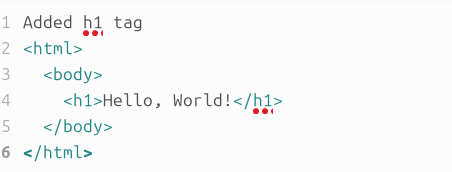


Рис. 8: fig8

Теперь добавьте это изменение в индекс git.

git add hello.html

Теперь добавьте заголовки HTML (секцию

) к странице «Hello, World».(рис. 9)



Рис. 9: fig9

Проверьте текущий статус (рис. 9):

git status

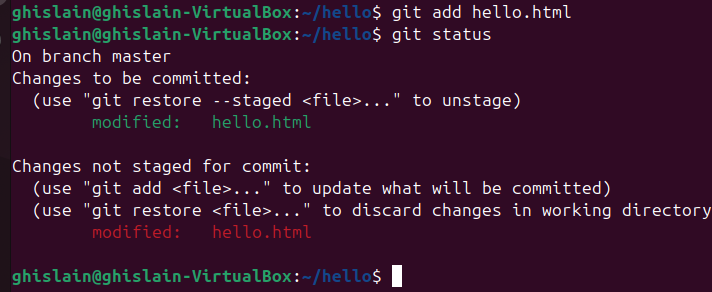


Рис. 10: fig10

Произведите коммит проиндексированного изменения (значение по умолча- нию), а затем еще раз проверьте состояние.(рис. 11)

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

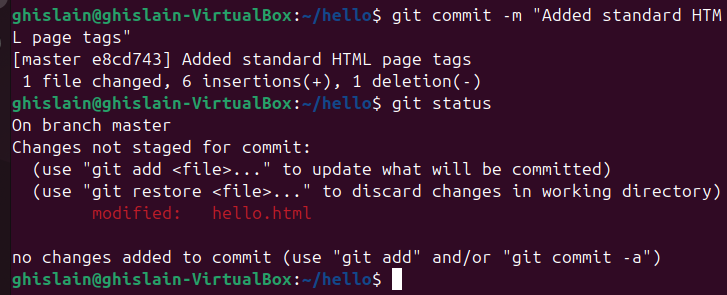


Рис. 11: fig11

Теперь добавьте второе изменение в индекс, а затем проверьте состояние с помощью команды git status.(рис. 12)

git add .  
git status

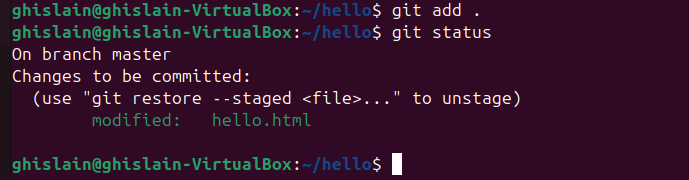


Рис. 12: fig12

Сделайте коммит второго изменения(рис. 13)

git commit -m "Added HTML header"

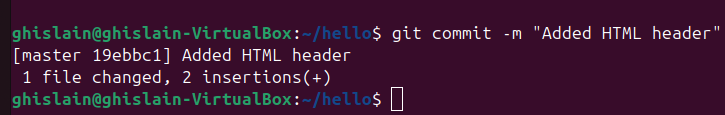


Рис. 13: fig13

### 3.4.3 История

Получим список произведенных изменений(рис. 14):

git log

Есть много вариантов отображения лога.

git log --pretty=oneline --max-count=2  
git log --pretty=oneline --since='5 minutes ago'  
git log --pretty=oneline --until='5 minutes ago'  
git log --pretty=oneline --author=<your name>  
git log --pretty=oneline --all

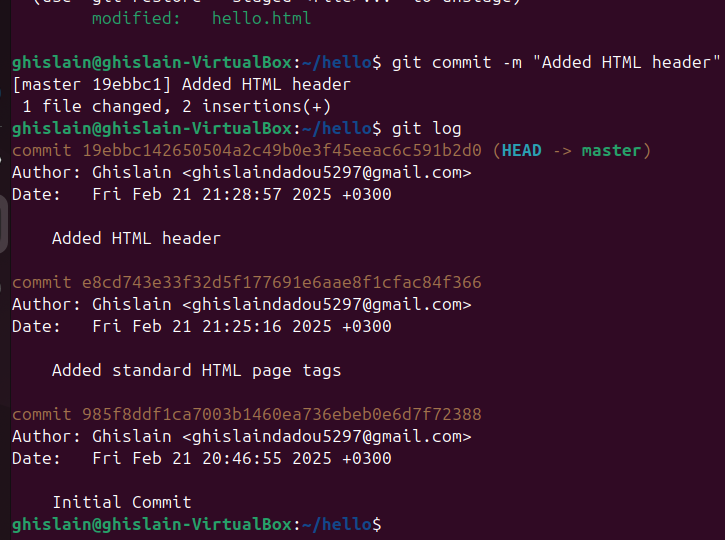


Рис. 14: fig15

Справочная страница(рис. 15):

man git-log



Рис. 15: fig19

### 3.4.4 Получение старых версий(рис. 16)

git log  
git checkout <hash>  
cat hello.htm

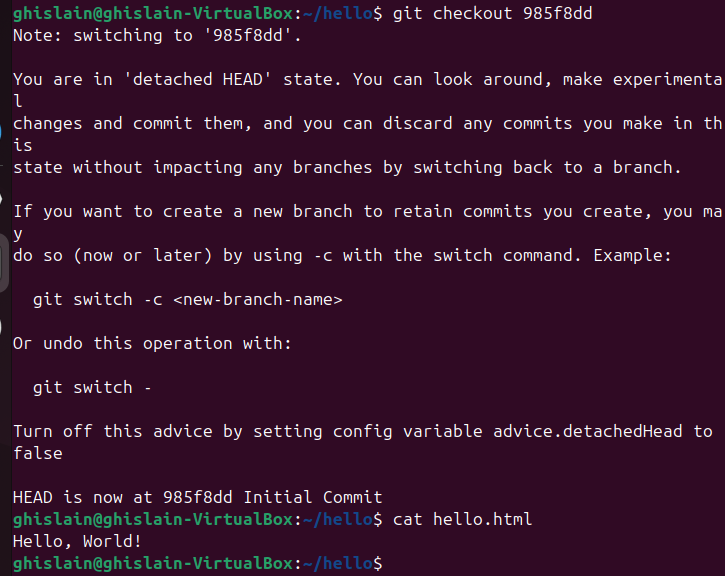


Рис. 16: fig21

Вернитесь к последней версии в ветке master(рис. 17)

