



МИНОБРНАУКИ РОССИИ

*Федеральное государственное бюджетное образовательное учреждение  
высшего образования*

*«МИРЭА — Российский технологический университет»*

***РТУ МИРЭА***

---

Институт Информационных технологий (ИТ)

Кафедра Математического обеспечения и стандартизации  
информационных технологий (МОСИТ)

**Отчет по практической работе №1**

**по дисциплине**

**«Технология разработки программных приложений»**

**Тема: «Основные команды Git»**

Выполнил студент группы ИКБО-06-21      Бондарь А.Р.

Принял      Туманова М.Б.

Практическая работа выполнена    «\_»\_\_\_\_\_2023г.    (подпись студента)

«Зачтено»    «\_»\_\_\_\_\_2023 г.    (подпись руководителя)

Москва 2023

# Содержание

<b>Цель практической работы</b>	<b>3</b>
<b>1 Основные команды Git</b>	<b>4</b>
1.1 Установка и настройка клиент git . . . . .	4
1.1.1 Установка git . . . . .	4
1.1.2 Настройка git . . . . .	4
1.2 Созданиe локальный репозиторий и добавление в него несколько файлов . . . . .	6
1.3 Добавление файлов в репозиторий . . . . .	6
1.4 Изменение файла . . . . .	7
1.5 Коммит . . . . .	7
1.6 Как работает индексация . . . . .	8
1.7 История коммитов . . . . .	9
1.8 Отмена изменений . . . . .	10
1.9 Создание тега . . . . .	11
1.9.1 Аннотированные теги . . . . .	11
1.9.2 Легковесный теги . . . . .	12
1.10 Отмена изменений в индексе . . . . .	12
1.10.1 Отмена изменений (до индексации) . . . . .	12
1.10.2 Отмена изменений (после индексации) . . . . .	12
1.11 Отмена коммита . . . . .	14

# Цель практической работы

Получить навыки по работе с командной строкой и git'ом.

# Глава 1

## Основные команды Git

### 1.1 Установка и настройка клиент git

#### 1.1.1 Установка git

Установка в Linux и Unix

- Используйте обычный менеджер пакетов вашего дистрибутива. Откройте терминал и введите подходящие команды.
- Если у вас 21 или более ранняя версия Fedora, используйте `yum install git`.
- Для 22 и последующих версий Fedora вводите `dnf install git`.
- Для дистрибутивов, основанных на Debian, например, Ubuntu, используйте `apt-get`: `sudo apt-get install git`.

#### 1.1.2 Настройка git

1. Открываем терминал.
2. Необходимо выполнить следующие команды:

```
git config --global user.name "Your Name"
```

```
git config --global user.email "your_email@whatever.com"
```

3. Необходимо выполнить следующие команды:

