Математическое моделирование. Лабораторная Работа №1

Работа с git

Барсегян Вардан Левонович

Содержание

Список иллюстраций

Список таблиц

# 1 Цель работы

Знакомство с системой контроля версий git

# 2 Задание

# 3 Теоретическое введение

Git (произносится «гит») — распределённая система управления версиями. Проект был создан Линусом Торвальдсом для управления разработкой ядра Linux, первая версия выпущена 7 апреля 2005 года; координатор — Дзюн Хамано.

Среди проектов, использующих Git, — ядро Linux, Swift, Android, Drupal, Cairo, GNU Core Utilities, Mesa, Wine, Chromium, Compiz Fusion, FlightGear, jQuery, PHP, NASM, MediaWiki, DokuWiki, Qt, ряд дистрибутивов Linux.

Программа является свободной и выпущена под лицензией GNU GPL версии 2. По умолчанию используется TCP-порт 9418.

Разработка ядра Linux велась на проприетарной системе BitKeeper, которую автор — Ларри Маквой, сам разработчик Linux — предоставил проекту по бесплатной лицензии. Разработчики, высококлассные программисты, написали несколько утилит, и для одной Эндрю Триджелл произвёл реверс-инжиниринг формата передачи данных BitKeeper. В ответ Маквой обвинил разработчиков в нарушении соглашения и отозвал лицензию, и Торвальдс взялся за новую систему: ни одна из открытых систем не позволяла тысячам программистов кооперировать свои усилия (тот же конфликт привёл к написанию Mercurial). Идеология была проста: взять подход CVS и перевернуть с ног на голову, и заодно добавить надёжности.

Начальная разработка велась меньше чем неделю: 3 апреля 2005 года разработка началась, и уже 7 апреля код Git управлялся неготовой системой. 16 июня Linux был переведён на Git, а 25 июля Торвальдс отказался от обязанностей ведущего разработчика.[1].

# 4 Выполнение лабораторной работы

## 4.1 Подготовка

1. Настроим параметры установки окончаний строк и отображения unicode для устройства Linux. После, в папке *hello* создадим файл *hello.html* с содержимым, и инициализируем пустой git-репозиторий (рис. 1).

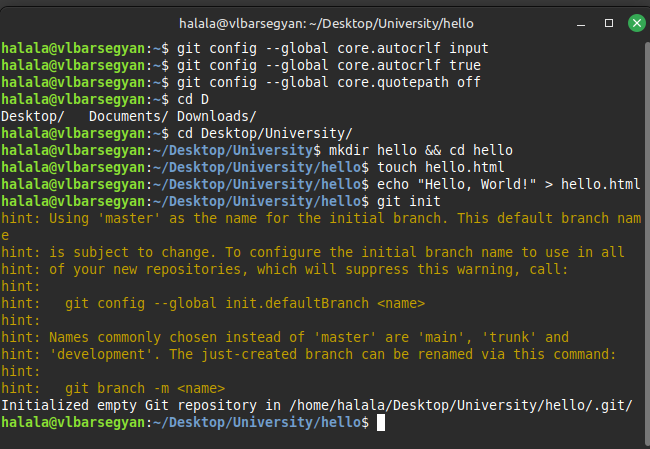


Рис. 1: Настройка git, инициализация репозитория

## 4.2 Создание проекта и внесение изменений

1. Добавим файл в репозиторий.  
   *git add hello.html*  
   *git commit -m “Initial Commit”*  
   Проверим текущее состояние репозитория командой *git status* (рис. 2).

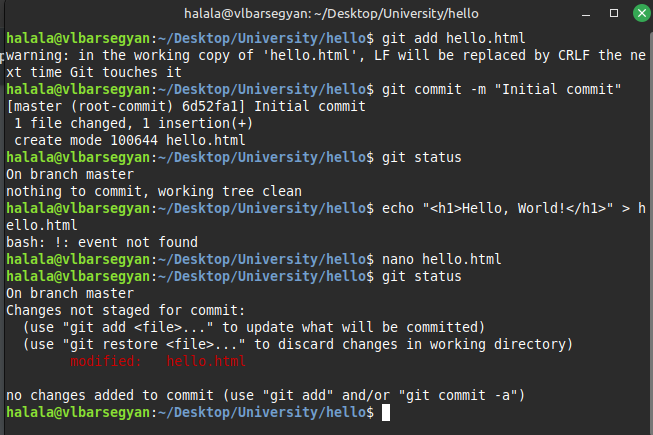


Рис. 2: Добавление файла в репозиторий и проверка состояния репозитория

## 4.3 Индексация изменений

1. Текущее содержимое файла *hello.html* (рис. 3).

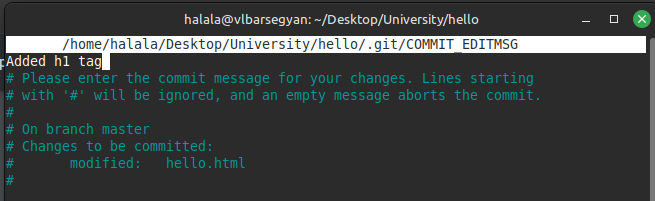


Рис. 3: hello.html

1. Индексирую изменения командой *git add hello.html* и проверяю. Теперь изменения проиндексированы.  
   Коммичу изменения командой *git commit* и снова проверяю (рис. 4).

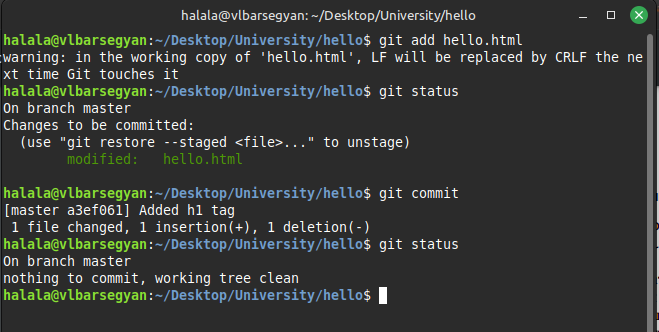


Рис. 4: Индексация и коммит изменений

1. Обновляю содержимое *hello.html* (рис. 5).

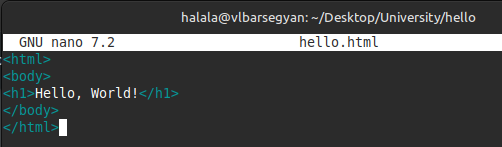


Рис. 5: hello.html

1. Добавляю это изменение в индекс командой *git add hello.html*. Снова обновляю *hello.html* (рис. 6).

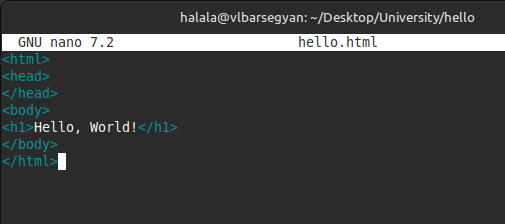


Рис. 6: Обновление hello.html

1. Проверяю текущие изменения. Добавляю последние изменения и коммичу. Снова проверяю статус (рис. 7).