git checkout master  
cat hello.html

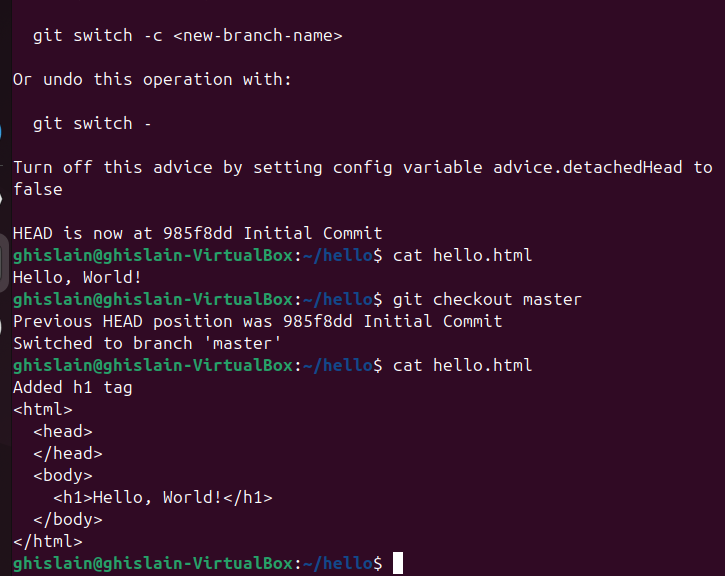


Рис. 17: fig22

### 3.4.5 Создание тегов версий

git tag v1

Теперь текущая версия страницы называется v1. Теги для предыдущих версий Давайте создадим тег для версии, которая идет перед текущей версией и назовем его v1-beta и сделаем ее версией v1-beta.(рис. 18)

git checkout v1^  
cat hello.html  
git tag v1-beta

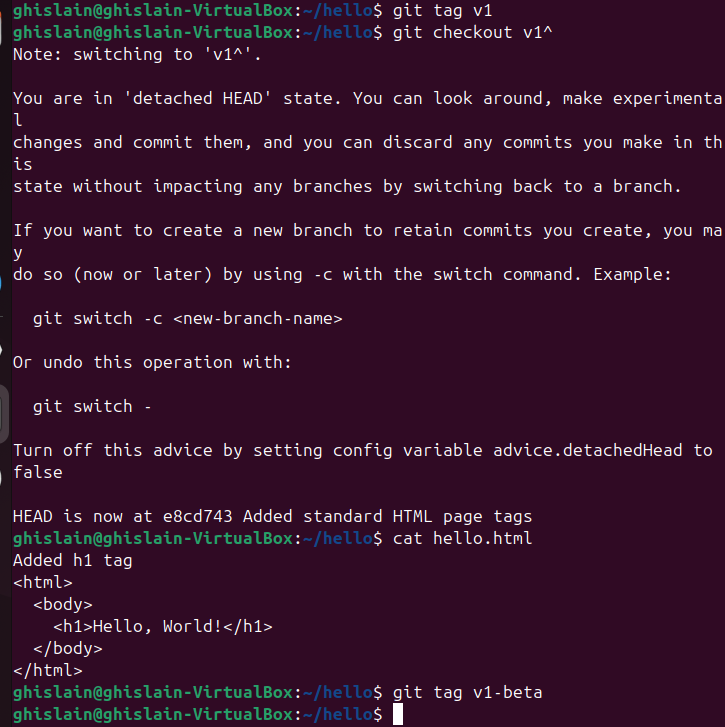


Рис. 18: fig24

### 3.4.6 Переключение по имени тега

Теперь попробуйте попереключаться между двумя отмеченными версиями.(рис. 19)

git checkout v1  
git checkout v1-beta

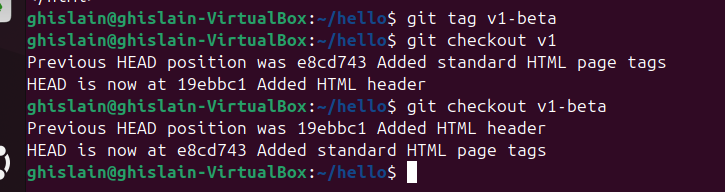


Рис. 19: fig26

### 3.4.7 Просмотр тегов с помощью команды tag

Вы можете увидеть, какие теги доступны, используя команду git tag и также можете посмотреть теги в логе.(рис. 20)

git tag  
git log master --all

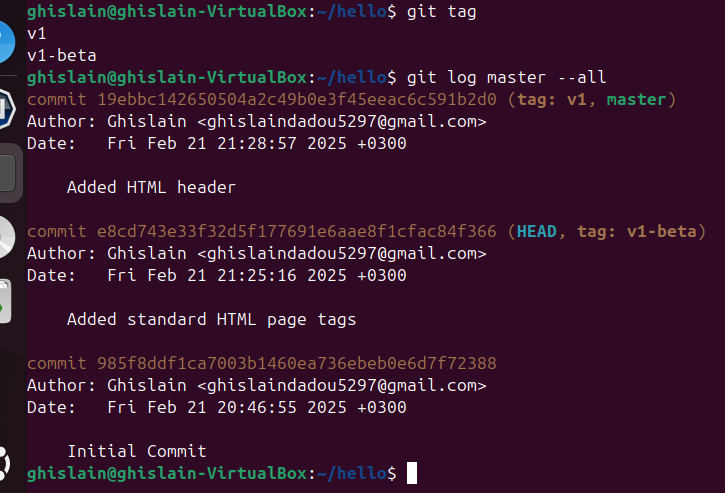


Рис. 20: fig27

## 3.5 Отмена локальных изменений (до индексации)

### 3.5.1 Переключитесь на ветку master

Убедитесь, что вы находитесь на последнем коммите ветки master, прежде чем продолжить работу.(рис. 21)

git checkout master

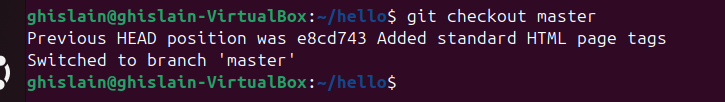


Рис. 21: fig28

### 3.5.2 Измените hello.html

Внесите изменение в файл hello.html в виде нежелательного комментария.(рис. 22) и проверьте состояние.(рис. 23)