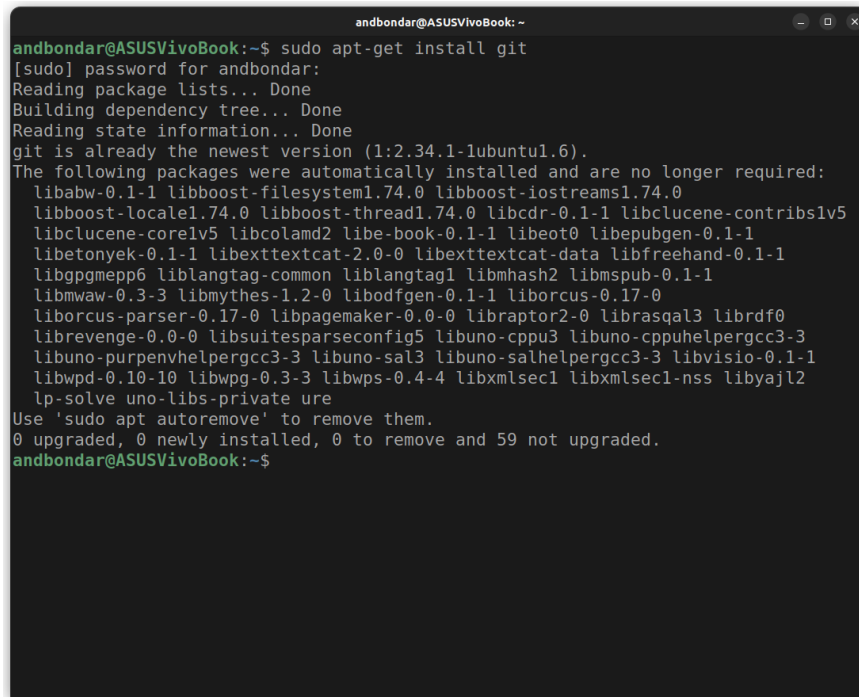
A terminal window titled 'andbondar@ASUSVivoBook: ~' showing the command 'sudo apt-get install git'. The output indicates that git is already the newest version (1:2.34.1-1ubuntu1.6). It then lists several packages that were automatically installed and are no longer required, including libabw-0.1-1, libboost-filesystem1.74.0, libboost-iostreams1.74.0, libboost-locale1.74.0, libboost-thread1.74.0, libcdr-0.1-1, libclucene-contribs1v5, libclucene-core1v5, libcolamd2, libe-book-0.1-1, libeot0, libepubgen-0.1-1, libetonyek-0.1-1, libexttextcat-2.0-0, libexttextcat-data, libfreehand-0.1-1, libgpgmepp6, liblangtag-common, liblangtag1, libmhash2, libmspub-0.1-1, libmwaw-0.3-3, libmythes-1.2-0, libodfgen-0.1-1, liborcus-0.17-0, liborcus-parser-0.17-0, libpagemaker-0.0-0, libraptor2-0, librasqal3, librdf0, librevenge-0.0-0, libsuitesparseconfig5, libuno-cppu3, libuno-cppuhelpergcc3-3, libuno-purpenvhelpergcc3-3, libuno-sal3, libuno-salhelpergcc3-3, libvisio-0.1-1, libwpd-0.10-10, libwpg-0.3-3, libwps-0.4-4, libxmlsec1, libxmlsec1-nss, libyajl2, and lp-solve. It also shows 'uno-lib-private' and 'ure'. The terminal concludes with '0 upgraded, 0 newly installed, 0 to remove and 59 not upgraded.' and the prompt 'andbondar@ASUSVivoBook:~\$'.

Рисунок 1.1. Установка git

```
git config --global core.autocrlf input
```

```
git config --global core.safecrlf warn
```

4. Необходимо выполнить следующие команды:

```
git config --global core.quotepath off
```

5. Необходимо выполнить следующие команды:

```
git config --global core.quotepath off
```