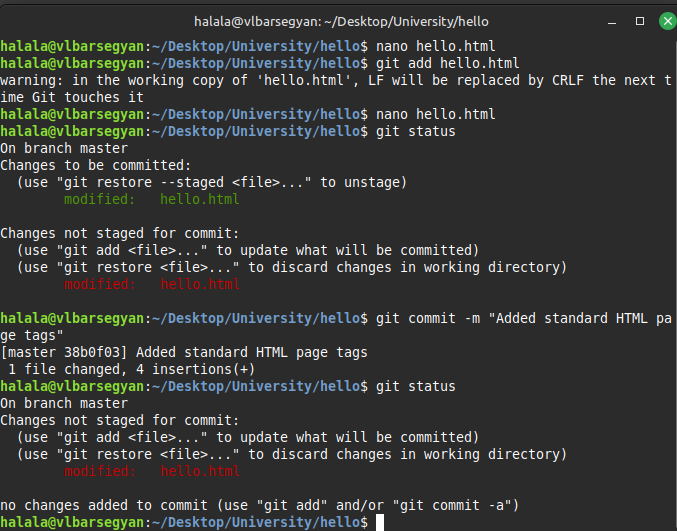


Рис. 7: Проверка статуса и коммит

1. Добавляю второе изменение в индекс и проверяю (рис. 8).

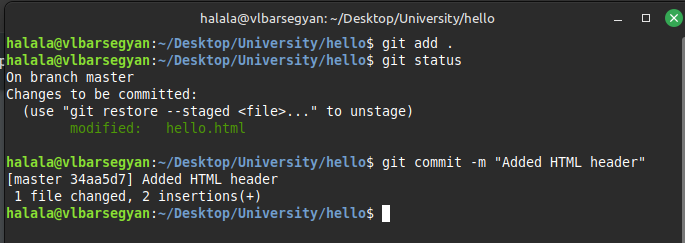


Рис. 8: Добавление второго изменения в индекс, проверка

## 4.4 История

1. Получим список изменений командой *git log*  
   Для однострочного формата введем *git log –pretty=oneline* (рис. 9).

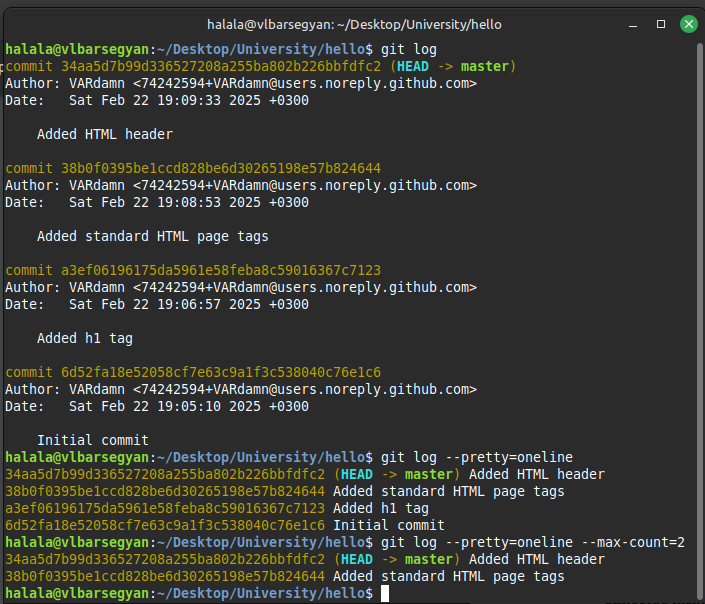


Рис. 9: Список изменений

1. Выведем справочную информацию командой *man git-log* (рис. 10).



Рис. 10: man git-log

1. Ввожу команду *git checkout* , где хэш - хэш первого коммита. Также просматриваю содержимое файла - оно равняется содержимому первого коммита (рис. 11).

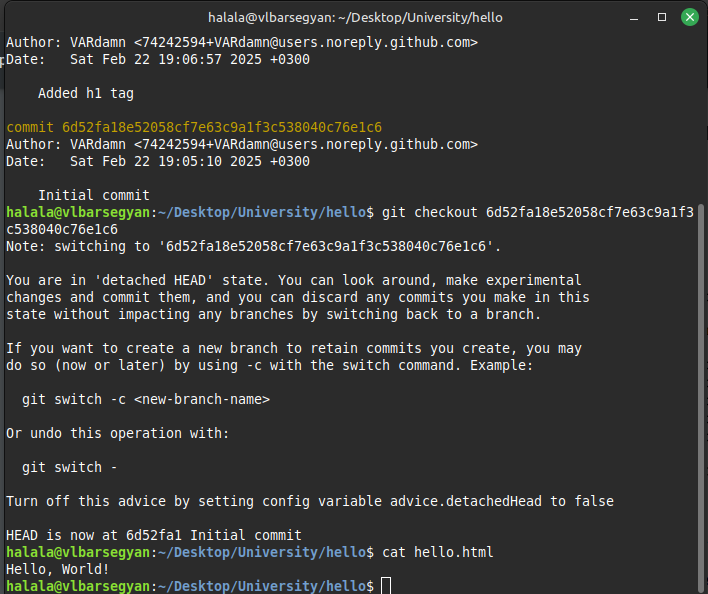


Рис. 11: Переход к первому коммиту

1. Перехожу к последней версии ветки командой *git checkout master* и просматриваю содержимое файла - теперь оно актуальное. Создаю тег первой версии командой *git tag v1*. Перехожу к версии, предшествующей v1, командой *git checkout v1^*. (рис. 12).

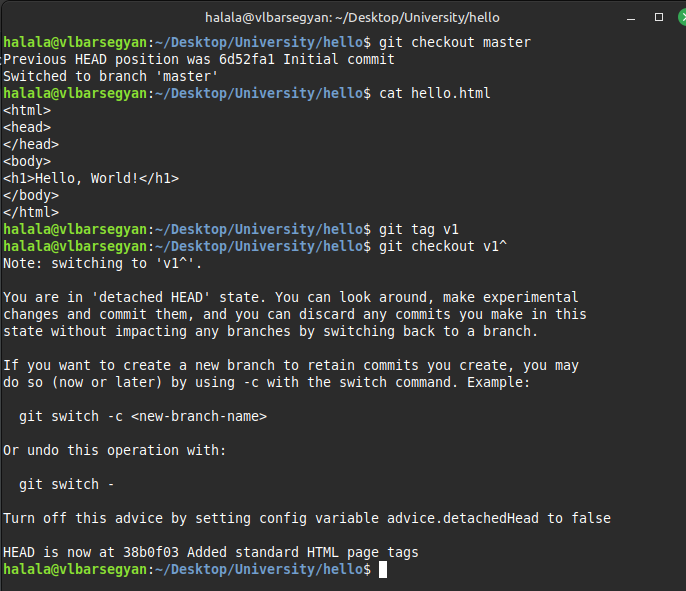


Рис. 12: Переход к последней версии и создание тега v1

1. Создаю для предшествующей версии тэг *v1-beta*. Перехожу к версии v1 и просматриваю все доступные тэги. Также просматриваю тэги в логе командой *git log master –all* (рис. 13).

