# РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

### Факультет физико-математических и естественных наук

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1 =============== ## Система контроля версий Git

дисциплина: Математическое моделирование

Студент: Койфман Кирилл Дмитриевич

Группа: НПИбд-01-21

## Введение.

### Цель работы.

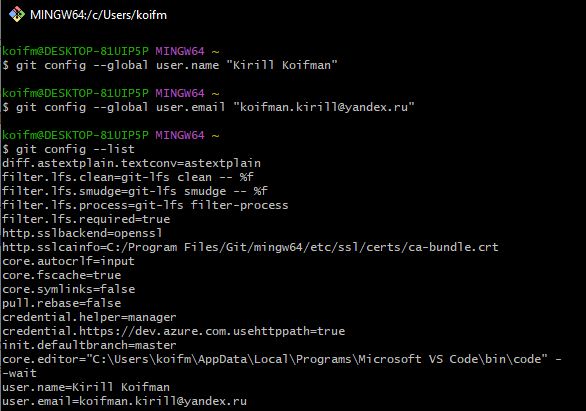
Создать и настроить рабочее пространство для выполнения работы. Получить навыки работы с системой контроля версий Git.

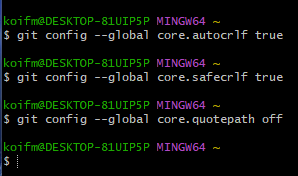
### Задача.

Создать и настроить репозиторий курса с помощью Git и выполнить задания из файла по работе с Git.

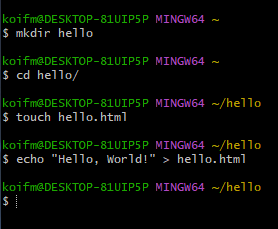
## Ход работы

1. Для начала установим имя, почту, а также настроим core.autocrlfс параметрами true и input, чтобы все переводы строк текстовых файлов в главном репозитории были одинаковы. И добавим отображение Unicode (рис.1 - рис.2):

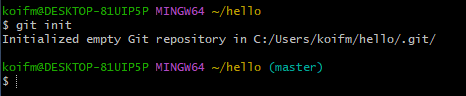
 *РИС.1*

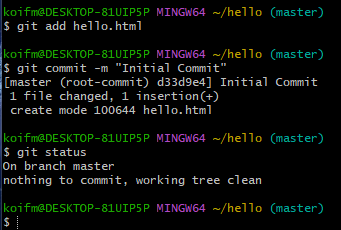
 *РИС.2*

1. Теперь создаим проект “Hello World!”: Для этого создадим пустой каталог с именем hello, затем войдём в него и создадим там файл с именем hello.html (рис.3):

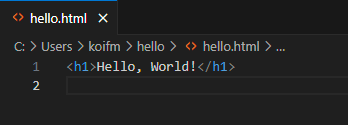
 *РИС.3*

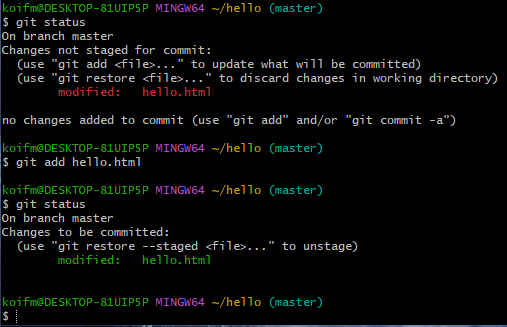
Также создадим git репозиторий из этого каталога, выполнив команду git init (рис.4) и добавим файл в репозиторий (рис.5):

 *РИС.4*

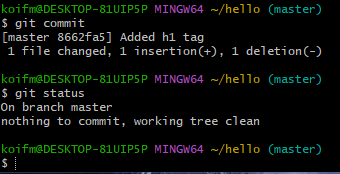
 *РИС.5(коммитить нечего)*

1. Добавим кое-какие HTML-теги к нашему приветствию (рис.6) и проверим статус:
2. Теперь проиндексируем изменения (рис.7)

 *РИС.6*

 *РИС.7(файл был изменён и проиндексирован)*

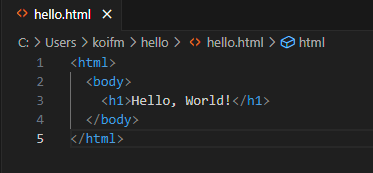
Создадим коммит (рис.8):

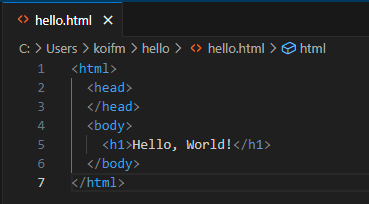
 *РИС.8(коммит был успешно сделан)*

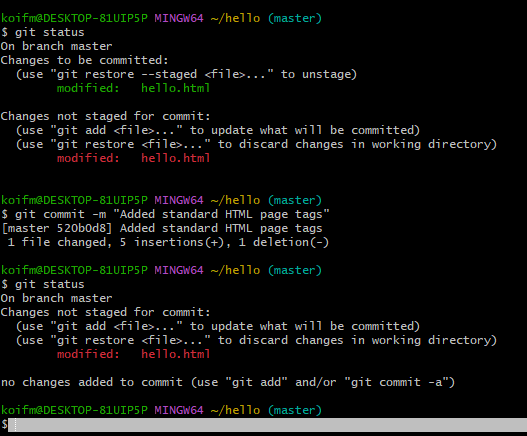
Изменим страницу «Hello, World», чтобы она содержала стандартные теги

и

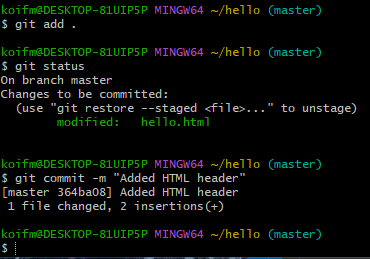
, и добавим изменения в индекс, после чего ещё раз изменим файл и создадим коммит (рис.9 - рис.11):

 *РИС.9*

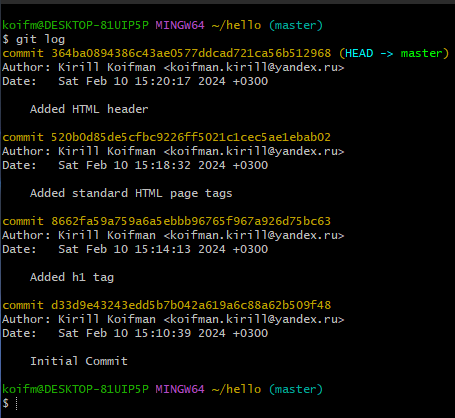
 *РИС.10*

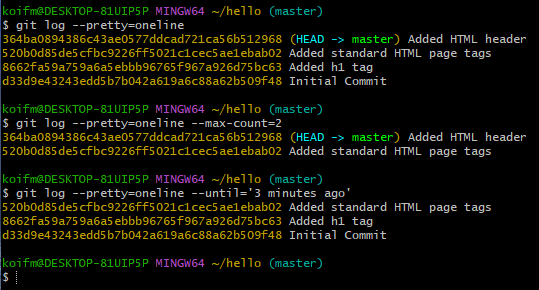
 *РИС.11(hello.html имеет незафиксированные изменения, но уже не в буферной зоне)*

Теперь добавим второе изменение в индекс, а затем проверим состояние с помощью командыgit status (рис.12):

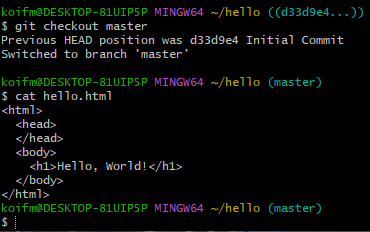
 *РИС.12(изменения были успешно закоммичены)*

Проверим список произведенных изменений в разных форматах (рис.13, рис.14):