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

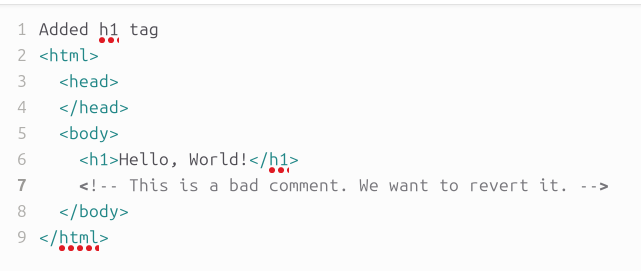


Рис. 22: fig29

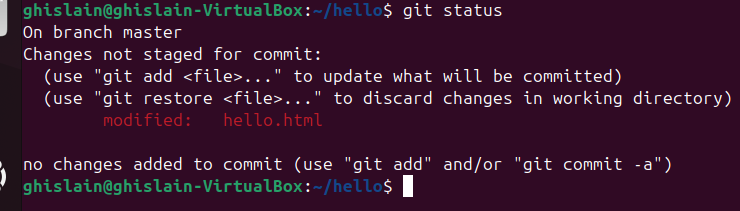


Рис. 23: fig30

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

### 3.5.3 Отмена изменений в рабочем каталоге

Используйте команду git checkout для переключения версии файла hello.html в репозитории.(рис. 24)

git checkout hello.html  
git status  
cat hello.html

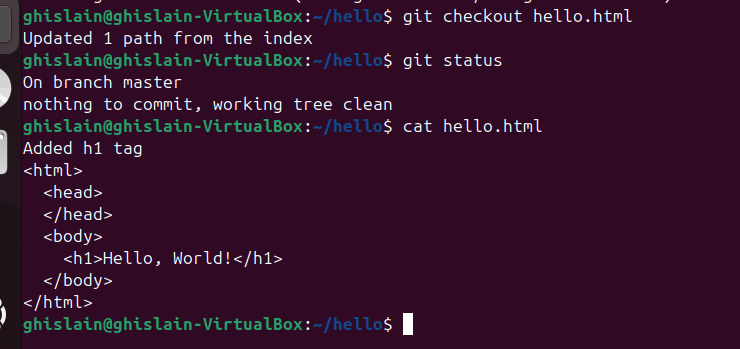


Рис. 24: fig31

## 3.6 Отмена проиндексированных изменений (перед коммитом)

### 3.6.1 Измените файл и проиндексируйте изменения

Внесите изменение в файл hello.html в виде нежелательного комментария(рис. 25).

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

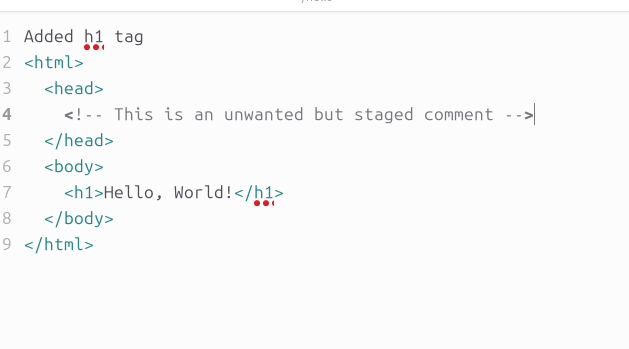


Рис. 25: fig32

Проиндексируйте это изменение и проверьте состояние(рис. 26).

git add hello.html  
git status

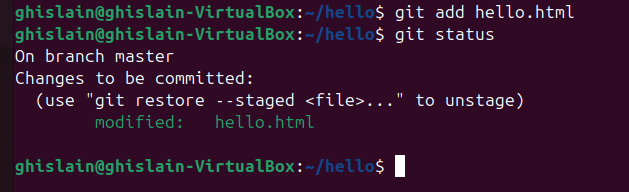


Рис. 26: fig33

### 3.6.2 Выполните сброс буферной зоны

К счастью, вывод состояния показывает нам именно то, что мы должны сделать для отмены индексации изменения.(рис. 27).

git reset HEAD hello.html

Команда git reset сбрасывает буферную зону к HEAD. Это очищает буферную зону от изменений, которые мы только что проиндексировали. Команда git reset (по умолчанию) не изменяет рабочий каталог. Поэтому рабочий каталог все еще содержит нежелательный комментарий. Мы можем ис- пользовать команду git checkout, чтобы удалить нежелательные изменения в рабочем каталоге.

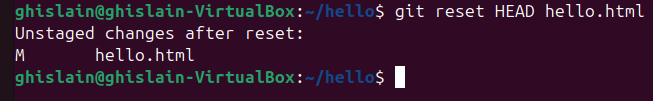


Рис. 27: fig34

### 3.6.3 Переключитесь на версию коммита

git checkout hello.html  
git status

Наш рабочий каталог опять чист(рис. 28).

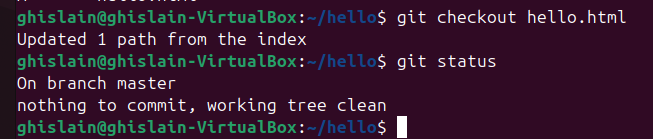


Рис. 28: fig35

## 3.7 Отмена коммитов

### 3.7.1 Отмена коммитов

Иногда вы понимаете, что новые коммиты являются неверными, и хотите их отменить. Есть несколько способов решения этого вопроса, здесь мы будем исполь- зовать самый безопасный. Мы отменим коммит путем создания нового коммита, отменяющего нежела- тельные изменения.

### 3.7.2 Измените файл и сделайте коммит

Измените файл hello.html на следующий(рис. 29).

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



Рис. 29: fig36

Выполните(рис. 30).:

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



Рис. 30: fig37

### 3.7.3 Сделайте коммит с новыми изменениями, отменяющими предыдущие

Чтобы отменить коммит, нам необходимо сделать коммит, который удаляет изме- нения, сохраненные нежелательным коммитом(рис. 31).

git revert HEAD

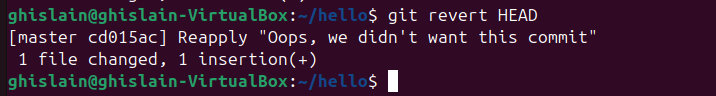


Рис. 31: fig39

### 3.7.4 Проверьте лог

git log

Эта техника будет работать с любым коммитом(рис. 32).