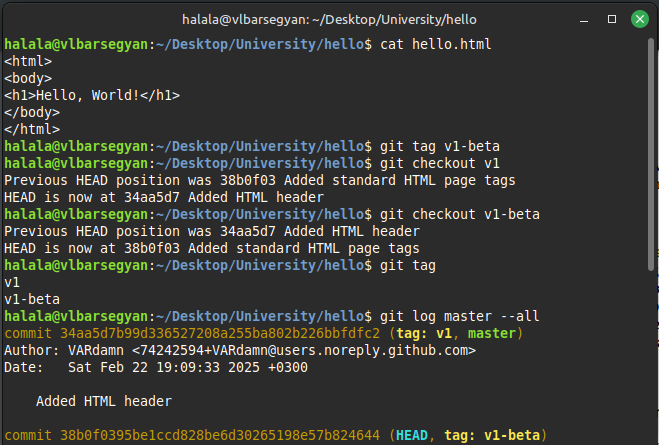


Рис. 13: Создание тэга предшествующей версии, просмотр

## 4.5 Отмена локальных изменений (до индексации)

1. Обновляю содержимое hello.html (рис. 14).

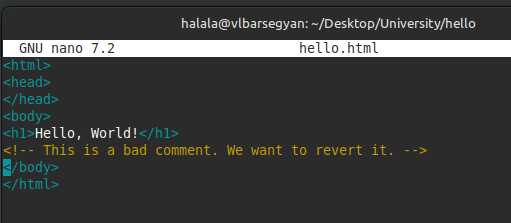


Рис. 14: hello.html

1. Проверяю состояние рабочего каталога (hello.html был изменен, но еще не проиндексирован).  
   Командой *git checkout* переключаю версии файла hello.html в репозитории  
   *git checkout hello.html*  
   *git status*  
   *cat hello.html*  
   Команда *git status* показывает нам, что не было произведено никаких изменений, не зафиксированных в рабочем каталоге. (рис. 15).

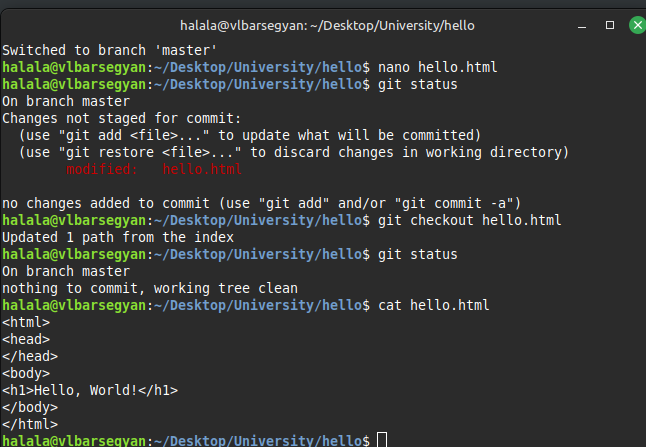


Рис. 15: Проверка состояния каталога до и после индексации изменений

## 4.6 Отмена проиндексированных изменений (перед коммитом)

1. Изменяю содержимое hello.html (рис. 16).

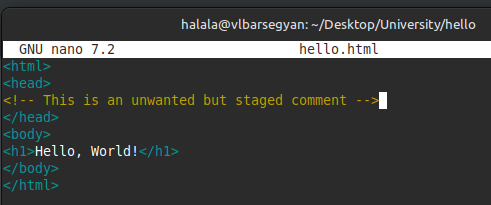


Рис. 16: hello.html

1. Индексирую изменения командой *git add hello.html*. Проверяю состояние изменения, после чего сбрасываю буферную зону командой *git reset HEAD hello.html*. Переключаюсь на версию коммита командой *git checkout hello.html* и снова проверяю статус. Проверяю содержимой файла - изменений теперь нет. (рис. 17).

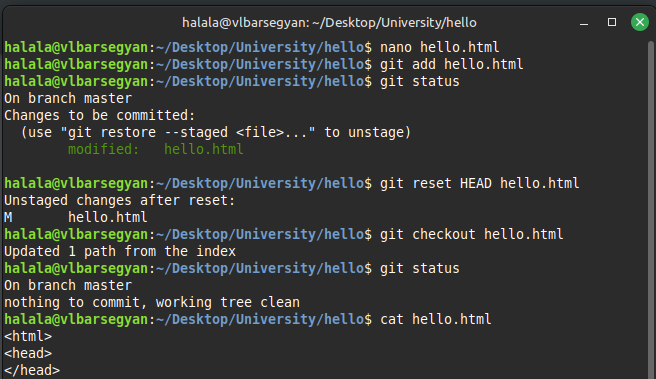


Рис. 17: Отмена проиндексированных изменений перед коммитом

## 4.7 Отмена коммитов

1. Изменяю содержимое hello.html (рис. 18).

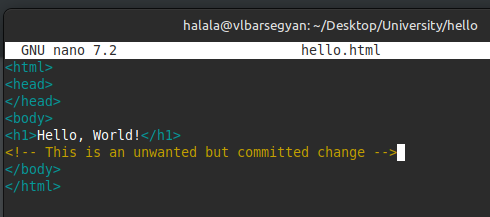


Рис. 18: hello.html

1. Индексирую изменения и коммичу их командой *git commit -m “Oops, we didn’t want this commit”*. Отменяю коммит командой *git revert HEAD* - в результате получаю такой коммит (рис. 19).

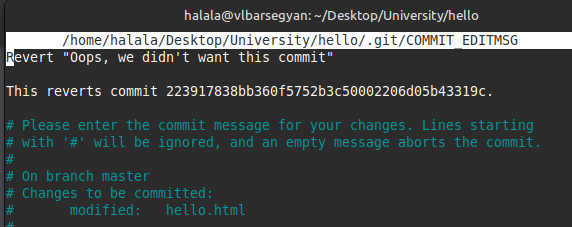


Рис. 19: Отмена коммита

1. Проверяю логи (рис. 20).

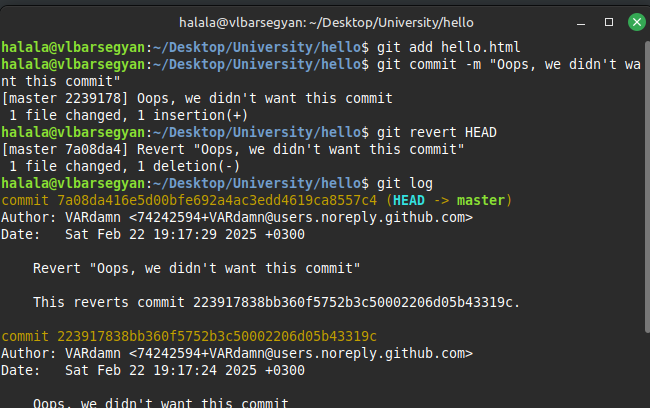


Рис. 20: Проверка логов

## 4.8 Удаление коммитов из ветки

1. Отмечаю последний коммит тэгом командой *git tag oops*. Сбрасываю ветку до точки v1 командой *git reset –hard v1*. Проверяю логи - информация о двух последних коммитах пропала (рис. 21).

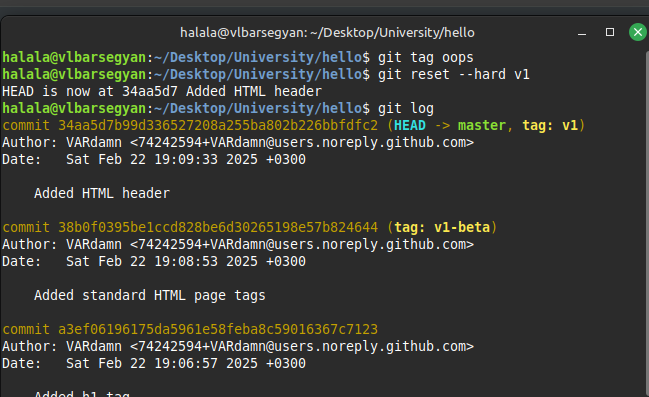


Рис. 21: Сброс ветки и проверка логов

1. Просматриваю все коммиты командой *git log –all* - здесь есть и последние два коммита (рис. 22).