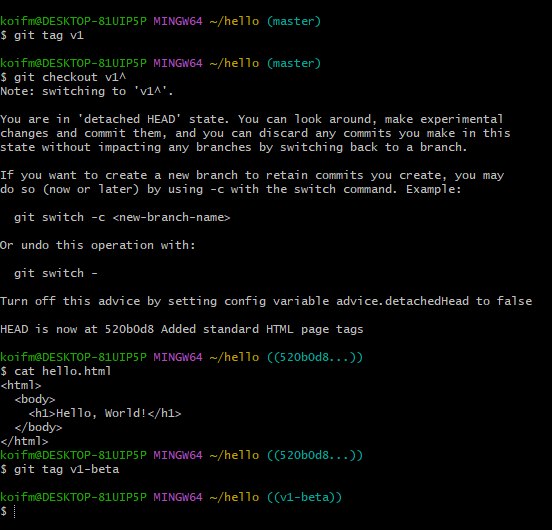
 *РИС.13(спиосок ранее сделанных коммитов с прилагающейся информацией)*

 *РИС.14(спиосок ранее сделанных коммитов с прилагающейся информацией: - в однострочном формате; - в однострочном формате, последние 2; - в однострочном формате, сделанные за последние 3 минуты)*

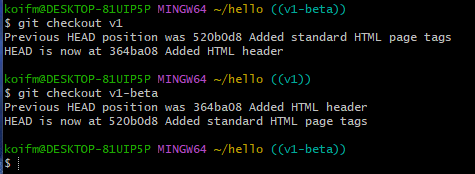
Вернёмся к предыдущей версии с помощью git checkout (рис.15):

 *РИС.16(успешно откатились до последнего коммита и вывели последний вариант файла hello.html)*

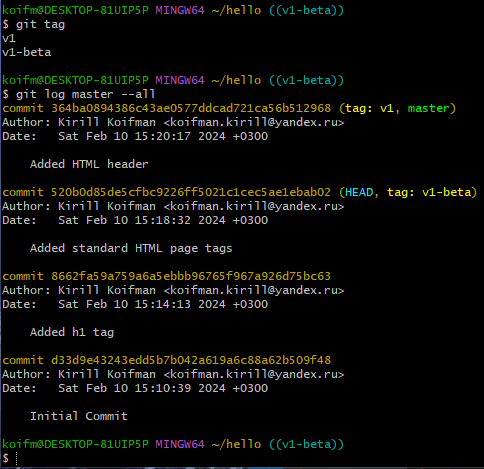
Назовём текущую версию страницы hello первой (v1) и поднимемся к предущему коммиту и ему тоже присвоем тег (рис.17):

 *РИС.17*

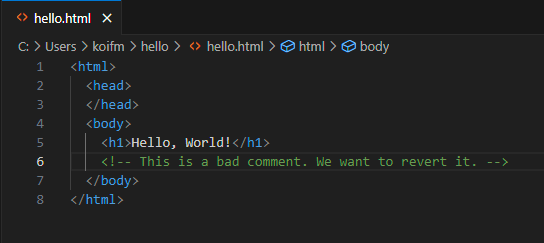
Теперь попробуем попереключаться между двумя отмеченными версиями (рис.18):

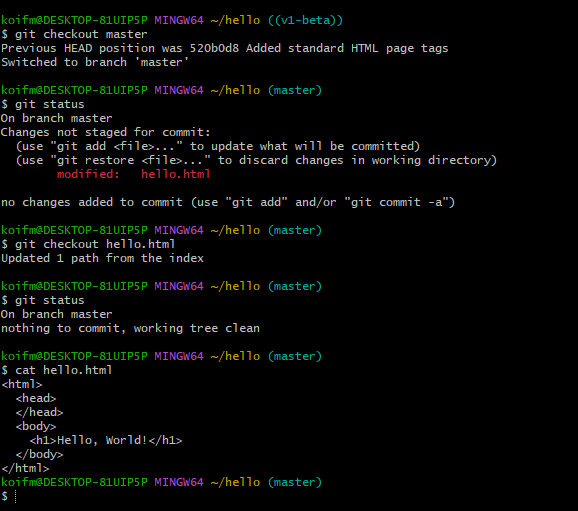
 *РИС.18*

Проверим доступные теги (рис.19):

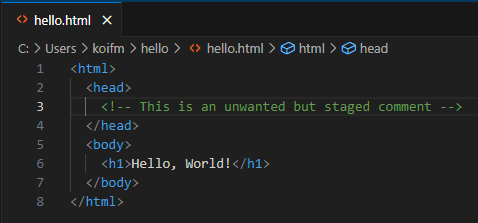
 *РИС.19*

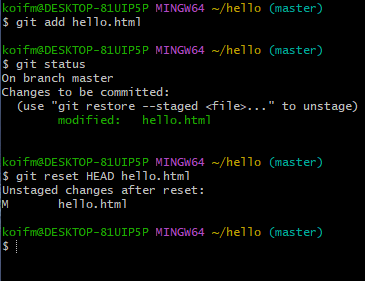
1. Теперь уделим внимание отмене локальных изменения: Переместимся на ветку master и изменим hello.html (рис.20 - рис.21):

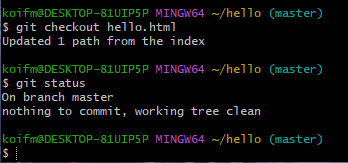
 *РИС.20*

 *РИС.21(файл был изменён, но не проиндексирован, поэтому мы успешно его откатили до предыдущей версии)*

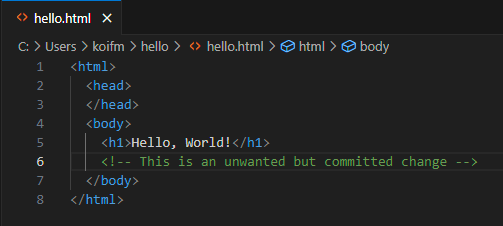
1. Теперь займёмся процесом отмены проиндексированных изменений: Переместимся на ветку master и изменим hello.html: Внесём изменение в файл hello.html в виде нежелательного комментария, проиндексируем его и проверим состояние, после чего выполним сброс буферной зоны к HEAD (рис.22 - рис.23). Теперь переключимся на версию коммита (рис.24):

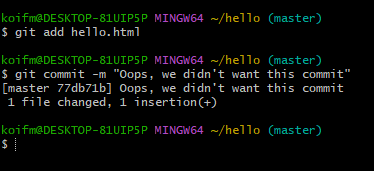
 *РИС.22*

 *РИС.23*

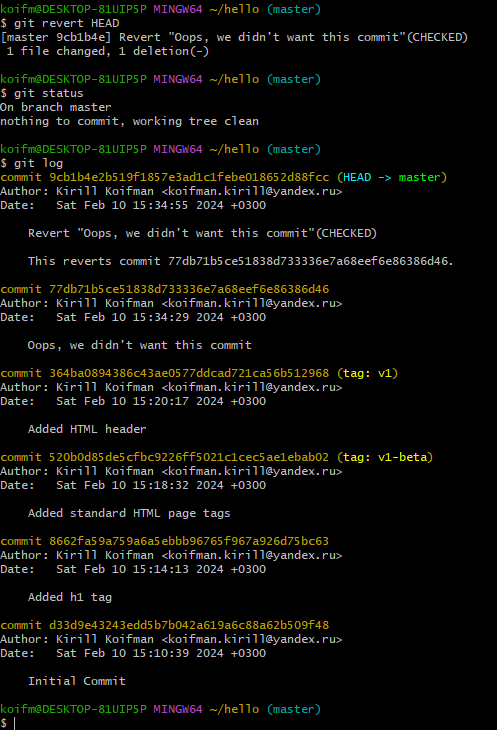
 *РИС.24(рабочий каталог снова чист)*

1. Попробуем создать коммит, а потом отменить его с помощью создания следующего коммита (рис.25 - рис.26):