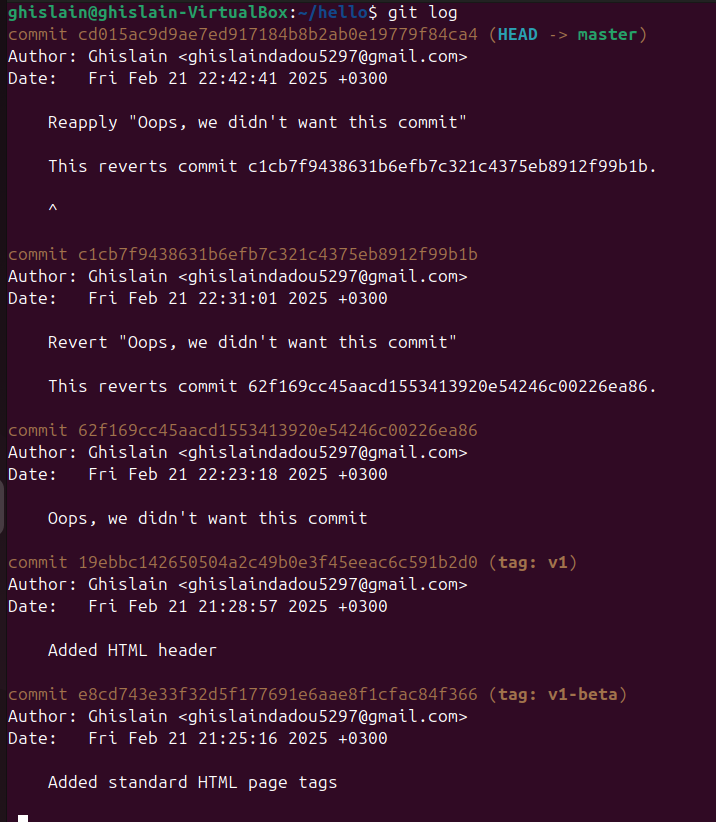


Рис. 32: fig40

## 3.8 Удаление коммиттов из ветки

### 3.8.1 Для начала отметьте эту ветку

Но прежде чем удалить коммиты, давайте отметим последний коммит тегом, чтобы потом можно было его найти(рис. 33).

git tag oops

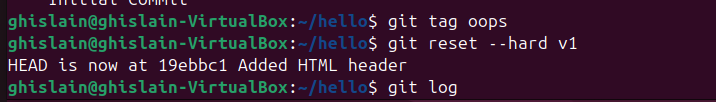


Рис. 33: fig42

### 3.8.2 Сброс коммитов к предшествующим коммиту Oops

git reset --hard v1  
git log

Мы видим, что ошибочные коммиты не исчезли. Они все еще находятся в ре- позитории. Просто они отсутствуют в ветке master. 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(рис. 34).

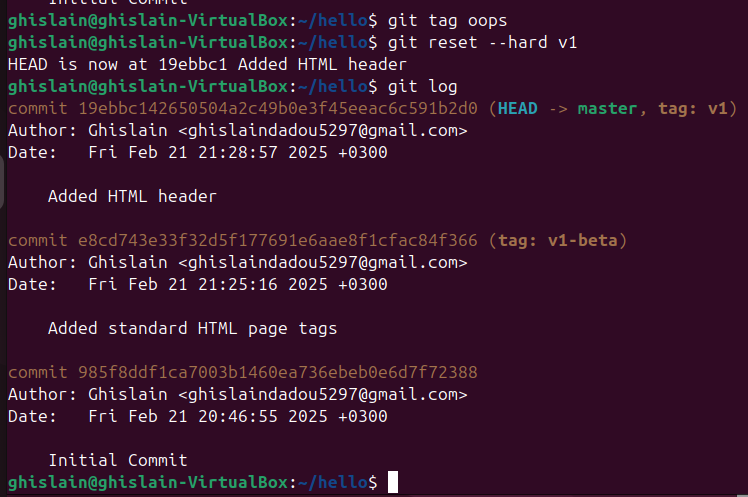


Рис. 34: fig43

## 3.9 Удаление тега oops

### 3.9.1 Удаление тега oops

Тег oops свою функцию выполнил. Давайте удалим его и коммиты, на которые он ссылался, сборщиком мусора(рис. 35).

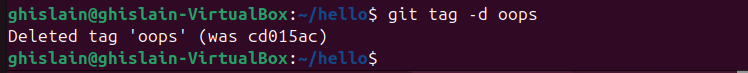


Рис. 35: fig45

## 3.10 Внесение изменений в коммиты

### 3.10.1 Измените страницу, а затем сделайте коммит

(рис. 36).



Рис. 36: fig46

Выполните:

git add hello.html  
git commit -m "Add an author comment"

(рис. 37).

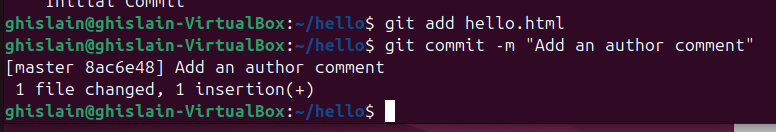


Рис. 37: fig47

### 3.10.2 Необходим email

(рис. 38).

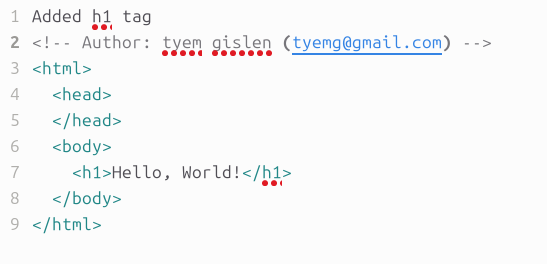


Рис. 38: fig48

### 3.10.3 Измените предыдущий коммит

(рис. 39).

git add hello.html  
git commit --amend -m "Add an author/email comment"