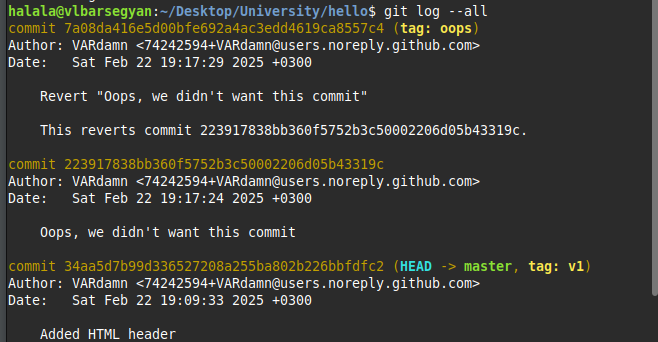


Рис. 22: Все коммиты

## 4.9 Удаление тега oops

1. Удаляю тэг oops командой *git tag -d oops* и проверяю (рис. 23).

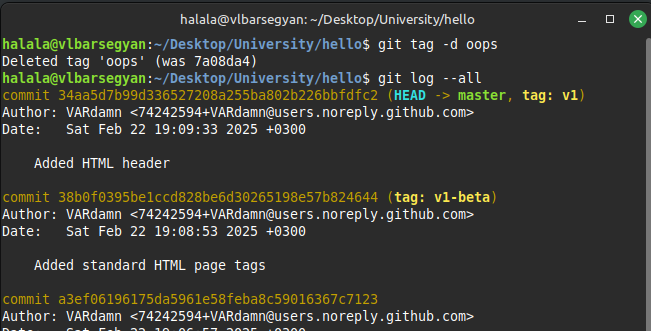


Рис. 23: Удаление тэга

## 4.10 Внесение изменений в коммиты

1. Изменяю содержимое hello.html (рис. 24).

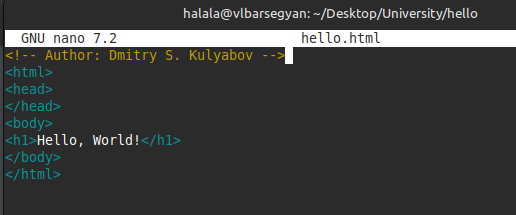


Рис. 24: hello.html

1. Индексирую изменения и коммичу их. После, снова редактирую файл, добавляя почту, и коммичу изменения командой *git commit –amend -m “Add an author/email comment”*, т.е. изменяя последний коммит. Проверяю логи и вижу один новый коммит (рис. 25).

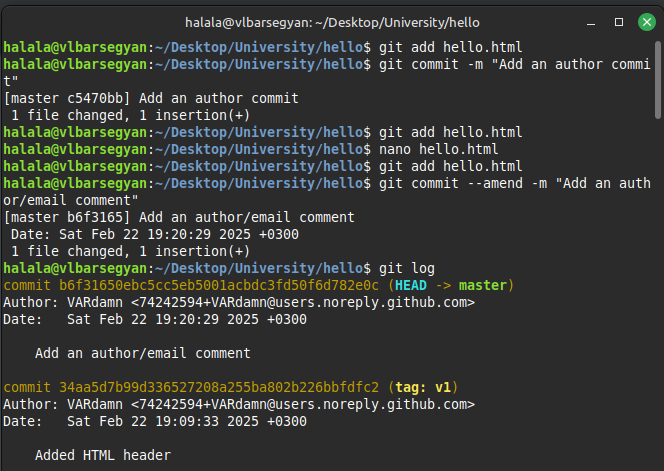


Рис. 25: Создание коммита и его изменение

## 4.11 Перемещение файлов

1. Создаю папку lib и переношу страницу в каталог lib командой *git mv*. Проверяю изменения командой *git status*. Коммичу изменения (рис. 26).

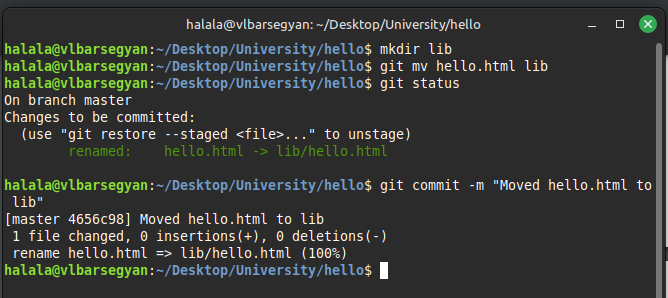


Рис. 26: Перемещение файла в папку lib

## 4.12 Подробнее о структуре

1. Создаю в корне файл *index.html* и заполняю его (рис. 27). Коммичу изменения

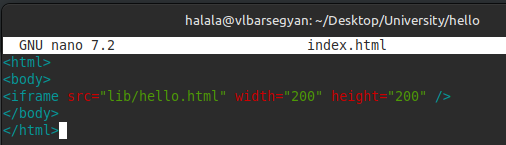


Рис. 27: index.html

## 4.13 Git внутри: Каталог .git

1. Просматриваю каталог со всей информацией о git командой *ls -C .git*. Выполняю команду *ls -C .git/objects*. Также просматриваю один из каталогов с именем из 2 букв. Просматриваю файл конфигурации командой *cat .git/config*. Далее выполняю  
   *ls .git/refs*  
   *ls .git/refs/heads*  
   *ls .git/refs/tags*  
   *cat .git/refs/tags/v1* (рис. 28).

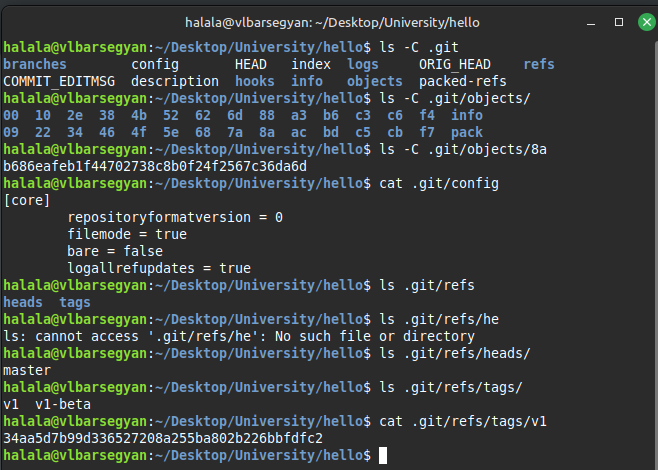


Рис. 28: Просмотр каталога .git

## 4.14 Работа непосредственно с объектами git

1. Выполняю команду *cat .git/HEAD* - содержится ссылка на ветку master. Просматриваю последний коммит командой *git log –max-count=1*. Также вывожу последний коммит с помощью хэша командой *git cat-file* (рис. 29).