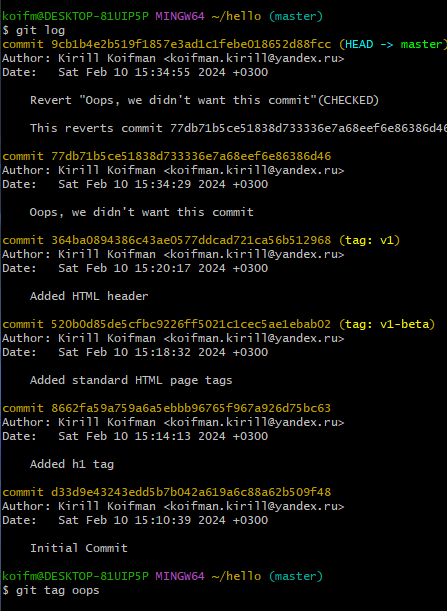
 *РИС.25*

 *РИС.26(сделали неправильный коммит)*

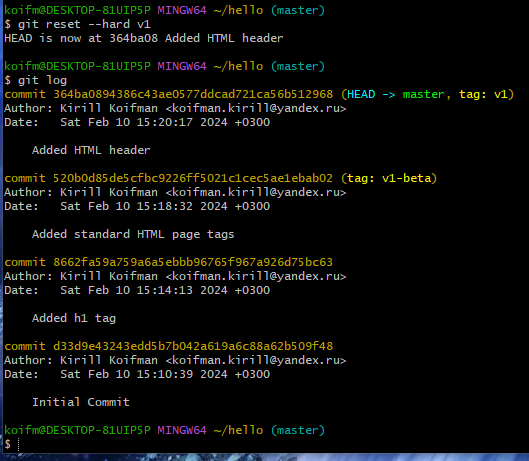
Чтобы отменить коммит, нам необходимо сделать коммит, который удаляет изме- нения, сохраненные нежелательным коммитом. После этого проверим лог (рис.27):

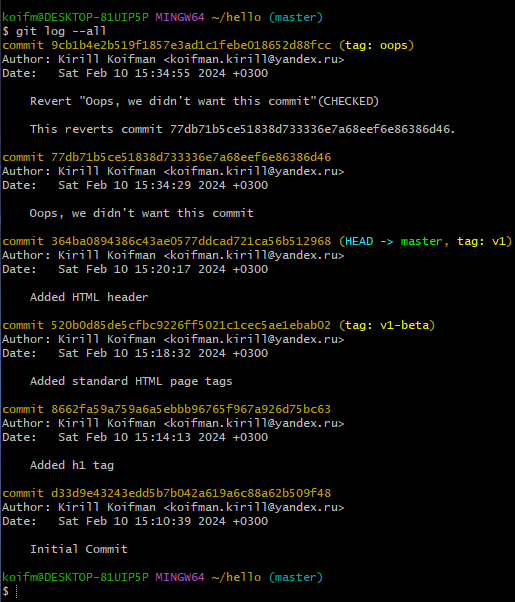
 *РИС.27(нежелытельный коммит сохранился в истории)*

1. А сейчас попробуем удалить коммит из ветки: Предположим, 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: Вновь проверим историю коммитов и отметим тегом последний (рис.28):

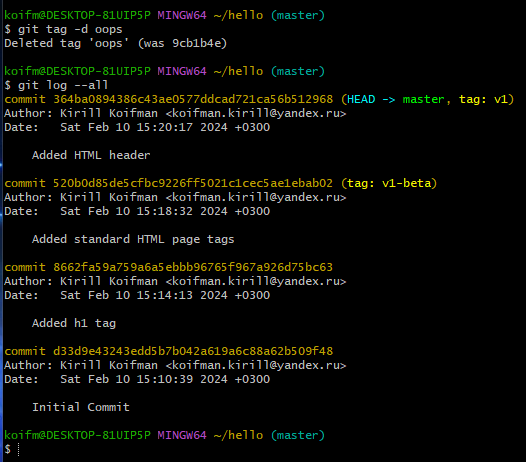
 *РИС.28*

Сбросим ветку до нужной точки (рис.29):

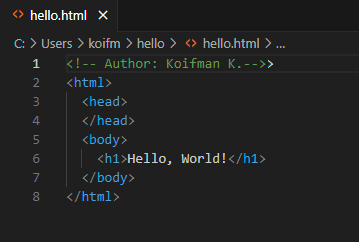
 *РИС.29*

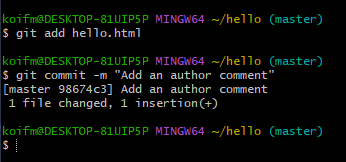
 *РИС.30*

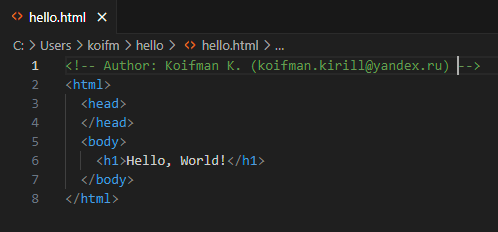
1. Удалим тег oops и коммиты, на которые он ссылался, сборщиком мусора (рис.31)

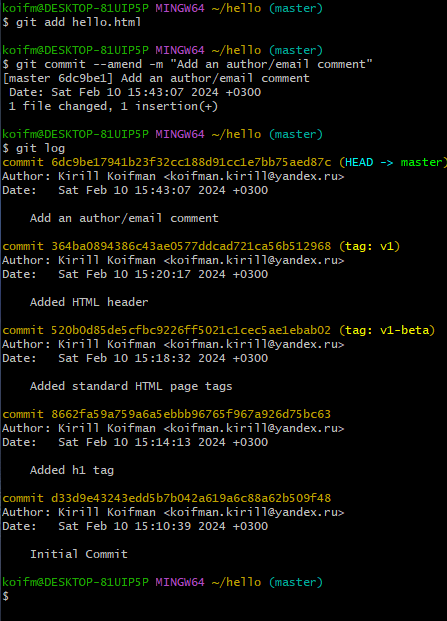
 *РИС.31(«oops» больше не будет отображаться в репозитории)*

1. Также попробуем внести изменения в сделанный коммит (рис.32 - рис.35):

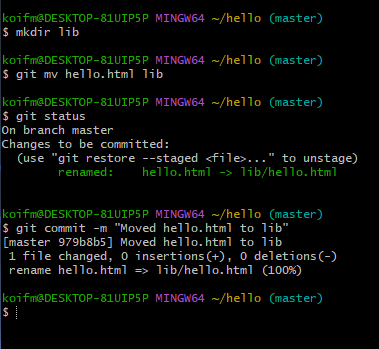
 *РИС.32*

 *РИС.33(сделали коммит)*

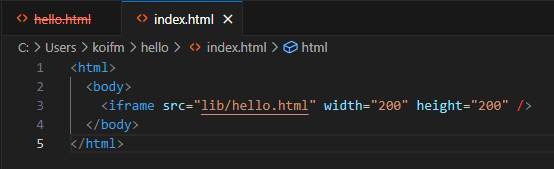
 *РИС.34*

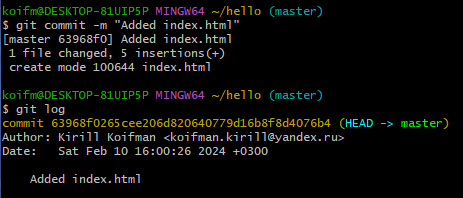
 *РИС.35(заменили предыдущий коммит новым)*