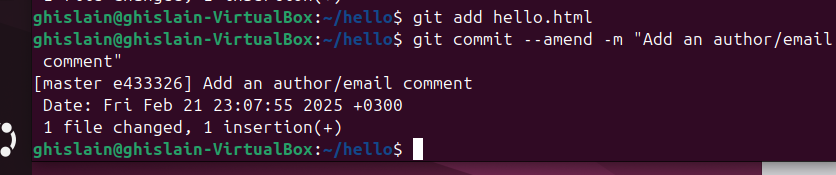


Рис. 39: fig50

## 3.11 Перемещение файлов

### 3.11.1 Переместите файл hello.html в каталог lib

Сейчас мы собираемся создать структуру нашего репозитория. Давайте перенесем страницу в каталог lib.(рис. 40).

mkdir lib  
git mv hello.html lib  
git status

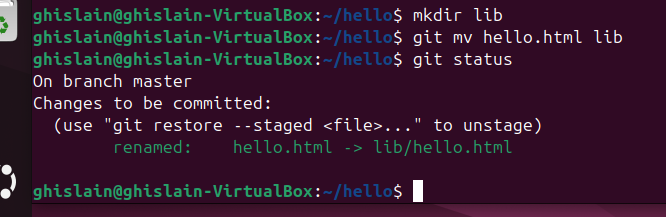


Рис. 40: fig52

### 3.11.2 Второй способ перемещения файлов

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

### 3.11.3 Коммит в новый каталог(рис. 41)

git commit -m "Moved hello.html to lib"

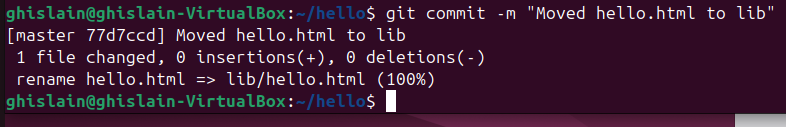


Рис. 41: fig53

## 3.12 Подробнее о структуре

### 3.12.1 Добавление index.html

Добавим файл index.html в наш репозиторий и сделайте коммит(рис. 42)



Рис. 42: fig54

## 3.13 Git внутри: Каталог .git

### 3.13.1 Каталог .git

Выполните: (рис. 43), (рис. 44)

la -C .git  
ls -C .git/objects  
ls -C .git/objects/<dir>

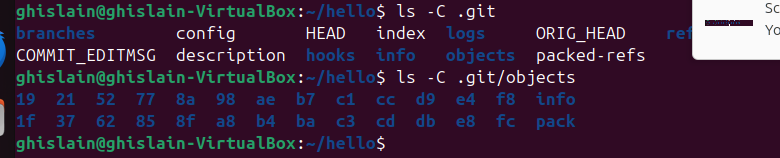


Рис. 43: fig57

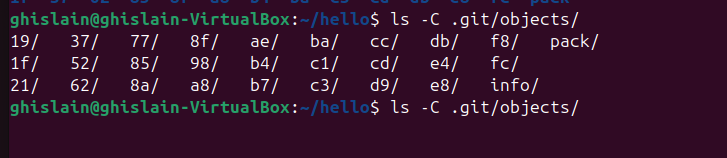


Рис. 44: fig58

### 3.13.2 Config File

Выполните:(рис. 45)

cat .git/config

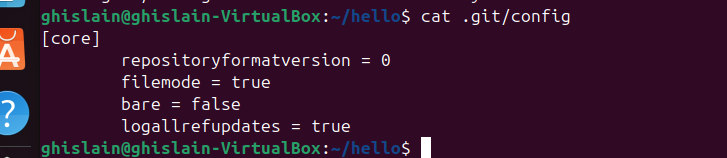


Рис. 45: fig59

### 3.13.3 Ветки и теги, Файл HEAD

Выполните:(рис. 46), (рис. 47)

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

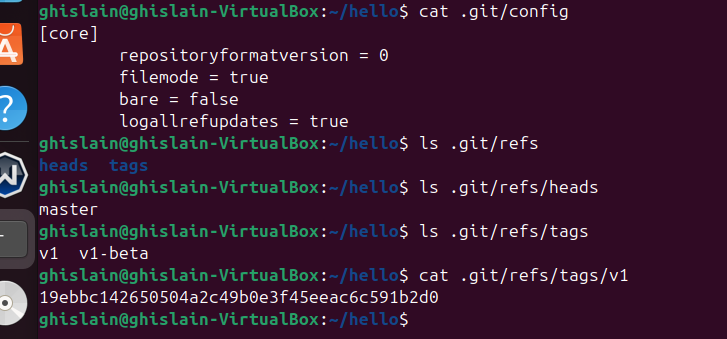


Рис. 46: fig60

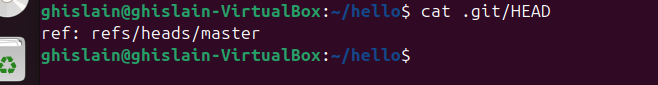


Рис. 47: fig61

## 3.14 Работа непосредственно с объектами git

### 3.14.1 Поиск последнего коммита (рис. 48)

git log --max-count=1

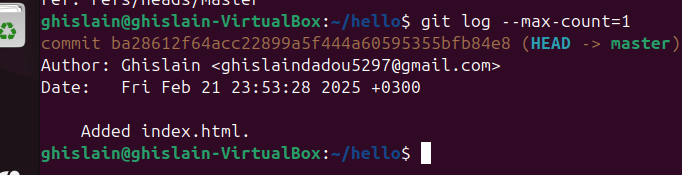


Рис. 48: fig62

### 3.14.2 Вывод последнего коммита с помощью SHA1 хэша

Выполните:(рис. 49)

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

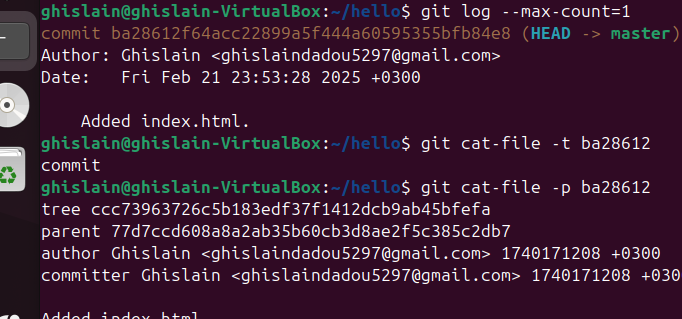


Рис. 49: fig63

### 3.14.3 Поиск дерева

Мы можем вывести дерево каталогов, ссылка на который идет в коммите. Это долж- но быть описание файлов (верхнего уровня) в нашем проекте (для конкретного коммита). Используйте SHA1 хэш из строки «дерева», из списка выше. Выполните:(рис. 50)