Для проверки верности всех введенных команд, введите:

```
git config --list
```

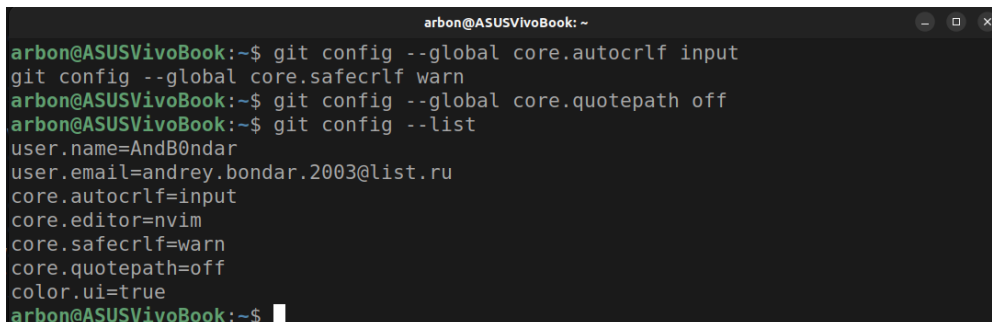
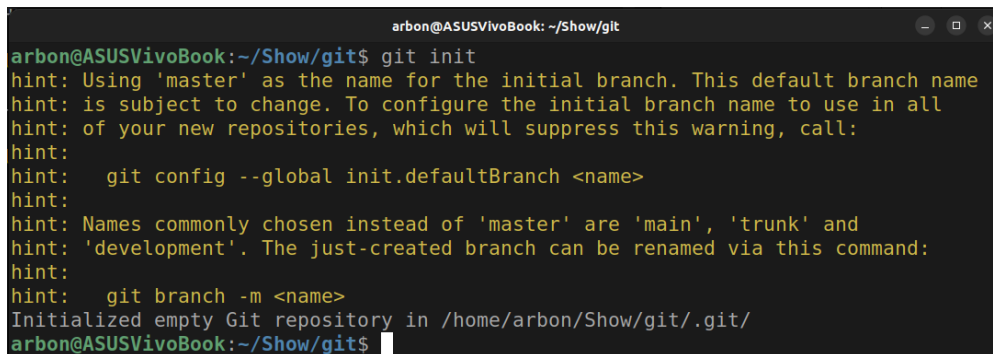
A terminal window titled 'arbon@ASUSVivoBook: ~' showing the command 'git config --list'. The output lists the following configuration: 'user.name=AndB0ndar', 'user.email=andrey.bondar.2003@list.ru', 'core.autocrlf=input', 'core.editor=nvim', 'core.safecrlf=warn', 'core.quotepath=off', and 'color.ui=true'. The terminal ends with the prompt 'arbon@ASUSVivoBook:~\$'.

Рисунок 1.2. Настройка git

## 1.2 Создарили локальный репозиторий и добавление в него несколько файлов

Выполняем команду `git init`. После выполнения данной команды, должно высветиться данное сообщение, показанное на рис. 1.3.



```
arbon@ASUSVivoBook: ~/Show/git
arbon@ASUSVivoBook:~/Show/git$ git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /home/arbon/Show/git/.git/
arbon@ASUSVivoBook:~/Show/git$
```

Рисунок 1.3. Создание репозитория git

## 1.3 Добавление файлов в репозиторий

Далее, командой `touch file` создадим несколько текстовый файлов. Чтобы добавить файл в репозиторий необходимо выполнить следующее:

1. Вводим команды:

```
git add <Название вашего файла>
```

```
git commit -m "Ваш текст для коммита"
```

2. Чтобы проверить состояние репозитория, выполним команду: `git status`. Команда проверки состояния сообщит, что коммитить нечего. Это означает, что в репозитории хранится текущее состояние рабочего каталога, и нет никаких изменений, ожидающих записи.

Результат этих действий показан на рисунке 1.4.

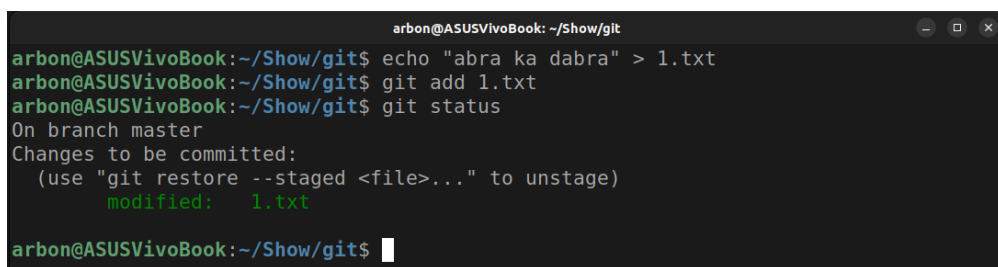


```
arbon@ASUSVivoBook: ~/Show/git
arbon@ASUSVivoBook:~/Show/git$ git add 1.txt 2.txt 3.txt
arbon@ASUSVivoBook:~/Show/git$ git commit -m "first commit"
[master (root-commit) 91e6554] first commit
3 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 1.txt
create mode 100644 2.txt
create mode 100644 3.txt
arbon@ASUSVivoBook:~/Show/git$
```