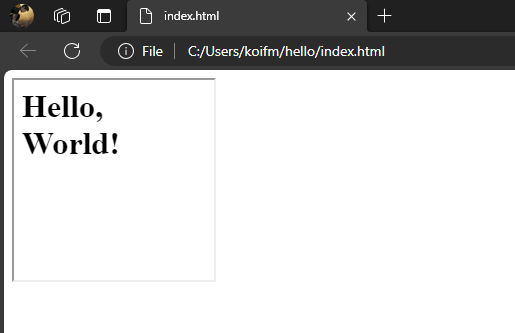
1. Сейчас мы собираемся создать структуру нашего репозитория. Давайте перенесем страницу в каталог lib (рис.36)

 *РИС.36(успешно перенесли)*

1. Добавим файл index.html в наш репозиторий (рис.37 - рис.39):

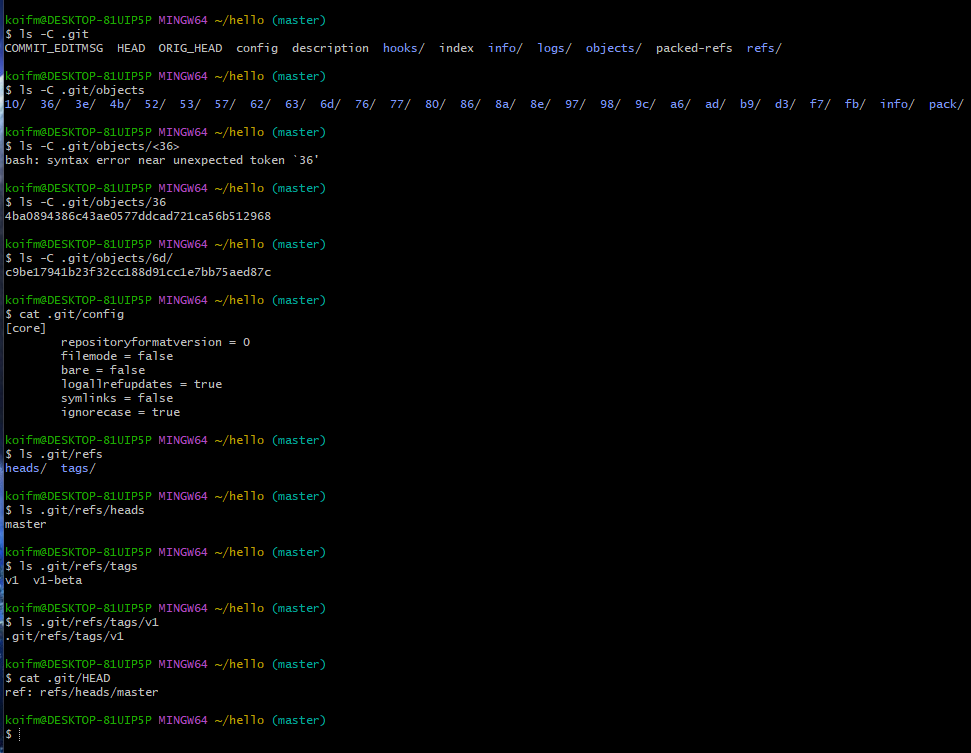
 *РИС.37*

 *РИС.38*

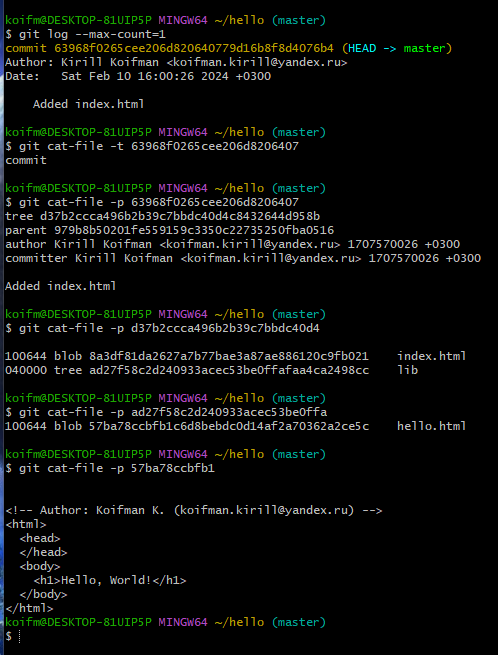
 *РИС.39(можно увидеть кусок страницы hello в маленьком окошке)*

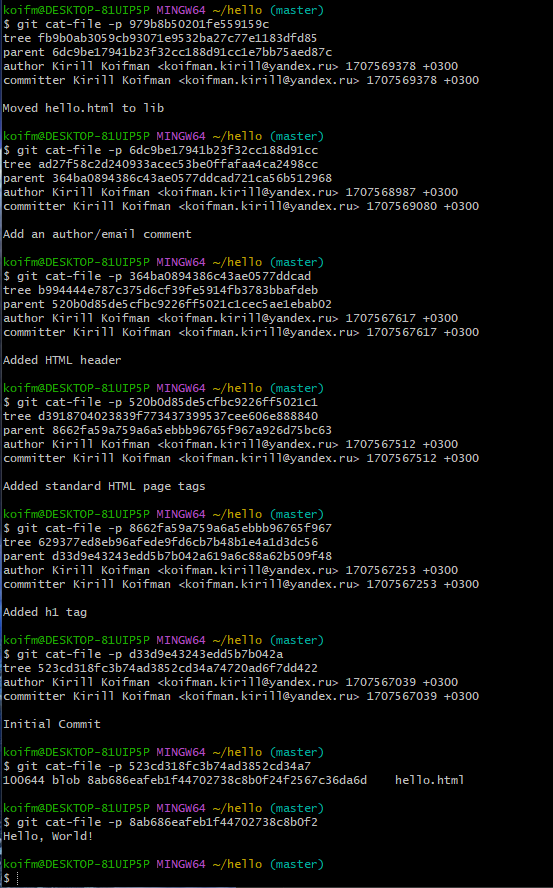
1. Рассмотрим детальней каталог .git (рис.40), где можно увидеть:

* информацию git;
* базу данных объектов;
* файл конфигурации;
* информацию о ветках, тегах;

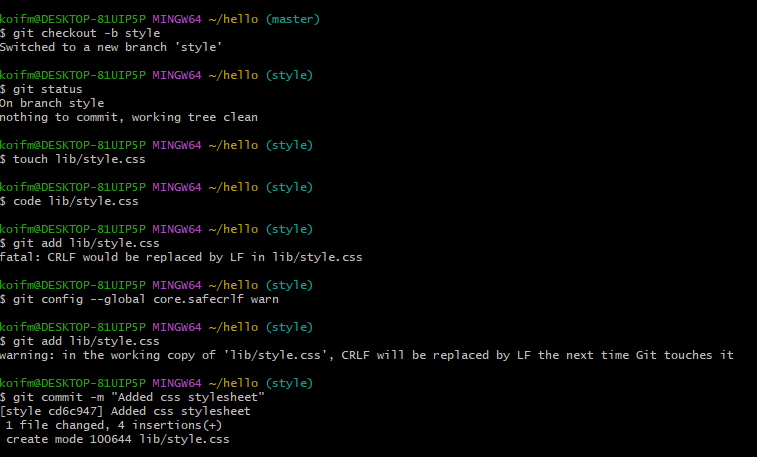
 *РИС.40*

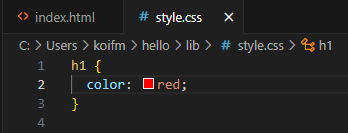
1. Исследуем информацию о наших коммитах с помощью объектов git (рис.41 - рис.42):

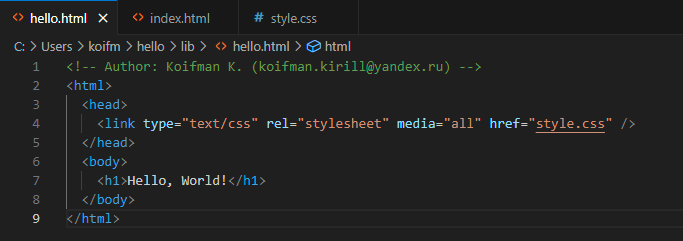
 *РИС.41(получили содержимое файла hello.html последней версии)*