git cat-file -p <treehash>

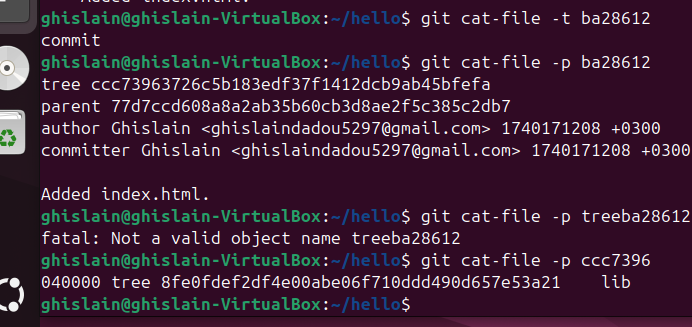


Рис. 50: fig64

### 3.14.4 Вывод каталога lib, Вывод файла hello.html

Выполните:(рис. 51)

git cat-file -p <libhash>  
git cat-file -p <hellohash>



Рис. 51: fig66

## 3.15 Создание ветки, Добавьте файл стилей style.css, Измените основную страницу и Измените index.html

(рис. 52), (рис. 53), (рис. 54), (рис. 55), (рис. 56), (рис. 57), (рис. 58)

git checkout -b style  
git status  
  
touch lib/style.css

Файл lib/style.css:

h1 {  
 color: red;  
}

git add lib/style.css  
git commit -m "Added css stylesheet"  
  
git add index.html  
git commit -m "Updated index.html"

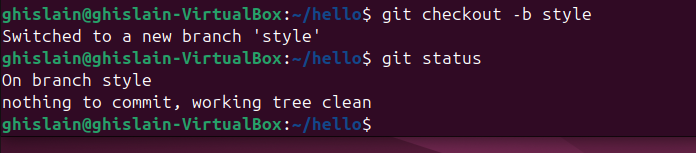


Рис. 52: fig68

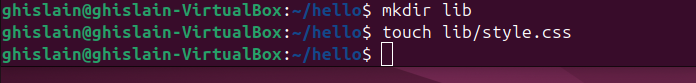


Рис. 53: fig69

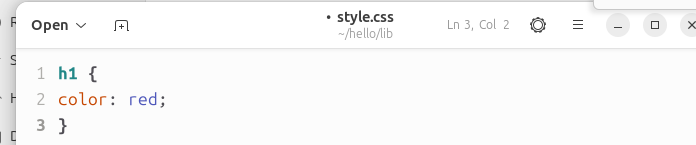


Рис. 54: fig70

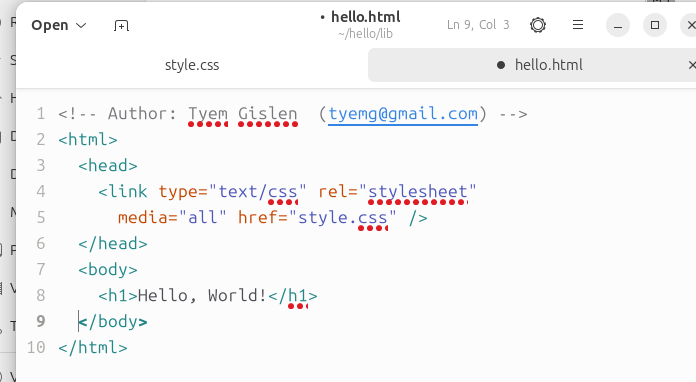


Рис. 55: fig72

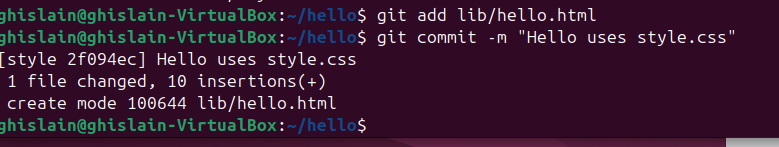


Рис. 56: fig73



Рис. 57: fig74

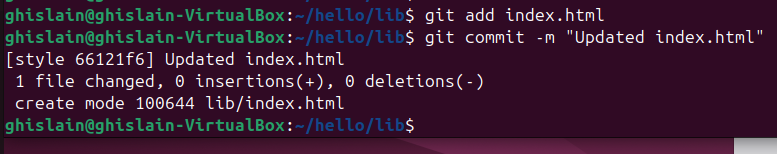


Рис. 58: fig75

## 3.16 Навигация по веткам

Теперь в вашем проекте есть две ветки: Выполните:(рис. 59)

git log --all

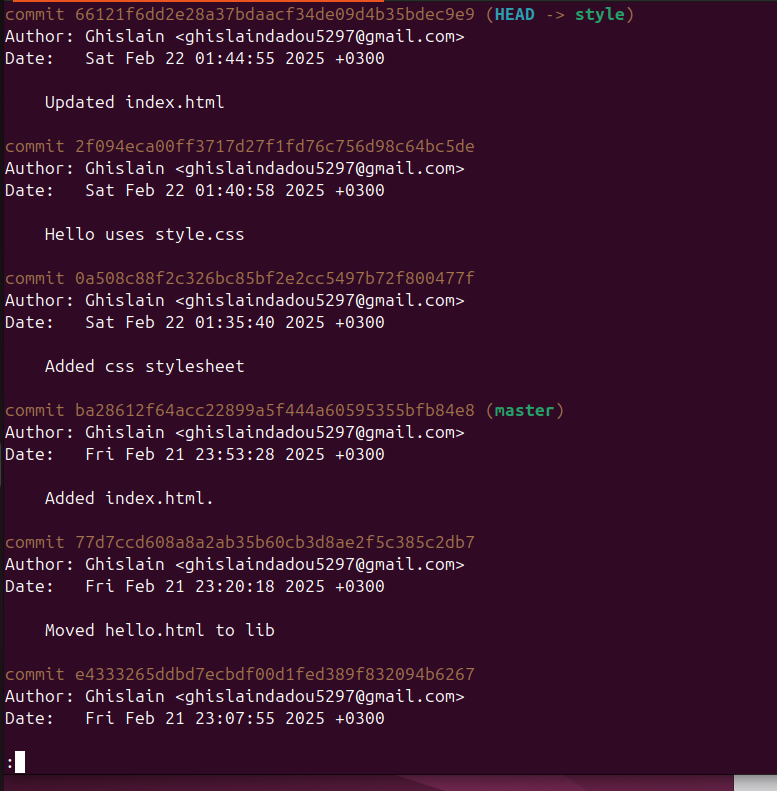


Рис. 59: fig76

### 3.16.1 Переключение на ветку master

Используйте команду git checkout для переключения между ветками:(рис. 60)