Рисунок 1.4. Добавление файлов в репозиторий и первый коммит

## 1.4 Изменение файла

Далее внесем изменения в один из файлов. Путем перенаправления вывода команды `echo` в файл. Добавим изменения в индекс командой: `git add <имя файла>`. Чтобы проверить внесение команды в индекс, `git` предоставляет команду `git status`. Все эти действия отображены на рисунке 1.5.



```
arbon@ASUSVivoBook: ~/Show/git
arbon@ASUSVivoBook:~/Show/git$ echo "abra ka dabra" > 1.txt
arbon@ASUSVivoBook:~/Show/git$ git add 1.txt
arbon@ASUSVivoBook:~/Show/git$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   1.txt
arbon@ASUSVivoBook:~/Show/git$
```

Рисунок 1.5. Индексирование изменений и их проверка

## 1.5 Коммит

Теперь, чтобы сохранить изменения в репозитории, необходимо сделать коммит. Для этого существует команда `git commit -m "текст коммита"` (см. рисунок 1.6). Флаг `-m` позволяет сразу добавить комментарий при вводе команды, без него `git`, для комментария, откроет текстовый редактор, установленный по умолчанию, обычно это `vim`.

```
arbon@ASUSVivoBook: ~/Show/git
arbon@ASUSVivoBook:~/Show/git$ git commit -m "Изменение файла 1.txt"
[master 4ae5fa5] Изменение файла 1.txt
1 file changed, 1 insertion(+)
arbon@ASUSVivoBook:~/Show/git$
```

Рисунок 1.6. Коммит

## 1.6 Как работает индексация

Выполним подряд следующие шаги:

1. Изменим еще один файл.
2. Добавим это изменение в индекс git.
3. Изменим файл еще раз.
4. Проверим состояние и произведем коммит проиндексированного изменения.
5. Добавим второе изменение в индекс, а затем проверим состояние с помощью команды `git status`.
6. Сделаем коммит второго изменения.

Результаты проделанных шагов проиллюстрированы на рисунках 1.7-1.8.

```
arbon@ASUSVivoBook: ~/Show/git
arbon@ASUSVivoBook:~/Show/git$ echo "kada bra bra" > 2.txt
arbon@ASUSVivoBook:~/Show/git$ git add 2.txt
arbon@ASUSVivoBook:~/Show/git$ echo "trulala la lala" >> 2.txt
arbon@ASUSVivoBook:~/Show/git$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   2.txt

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:   2.txt

arbon@ASUSVivoBook:~/Show/git$ git commit "Изменение 2.txt"
error: pathspec 'Изменение 2.txt' did not match any file(s) known to git
arbon@ASUSVivoBook:~/Show/git$ git commit -m "Изменение 2.txt"
[master e159314] Изменение 2.txt
1 file changed, 1 insertion(+)
arbon@ASUSVivoBook:~/Show/git$
```

Рисунок 1.7. Коммит первого проиндексированного изменения



```
arbon@ASUSVivoBook: ~/Show/git
arbon@ASUSVivoBook:~/Show/git$ git add 2.txt
arbon@ASUSVivoBook:~/Show/git$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   2.txt

arbon@ASUSVivoBook:~/Show/git$ git commit -m "Второе изменение 2.txt"
[master 1c00310] Второе изменение 2.txt
 1 file changed, 1 insertion(+)
arbon@ASUSVivoBook:~/Show/git$
```