 *РИС.42*

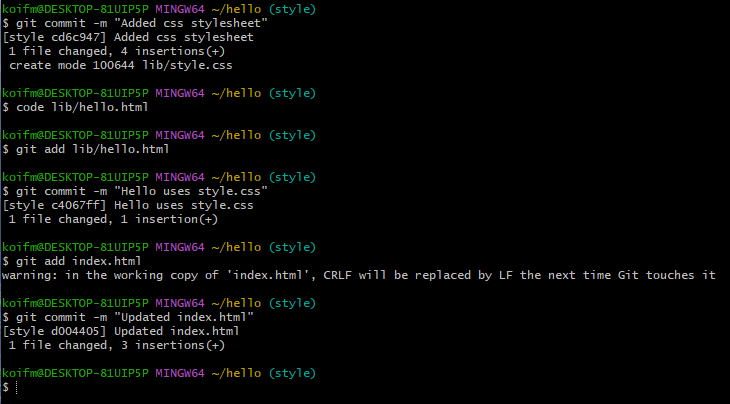
1. Создадим новую ветку (рис.43 - рис.46):

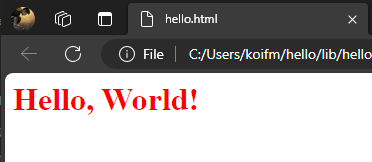
 *РИС.43(переключимся на новую ветку и закоммитим изменения)*

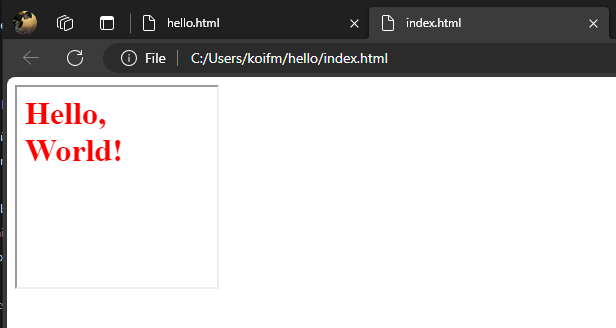
 *РИС.44(создадим таблицу стилей в новой ветве)*

 *РИС.45(изменим основную страницу)*

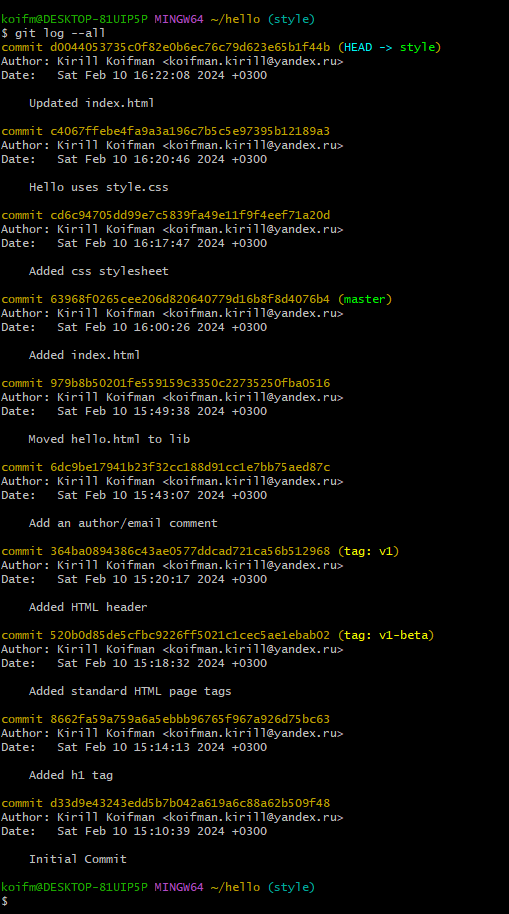
 *РИС.46(изменим вторую страницу)*

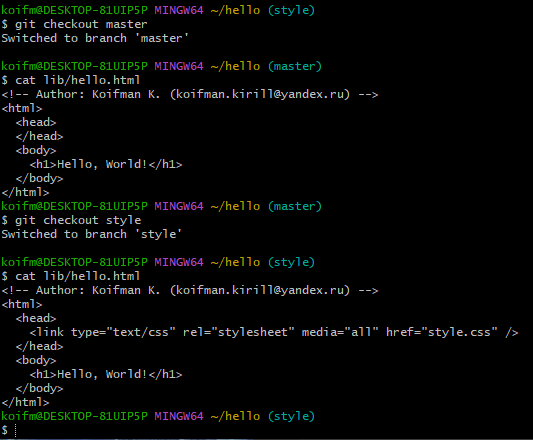
 *РИС.47(закоммитим изменения)*

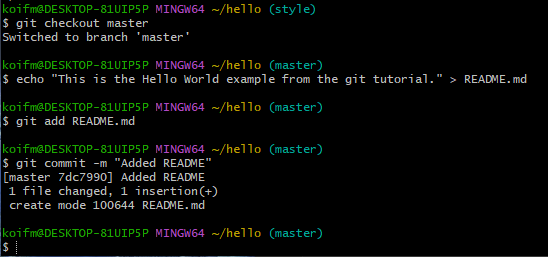
 *РИС.48(результат hello.html)*

 *РИС.49(результат index.html)*

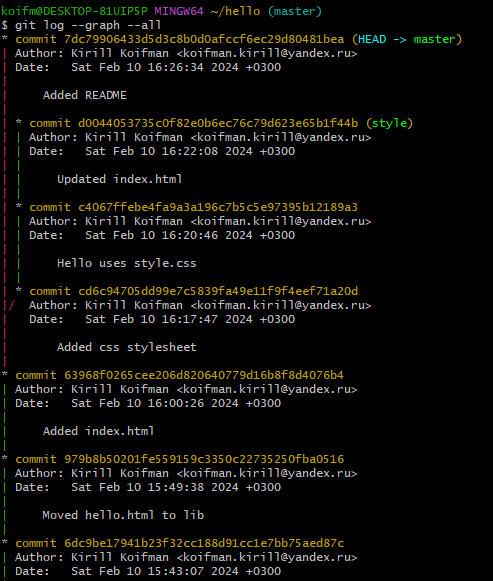
1. Теперь у нас есть 2 ветки, проверим информацию о них и попробуем произвести несколько оперций с ними (рис.50 - рис.54):

 *РИС.50(информация о последних коммитах)*

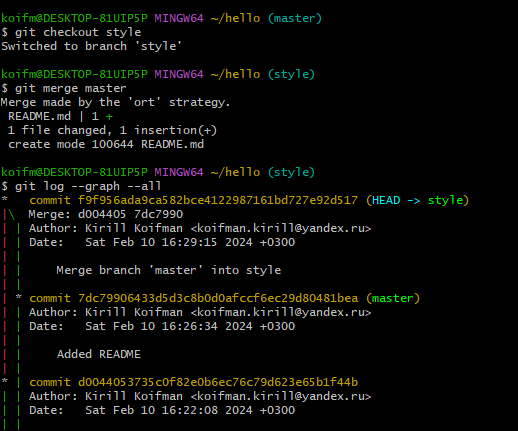
 *РИС.51(переключимся на master и проверим содержимое hello.html, после чего вернёмся к style и сделаем тоже самое)*

 *РИС.52(вернёмся в master и добавим новый файл. И сделаем коммит)*

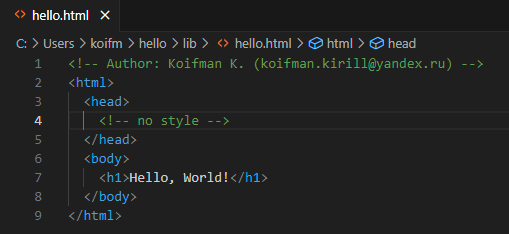
1. Сделаем коммит изменения README.md в ветку master (рис.53):