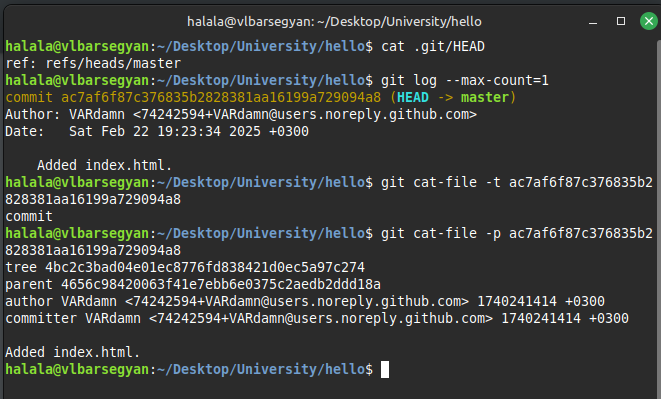


Рис. 29: Просмотр последнего коммита

1. Произвожу поиск дерева командой *git cat-file*, указав хеш дерева. Аналогично вывожу каталог lib и файл hello.html (рис. 30).

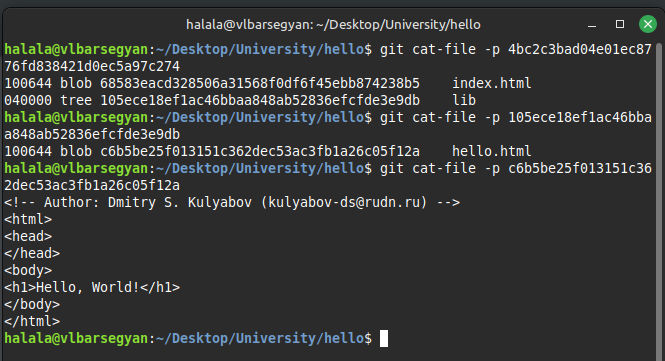


Рис. 30: Поиск по хэшу

1. Исследую репозиторий самостоятельно: перехожу к последующему коммиту по хэшу родителя (рис. 31).

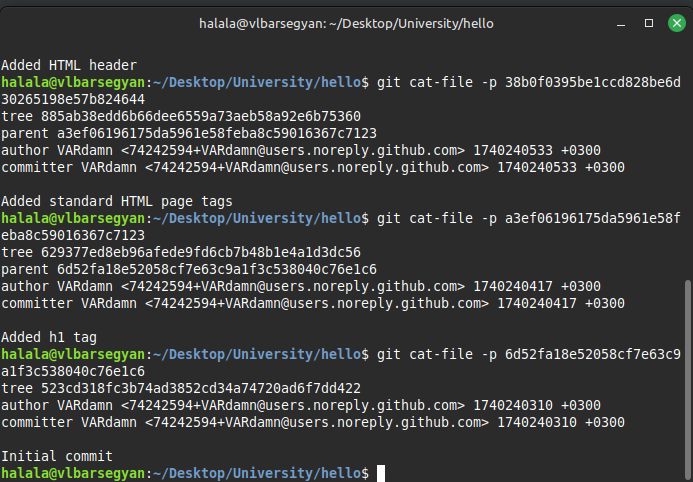


Рис. 31: Исследование репозитория

## 4.15 Создание ветки

1. Создаю ветку командой *git checkout -b style* и проверяю ее статус. Создаю файл style.css в папке lib, заполняю его, индексирую и коммичу. Обновляю файл lib/hello.html, добавляя в него ссылку на css-файл, и также индексирую и коммичу изменения. Обновляю файл index.html, индексирую и коммичу изменения (рис. 32).

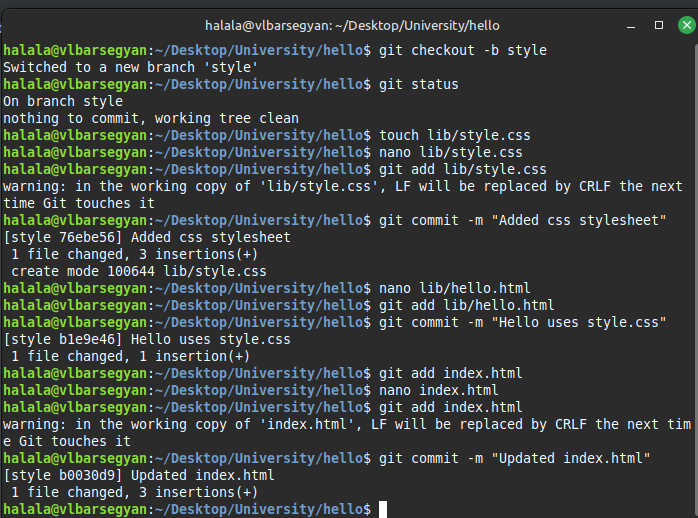


Рис. 32: Обновление файлов, индексация и коммиты

## 4.16 Навигация по веткам

1. Проверяю логи последних коммитов, вижу все созданные коммиты, они в ветке style (рис. 33).

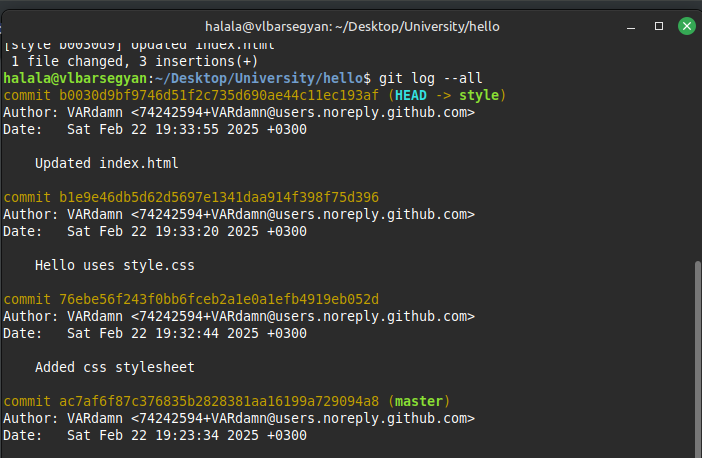


Рис. 33: Проверка логов

1. Перехожу в ветку master и просматриваю содержимое файла lib/hello.html - в нем нет последних изменений. Просматриваю тот же файл в ветке style - изменения есть (рис. 34).

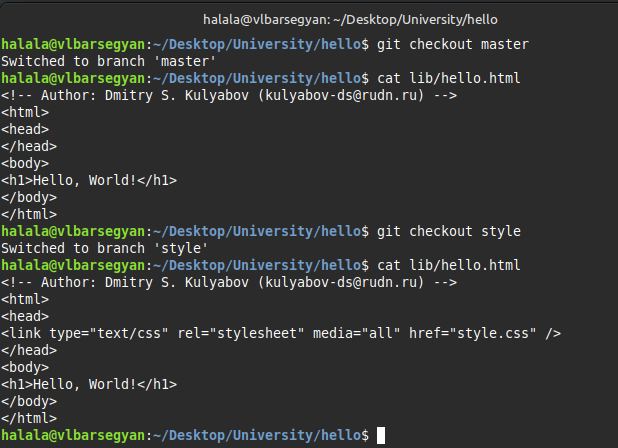


Рис. 34: Проверка файла в разных ветках

## 4.17 Изменения в ветке master

1. Перехожу в ветку master и создаю файл README.md командой *echo “This is the Hello World example from the git tutorial.” > README.md*. Коммичу изменения и просматриваю логи, включая различия между ветками. (рис. 35).