Рисунок 1.8. Коммит второго проиндексированного изменения

## 1.7 История коммитов

Для просмотра истории коммитов используется команда: `git log` (Рисунок 1.9).

```
arbon@ASUSVivoBook: ~/Show/git
arbon@ASUSVivoBook:~/Show/git$ git log
commit 1c003103c13222dac1bb1bf9dc13f1aab34b27ab (HEAD -> master)
Author: AndB0ndar <andrey.bondar.2003 at list dot ru>
Date:   Sat Feb 18 20:14:25 2023 +0300

    Второе изменение 2.txt

commit e159314d7c64d7a3359dfc76f67fd161d9ab53f5
Author: AndB0ndar <andrey.bondar.2003 at list dot ru>
Date:   Sat Feb 18 20:09:43 2023 +0300

    Изменение 2.txt

commit 4ae5fa55df9588a30ac3140817476d8f3d6dcbc8
Author: AndB0ndar <andrey.bondar.2003 at list dot ru>
Date:   Sat Feb 18 19:51:29 2023 +0300

    Изменение файла 1.txt

commit 9bebabbb5ae1ed4f836c5a2ab75ac4fbal1eeb8d0
Author: AndB0ndar <andrey.bondar.2003 at list dot ru>
Date:   Sat Feb 18 19:33:02 2023 +0300

    add proekt.html

commit 91e6554eb129f13245b99ec09405c2e61c0d9e7f
Author: AndB0ndar <andrey.bondar.2003 at list dot ru>
Date:   Tue Feb 14 11:32:16 2023 +0300

    first commit
arbon@ASUSVivoBook:~/Show/git$
```

Рисунок 1.9. История коммитов

Для уточнения формата лога используется флаг `--pretty=format:"..."`.

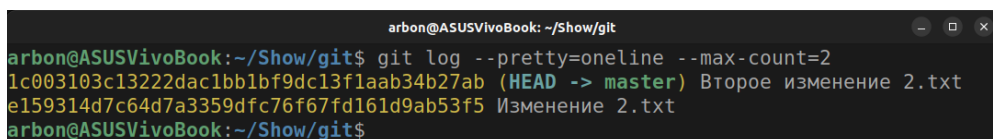
Для примера выполните команду:

```
git log --pretty=format:"%h %ad | %s%d [%an]" --graph --date=short
```

Рассмотрим её в деталях:

- %h — укороченный хэш коммита
- %d — дополнения коммита («ГОЛОВЫ» веток или теги)
- %ad — дата коммита
- %s — комментарий
- %an — имя автора
- --graph — отображает дерево коммитов в виде ASCII-графика
- --date=short — сохраняет формат даты коротким

Приведем пирмер одного из форматирования вывода истории (см. рис. 1.10).



```

arbon@ASUSVivoBook: ~/Show/git
arbon@ASUSVivoBook:~/Show/git$ git log --pretty=oneline --max-count=2
1c003103c13222dac1bb1bf9dc13f1aab34b27ab (HEAD -> master) Второе изменение 2.txt
e159314d7c64d7a3359dfc76f67fd161d9ab53f5 Изменение 2.txt
arbon@ASUSVivoBook:~/Show/git$

```

Рисунок 1.10. Форматирование истории коммитов

## 1.8 Отмена изменений

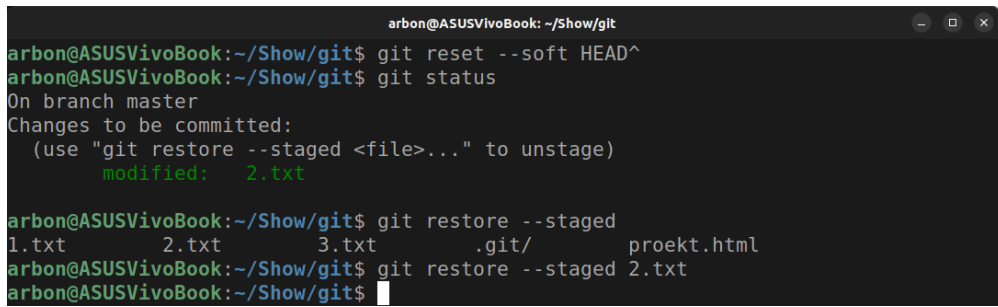
Для отмены изменений в репозитории существует команда `git reset`, которая переместит HEAD и текущую ветку обратно туда, куда вы укажете, отказавшись от любых коммитов, которые могут быть оставлены позади. Далее останется очистить индекс командой `git restore`. И так, чтобы вернуть директорию в предыдущее состояние введем слудующие команды:

```

git reser --soft HEAD~
git restore --staged 2.txt

```

Результат этих команд проиллюстрирован на рисунке 1.11.