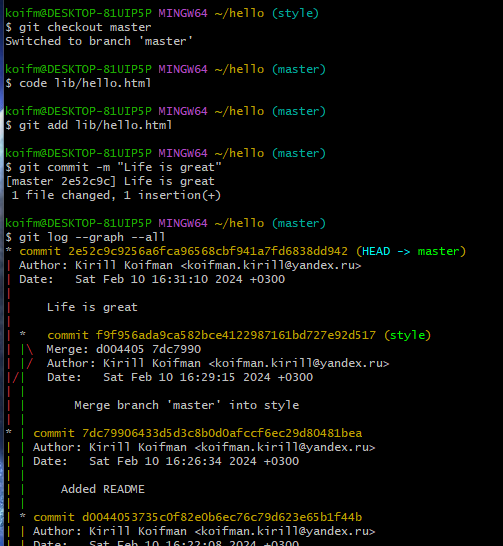
 *РИС.53(теперь есть 2 отличниме ветки)*

1. Давайте сольём эти ветки в одну (рис.54):

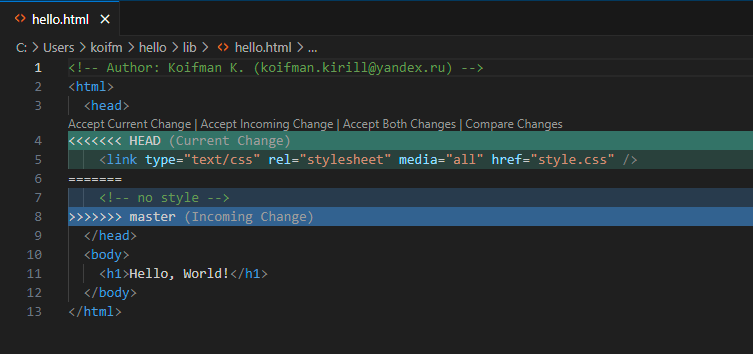
 *РИС.54(успешно слили ветки)*

1. Создадим искусственный конфликт при слиянии веток (рис.55 - рис.56):

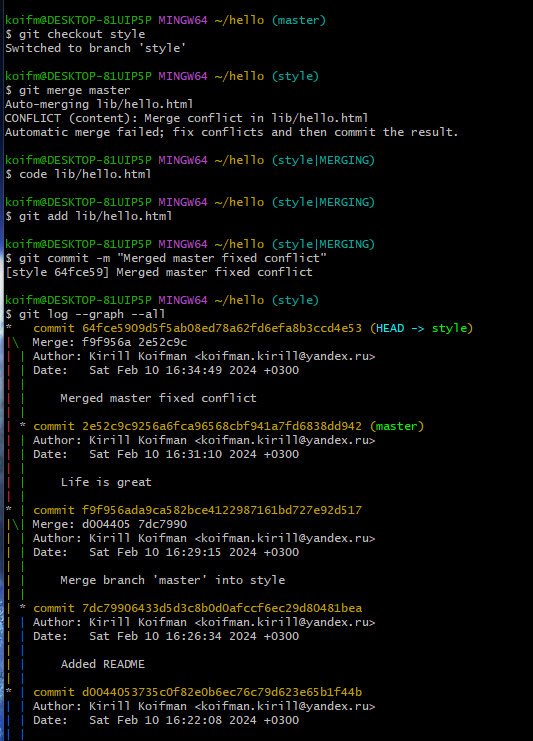
 *РИС.55(внесём изменения в ветке master)*

 *РИС.56(ветки вновь отделены)*

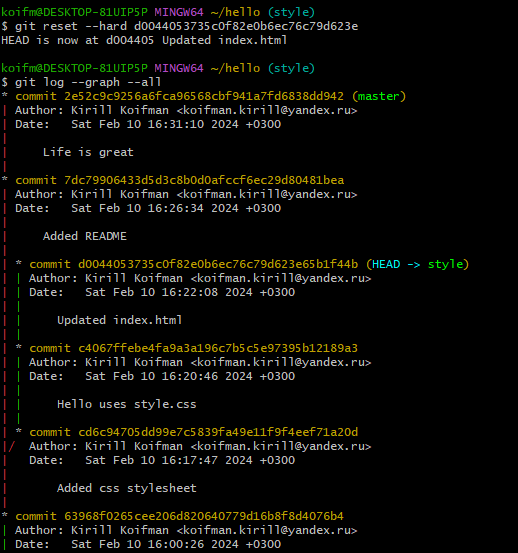
1. Решим созданный нами конфликт при слиянии веток (рис.57 - рис.59):

 *РИС.57(в файле можно увидеть причину конфликта)*

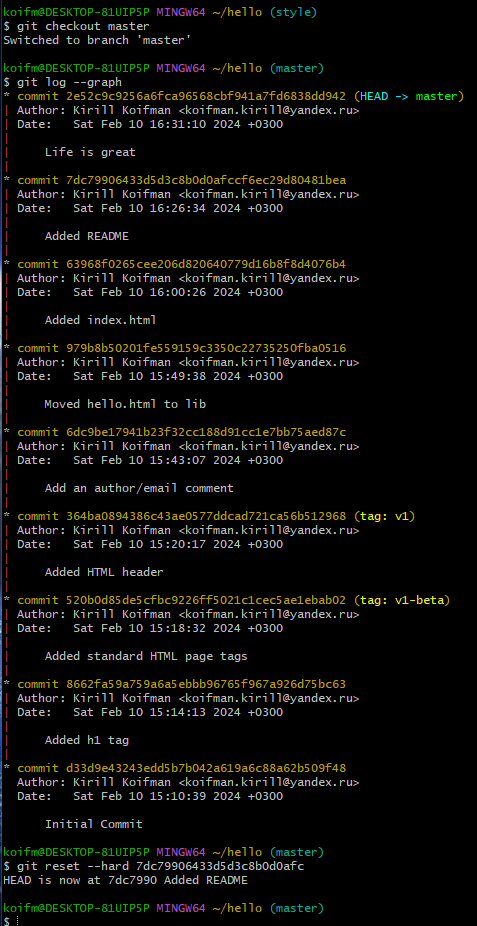
 *РИС.58(отредкатируем файл, чтобы исправить конфликт)*

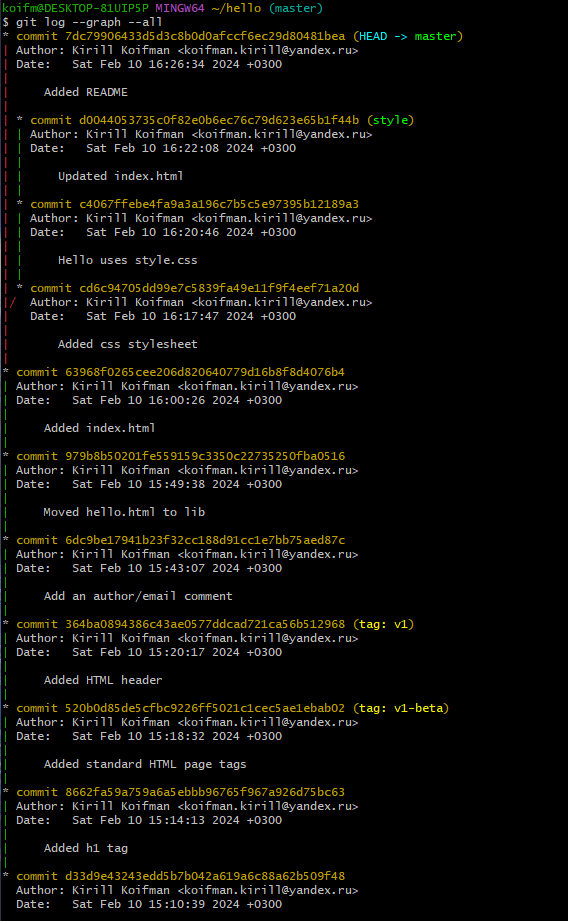
 *РИС.59(сделаем коммит. Ветки вновь успешно слились)*