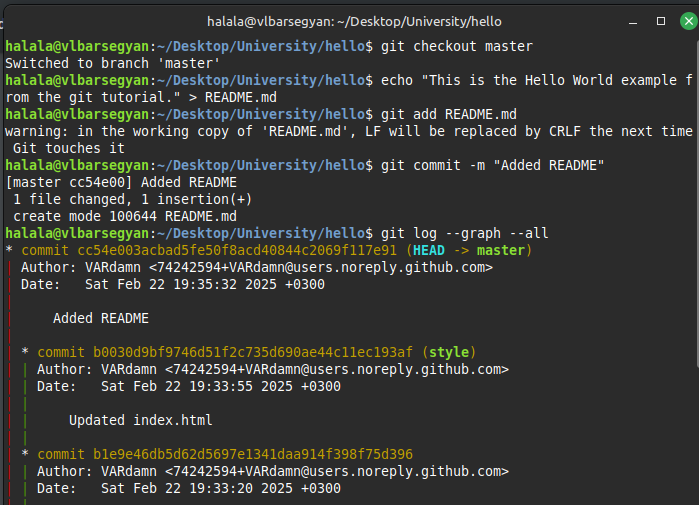


Рис. 35: Создание README в ветке master, проверка дерева веток

## 4.18 Слияние

1. Перехожу в ветку style командой *git checkout style*. Сливаю master c style командой *git merge master*. Вывожу логи с деревом всех веток (рис. 36).

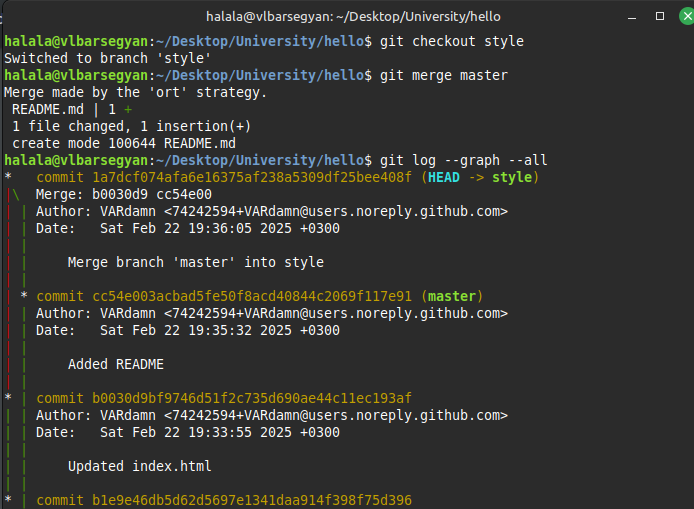


Рис. 36: Слияние master с style

## 4.19 Создание конфликта

1. Возвращаюсь в ветку master и редактирую файл lib/hello.html. Коммичу изменения, просматриваю логи (рис. 37).

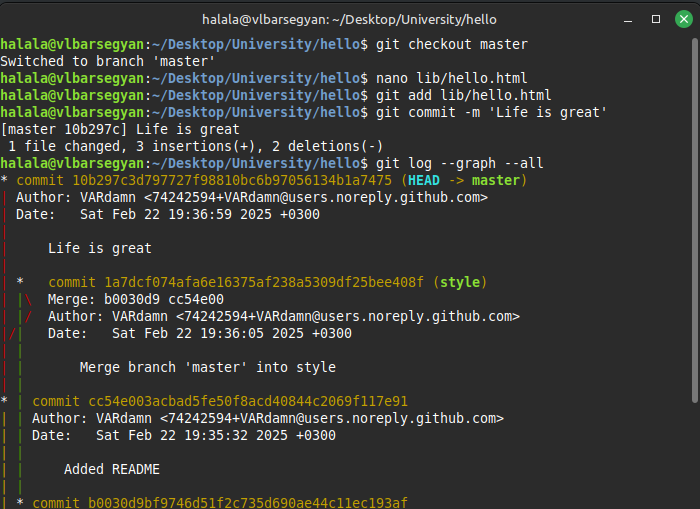


Рис. 37: Редактирование lib/hello.html в master

## 4.20 Разрешение конфликтов

1. Перехожу в ветку style и пытаюсь слить master с style. В файле lib/hello.html возникли конфликты. Разрешаю конфликты и коммичу изменения (рис. 38).

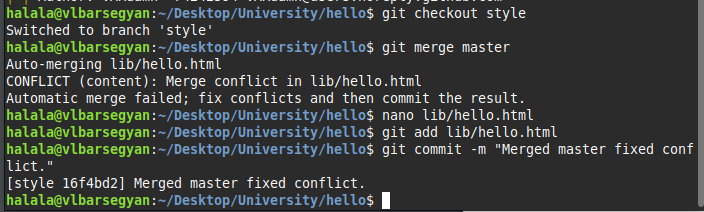


Рис. 38: Разрешение конфликта

## 4.21 Сброс ветки style

1. Перехожу в ветку style и просматриваю логи изменений (рис. 39).

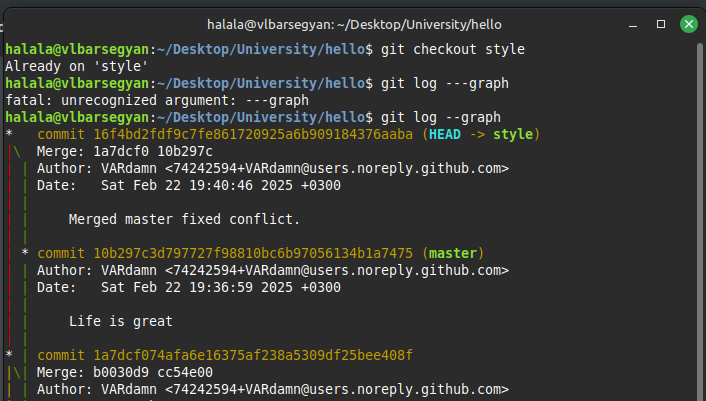


Рис. 39: Логи изменений ветки style

1. Сбрасываю ветку к коммиту «Updated index.html» командой *git reset –hard*  c использованием хэша коммита. Снова првоеряю лог ветки style (рис. 40).

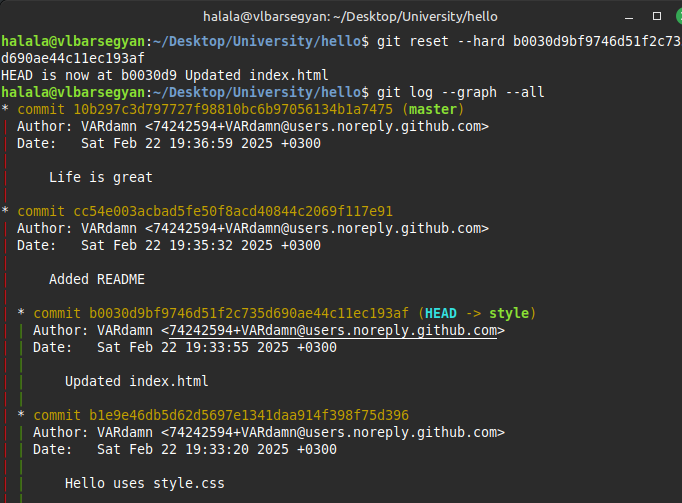


Рис. 40: Сброс ветки style с помощью git reset

## 4.22 Сброс ветки master

1. Перехожу в ветку master и аналогичным образом сбрасываю ее к коммиту «Added README». Просматриваю лог ветки master (рис. 41).