git checkout master  
cat lib/hello.html

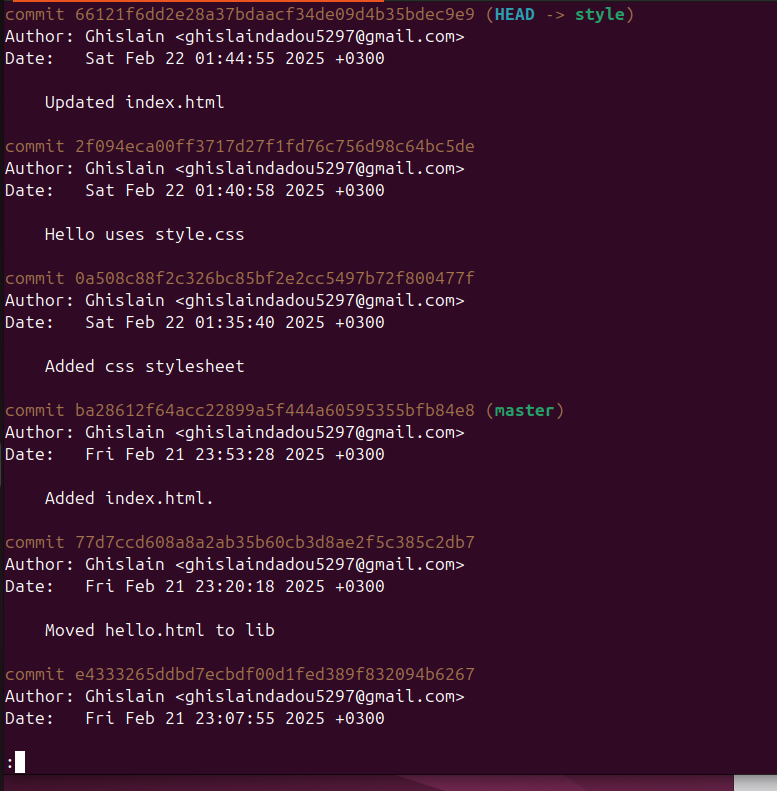


Рис. 60: fig77

### 3.16.2 Вернемся к ветке style

Выполните:(рис. 61)

git checkout style  
cat lib/hello.html

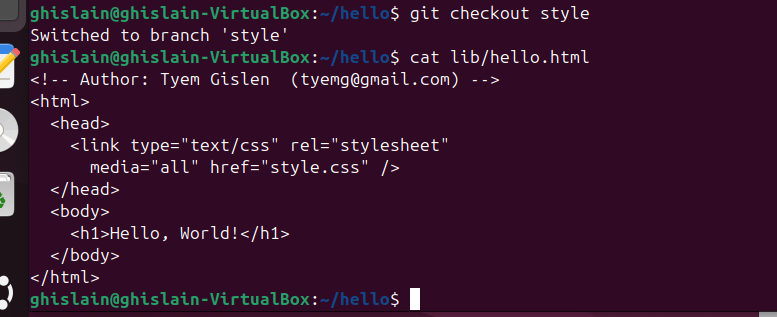


Рис. 61: fig78

## 3.17 Изменения в ветке master

### 3.17.1 Создайте файл README в ветке master(рис. 62)

git checkout master  
  
echo "This is the Hello World example from the git tutorial." > READ

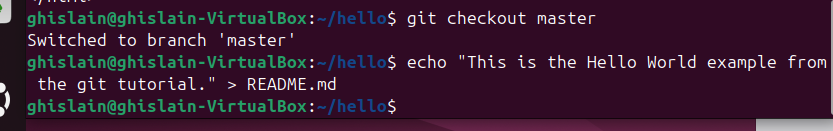


Рис. 62: fig79

### 3.17.2 Сделайте коммит изменений README.md в ветку master.(рис. 63)

git add README.md  
git commit -m "Added README"

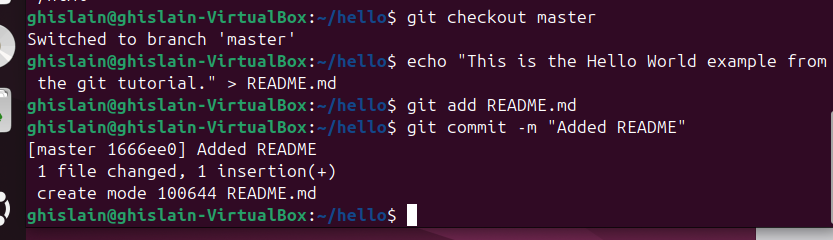


Рис. 63: fig80

## 3.18 Слияние веток

Слияние переносит изменения из двух веток в одну. Давайте вернемся к ветке style и сольем master с style. Выполните:(рис. 64)

git checkout style  
git merge master  
git log --graph --all

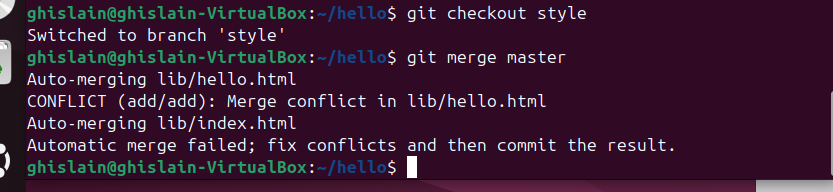


Рис. 64: fig82

## 3.19 Создание конфликта

### 3.19.1 Вернитесь в master и создайте конфликт

Вернитесь в ветку master и внесите следующие изменения:

git checkout master

Файл lib/hello.html(рис. 65)

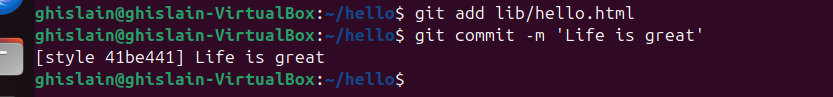


Рис. 65: fig84

git add lib/hello.html  
git commit -m 'Life is great'