1. Теперь произведём сброс ветки style (рис.60):

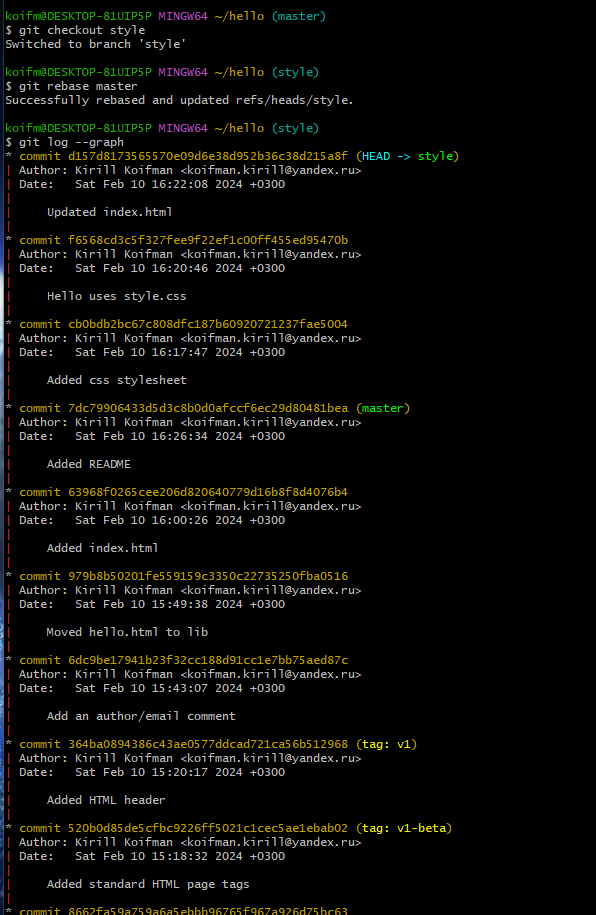
 *РИС.60(в истории больше нет коммитов слияний)*

1. Также попробуем произвезти сброс ветки style (рис.61 - рис.62):

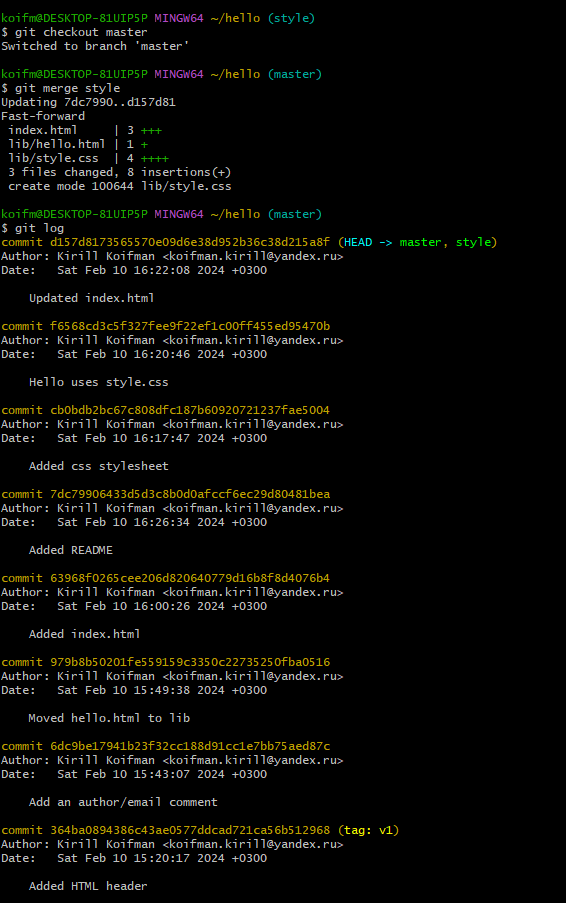
 *РИС.61*

 *РИС.62*

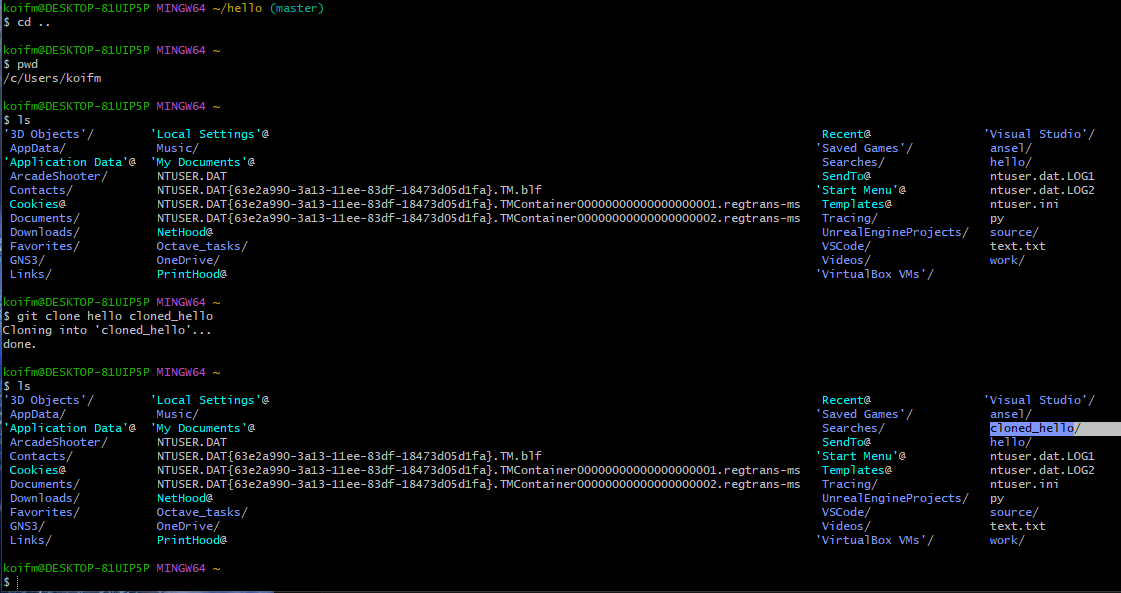
1. Воспользуемся командной base, чтобы вернуться в точку до 1-го слияния и перенести изменения из ветки master в ветку style (рис.63):

 *РИС.63(репозиторий находится в состоянии до выполнения слияния)*

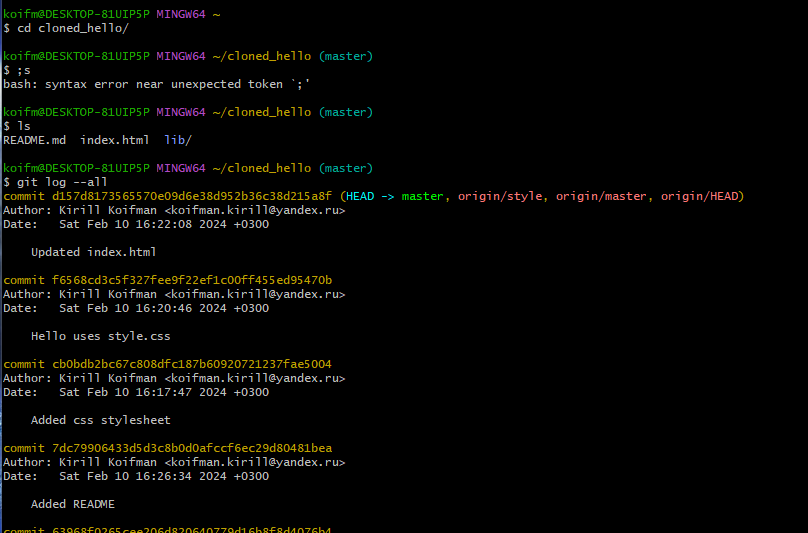
1. Теперь сольём изменения style в ветку master (рис.64):