A screenshot of a terminal window titled 'arbon@ASUSVivoBook: ~/Show/git'. The terminal shows the following commands and output:

```
arbon@ASUSVivoBook:~/Show/git$ git reset --soft HEAD^
arbon@ASUSVivoBook:~/Show/git$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   2.txt

arbon@ASUSVivoBook:~/Show/git$ git restore --staged
1.txt      2.txt      3.txt      .git/      projekt.html
arbon@ASUSVivoBook:~/Show/git$ git restore --staged 2.txt
arbon@ASUSVivoBook:~/Show/git$
```

Рисунок 1.11. Возвращение рабочего каталога к предыдущему состоянию

## 1.9 Создание тега

Git использует два основных типа тегов: легковесные и аннотированные.

Легковесный тег — это что-то очень похожее на ветку, которая не изменяется — просто указатель на определённый коммит.

А вот аннотированные теги хранятся в базе данных Git как полноценные объекты. Они имеют контрольную сумму, содержат имя автора, его e-mail и дату создания, имеют комментарий и могут быть подписаны и проверены с помощью GNU Privacy Guard (GPG). Обычно рекомендуется создавать аннотированные теги, чтобы иметь всю перечисленную информацию; но если вы хотите сделать временную метку или по какой-то причине не хотите сохранять остальную информацию, то для этого годятся и легковесные.

### 1.9.1 Аннотированные теги

Создание аннотированного тега в Git выполняется легко. Самый простой способ — это указать -a при выполнении команды tag:

```
git tag -a <имя тега> -m "my version 1.4"
```

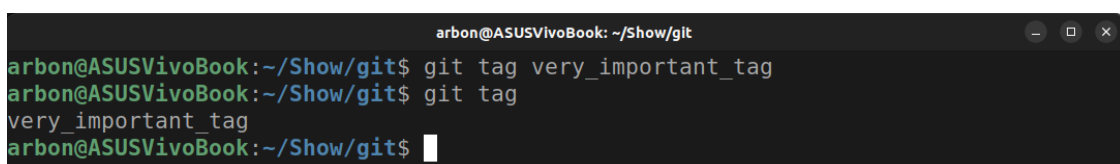
Опция -m задаёт сообщение, которое будет храниться вместе с тегом. Если не указать сообщение, то Git запустит редактор, чтобы вы смогли его ввести.

## 1.9.2 Легковесный теги

Легковесный тег — это ещё один способ пометить коммит. По сути, это контрольная сумма коммита, сохранённая в файл — больше никакой информации не хранится. Для создания легковесного тега не передавайте опций -a, -s и -m, укажите только название:

```
git tag <имя тега>
```

Создание легковесного тега показано на рисунке 1.12.

A screenshot of a terminal window with a dark background. The title bar reads 'arbron@ASUSVivoBook: ~/Show/git'. The terminal shows the following commands and output: 'arbron@ASUSVivoBook:~/Show/git\$ git tag very\_important\_tag', 'arbron@ASUSVivoBook:~/Show/git\$ git tag', 'very\_important\_tag', and 'arbron@ASUSVivoBook:~/Show/git\$' followed by a cursor.

```
arbron@ASUSVivoBook:~/Show/git$ git tag very_important_tag
arbron@ASUSVivoBook:~/Show/git$ git tag
very_important_tag
arbron@ASUSVivoBook:~/Show/git$
```

Рисунок 1.12. Создание тега

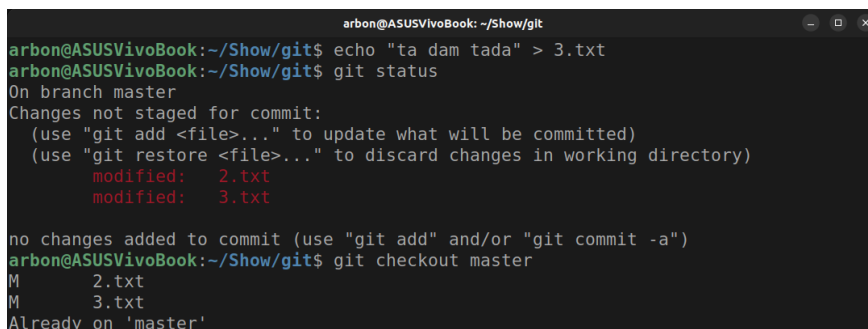
## 1.10 Отмена изменений в индексе

### 1.10.1 Отмена изменений (до индексации)

Для отмены изменений необходимо выполнить команду:

```
git checkout master
```

Результат выполнения этой команды показан на рисунке 1.13.