## 3.20 Разрешение конфликтов

### 3.20.1 Слияние master с веткой style

Теперь вернемся к ветке style и попытаемся объединить ее с новой веткой master. Выполните:(рис. 66)

git checkout style  
git merge master

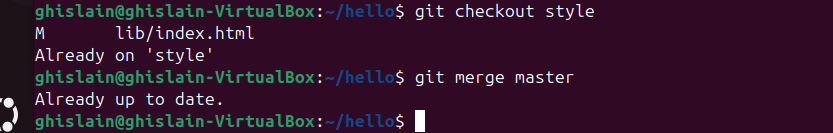


Рис. 66: fig86

Если вы откроете lib/hello.html, вы увидите:

<!-- Author: Dmitry S. Kulyabov (dskulyabov@rudn.ru) -->  
<html>  
 <head>  
<<<<<<< HEAD  
 <link type="text/css" rel="stylesheet" media="all" href="style.css" />  
=======  
 <!-- no style -->  
>>>>>>> master  
 </head>  
 <body>  
 <h1>Hello,World! Life is great!</h1>  
 </body>  
</html>

### 3.20.2 Решение конфликта

Вам необходимо вручную разрешить конфликт. Внесите изменения в lib/hello.html для достижения следующего результата

<!-- Author: Dmitry S. Kulyabov (dskulyabov@rudn.ru) -->  
<html>  
 <head>  
 <link type="text/css" rel="stylesheet"  
 media="all" href="style.css" />  
 </head>  
 <body>  
 <h1>Hello, World! Life is great!</h1>  
 </body>  
</html>

### 3.20.3 Сделайте коммит решения конфликта (рис. 67)

git add lib/hello.html  
git commit -m "Merged master fixed conflict."

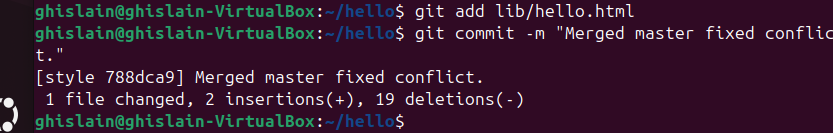


Рис. 67: fig86

## 3.21 Клонирование репозиториев

### 3.21.1 Перейдите в рабочий каталог, Создайте клон репозитория hello и давайте взглянем на клонированный репозиторий.

cd ..  
pwd  
ls  
  
git clone hello cloned\_hello  
ls  
  
cd cloned\_hello  
ls

Вы увидите список всех файлов на верхнем уровне оригинального репозитория README.md, index.html и lib.(рис. 68), (рис. 69),(рис. 70), (рис. 71)

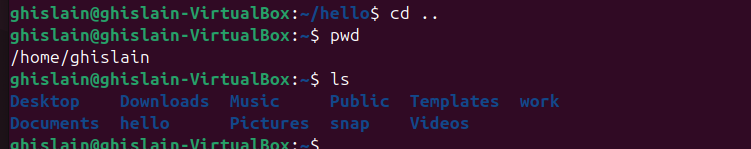


Рис. 68: fig96

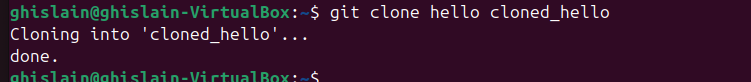


Рис. 69: fig97

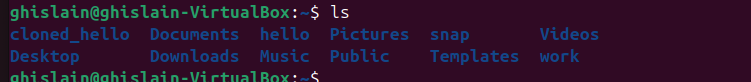


Рис. 70: fig98

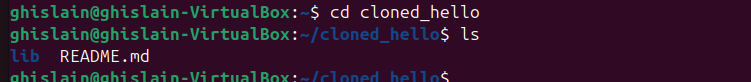


Рис. 71: fig99

## 3.22 Что такое origin?

Выполните:

git remote  
  
git remote show origin

Мы видим, что клонированный репозиторий знает об имени по умолчанию удаленного репозитория(рис. 72)

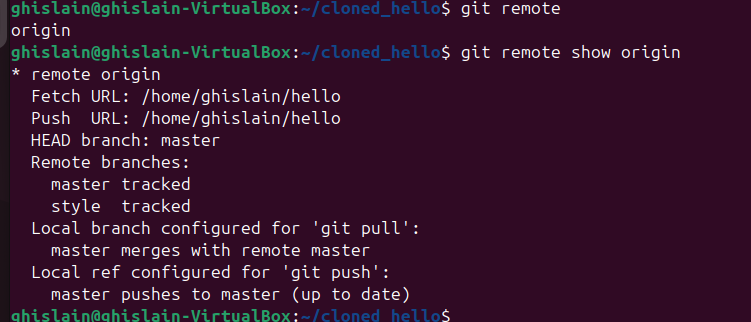


Рис. 72: fig101

## 3.23 Извлечение общих изменений(рис. 73)

cd ../cloned\_hello  
  
git remote add shared ../hello.git  
git branch --track shared master  
git pull shared master  
cat README.md

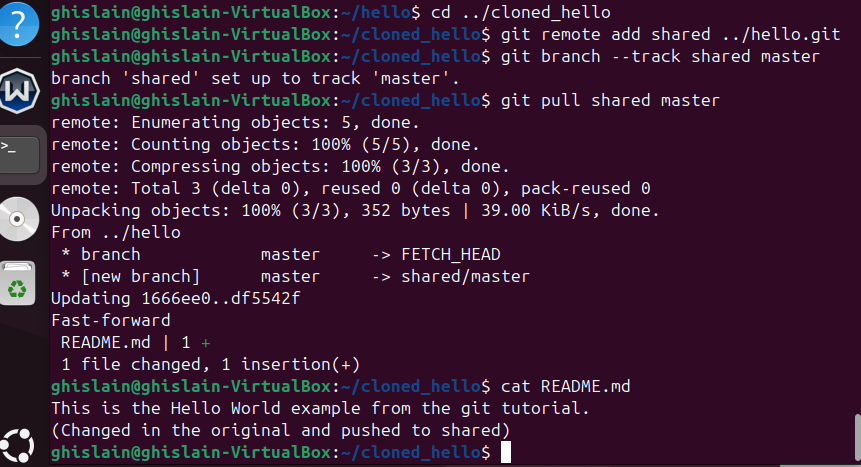


Рис. 73: fig119

# 4 Выводы

В процессе выполнения данной лабораторной работы я приобрела практические навыки работы с Git.