 *РИС.64(git выполнил ускоренное слияние-перемотку без каких-либо конфликтов)*

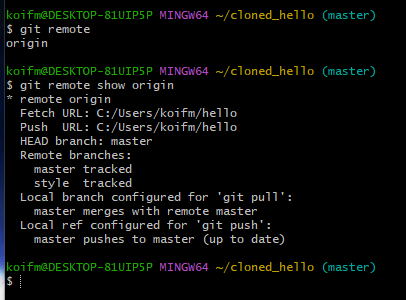
1. А сейчас займёмся клонированием репозитория (рис.65):

 *РИС.65(перейдём в рабочий каталог и клинируем репозиторий hello)*

1. Проверим содержимое клонированного репозитория (рис.66):

 *РИС.66(те же информация и коммиты, что и у исходного репозитория)*

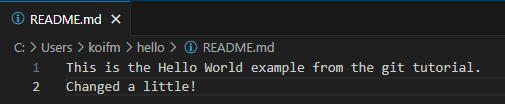
1. Проверим, что такое origin (рис.67):

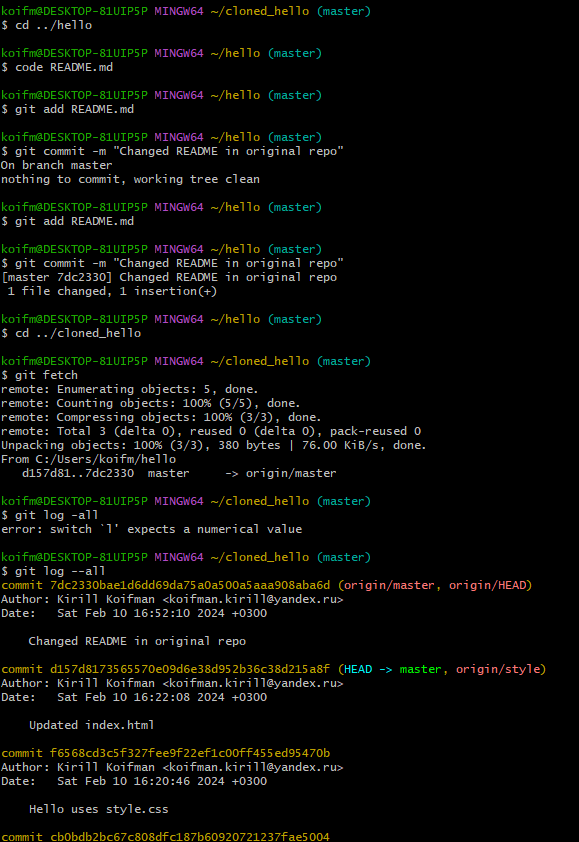
 *РИС.67(получили информацию об исходной репозитории)*

1. Проверим доступные ветки в нашем репозитории (рис.68):

 *РИС.68(можем увидеть как локальные ветки, так и удалённые)*

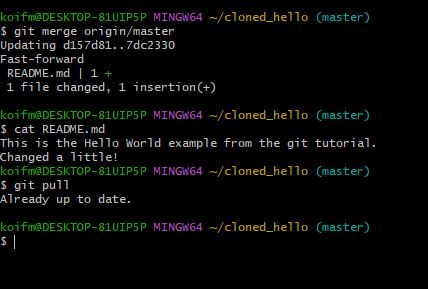
1. Попробуем внести изменения в оригинальный репозиторий, после чего извлечём и сольём изменения из удалённой ветки в текущую (рис.69 - рис.72):

 *РИС.69(изменим файл)*

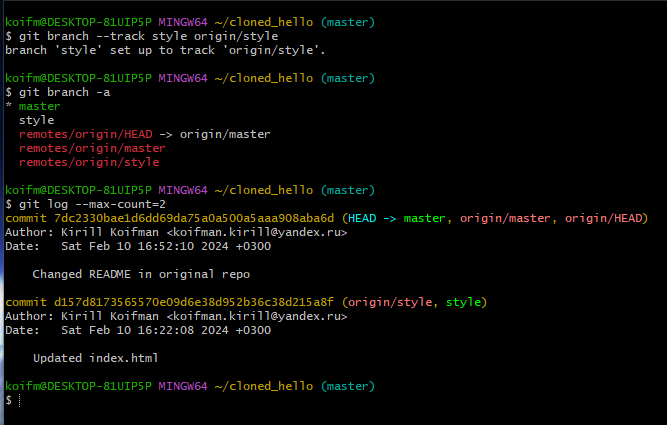
 *РИС.70(добавим изменения и сделаем коммит)*

pic *РИС.71(команда git fetch извлекла новые коммиты из удаленного репозитория, но не слила их с нашими наработками в локальных ветках)*

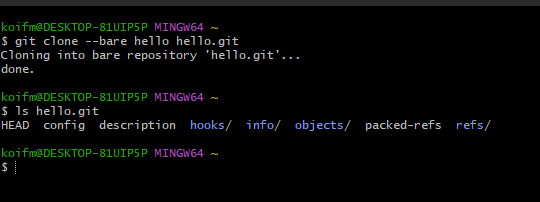
1. Произведём слияние извлечённых изменения (рис.72):

 *РИС.72(теперь изменения можно увидеть)*

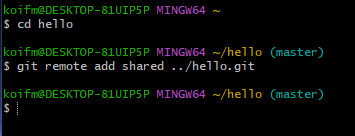
1. Давайте добавим наблюдателя, ветку которая будет отслеживать удалённую ветку (рис.73):

 *РИС.73(Теперь мы можем видеть ветку style в списке веток и логе)*

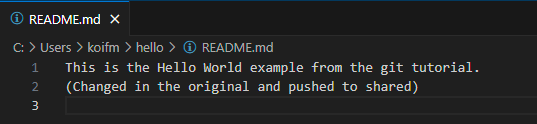
1. Создадим чистый репозиторий для расшаривания (рис.74)

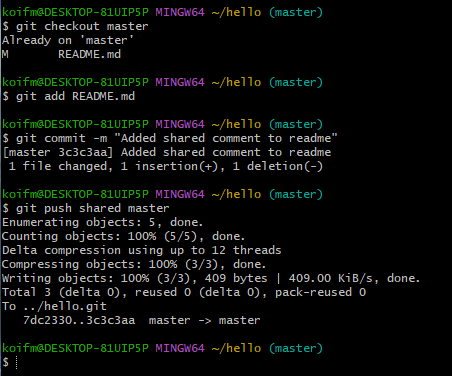
 *РИС.74*

1. Добавим удалённый репозиторий к оригинальному (рис.75):

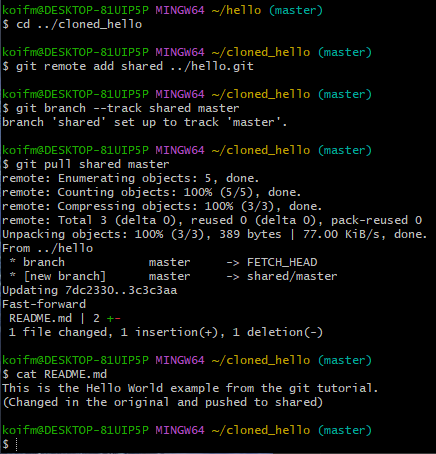
 *РИС.75*

1. Так как чистые репозитории, как правило, расшариваются на каком-нибудь сете- вом сервере, нам необходимо отправить наши изменения в другие репозитории (рис.76 - рис.77):

 *РИС.76(изменим файл)*

 *РИС.77(отправим изменения)*

1. А теперь извлечём эти изменения (рис.78):

 *РИС.78(получили отправленные изменения)*

## Заключение

В ходе проделанной лабораторной работы мной были усвоены основыне навыки создания и настройки рабочего пространства и работы с системой контроля версий Git.