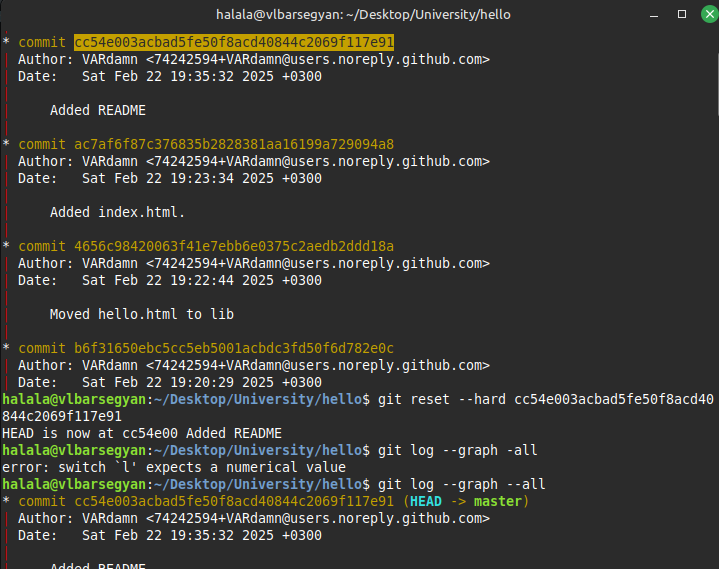


Рис. 41: Сброс ветки master

## 4.23 Перебазирование

1. Перехожу в ветку style и использую вместо слияния команду *git rebase master*. Просматриваю лог ветки. (рис. 42).

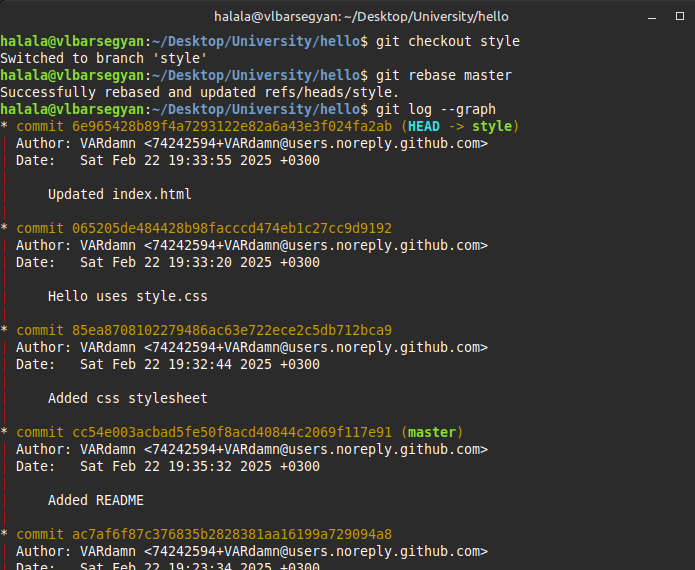


Рис. 42: Перебазирование ветки

## 4.24 Слияние в ветку master

1. Перехожу в ветку master и сливаю style c master. Просматриваю лог - теперь ветки style и master идентичны. (рис. 43).

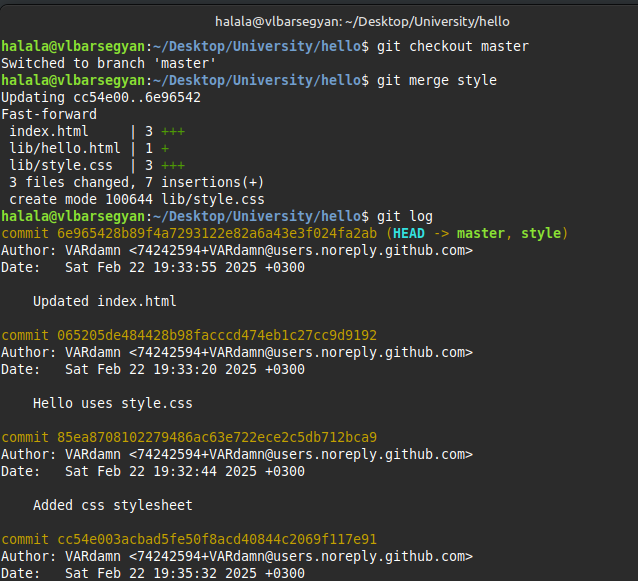


Рис. 43: Слияние style c master

## 4.25 Клонирование репозиториев и просмотр клонированного репозитория

1. Перехожу в родительский каталог. Создаю клон репозитория hello в папке cloned\_hello командой *git clone hello cloned\_hello*. Перехожу в cloned\_hello и просматриваю папки и файлы. Вывожу список всех коммитов командой *git log –all* (рис. 44).

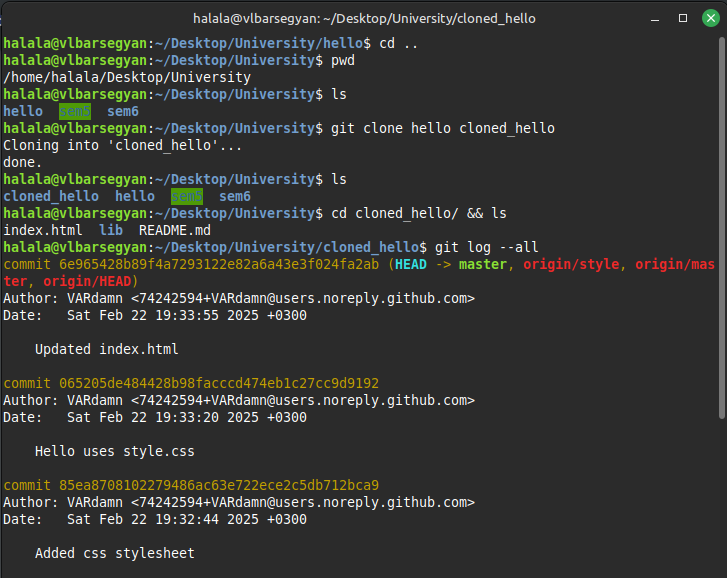


Рис. 44: Создание и просмотр клонированного репозитория

## 4.26 Что такое origin? Удаленные ветки

1. Выполняю команды *git remote* и *git remote show origin* для более подробнной информации. Просматриваю ветки, доступные в клонированном репозитории, командой *git branch*. Просматриваю все ветки командой *git branch -а* (рис. 45).

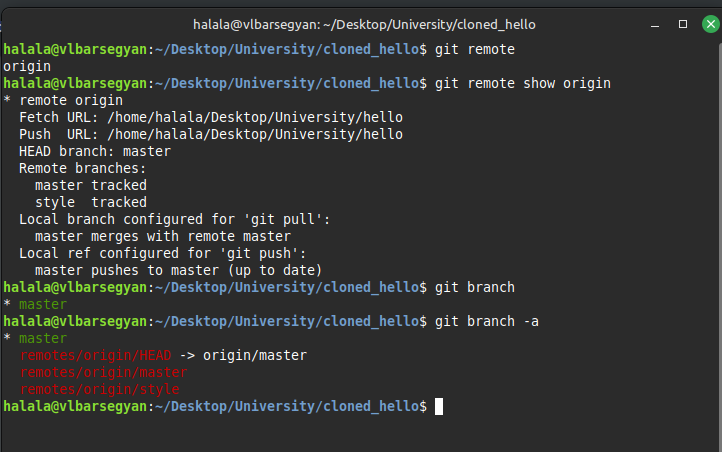


Рис. 45: git remote, просмотр веток

## 4.27 Изменение оригинального репозитория

1. Редактирую README.md в репозитории hello и коммичу изменения. В клонированном репозитории выполняю *git fetch* для извлечения изменений, проверяю лог - вижу последний сделанный коммит (рис. 46).

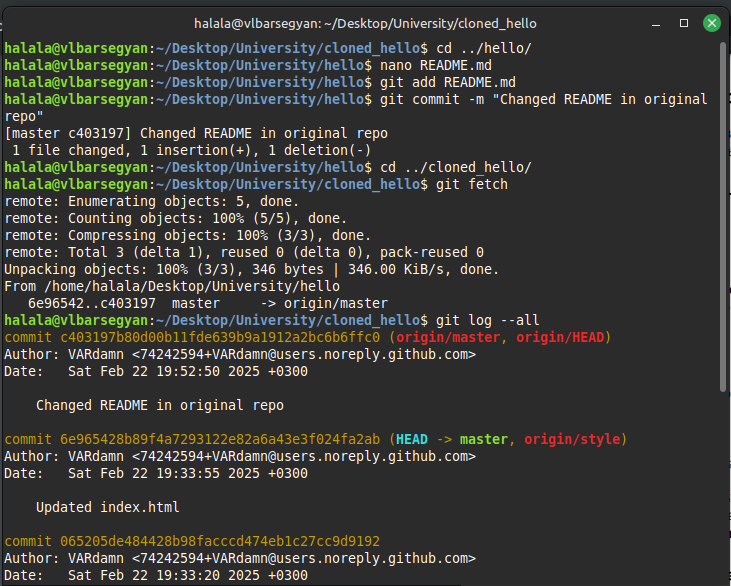


Рис. 46: Редактирование оригинального репозитория и извлечение изменений в клонированном

## 4.28 Слияние извлеченных изменений

1. Проверяю клонированный файл README.md - он не изменился. Выполняю слияние извлеченных изменений командой *git merge origin/master*. Снова проверяю содержимое файла - теперь оно обновилось. (рис. 47).

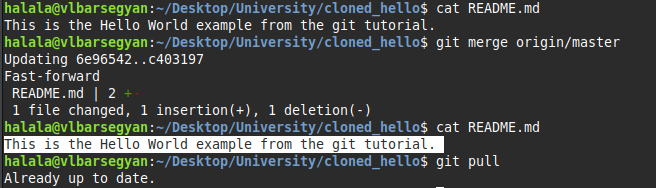


Рис. 47: Проверка файла до и после слияния изменений

## 4.29 Добавление ветки наблюдения

1. Выполняю:  
   *git branch –track style origin/style*  
   *git branch -a*  
   *git log –max-count=2*  
   Теперь можно видеть ветку style в списке веток и логе (рис. 48).

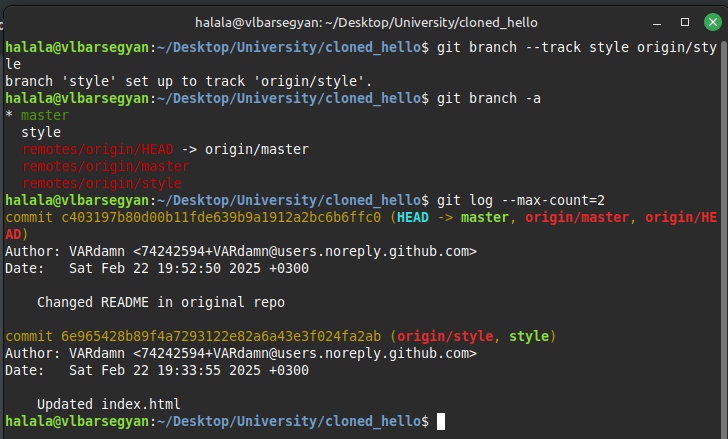


Рис. 48: Добавление ветки наблюдения

## 4.30 Создание чистого репозитория, добавление удаленного репозитория

1. Создаю чистый репозиторий командой *git clone –bare hello hello.git* и перехожу в него. Добавляю удаленный репозиторий командой *git remote add shared ../hello.git*. Редактирую файл README.md и отправляю изменения в общий репозиторий. В клонированном репозитории добавляю командой *git remote add shared ../hello.git*, и извлекаю сделанные изменения командой *git pull shared master*, проверяю файл (рис. 49).

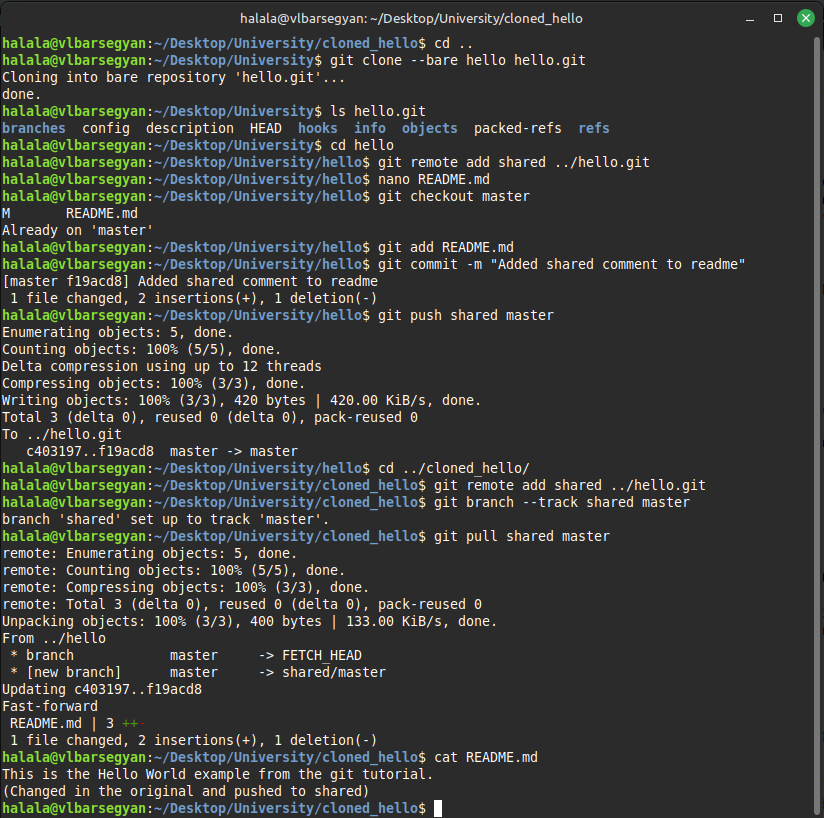


Рис. 49: Создание чистого репозитория, добавление удаленного репозитория, извлечение общих изменений

# 5 Выводы

Я настроил git, научился с ним работать, узнал про различные ветки, слияние, перебазирование, чистые репозитории.

# Список литературы

1. Git [Электронный ресурс]. 2025. URL: <https://ru.wikipedia.org/wiki/Git>.