A screenshot of a terminal window with a dark background. The title bar reads 'arbron@ASUSVivoBook: ~/Show/git'. The terminal shows the following commands and output: 'arbron@ASUSVivoBook:~/Show/git\$ echo "ta dam tada" > 3.txt', 'arbron@ASUSVivoBook:~/Show/git\$ git status', 'On branch master', '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: 2.txt', 'modified: 3.txt', 'no changes added to commit (use "git add" and/or "git commit -a")', 'arbron@ASUSVivoBook:~/Show/git\$ git checkout master', 'M 2.txt', 'M 3.txt', and 'Already on 'master''.

```
arbron@ASUSVivoBook:~/Show/git$ echo "ta dam tada" > 3.txt
arbron@ASUSVivoBook:~/Show/git$ git status
On branch master
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:   2.txt
        modified:   3.txt
no changes added to commit (use "git add" and/or "git commit -a")
arbron@ASUSVivoBook:~/Show/git$ git checkout master
M      2.txt
M      3.txt
Already on 'master'
```

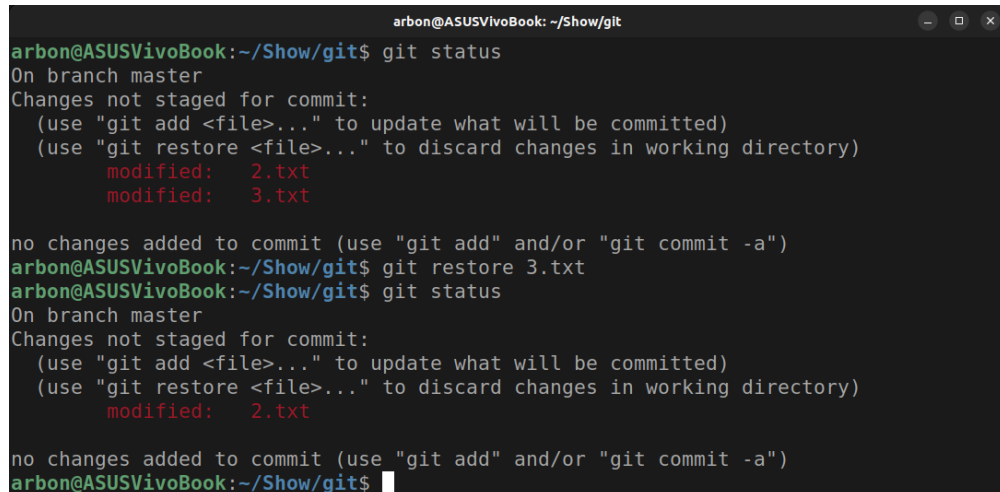
Рисунок 1.13. Отмена изменений (до индексации)

## 1.10.2 Отмена изменений (после индексации)

Для отмены изменений, уже проиндексированных, необходимо выполнить команду:

```
git reset HEAD имя_файла
```

Результат выполнения этой команды показан на рисунке 1.14.



```
arbron@ASUSVivoBook: ~/Show/git
arbron@ASUSVivoBook:~/Show/git$ git status
On branch master
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:   2.txt
        modified:   3.txt

no changes added to commit (use "git add" and/or "git commit -a")
arbron@ASUSVivoBook:~/Show/git$ git restore 3.txt
arbron@ASUSVivoBook:~/Show/git$ git status
On branch master
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:   2.txt

no changes added to commit (use "git add" and/or "git commit -a")
arbron@ASUSVivoBook:~/Show/git$
```

Рисунок 1.14. Отмена изменений (после индексации)

## 1.11 Отмена коммита

Для отмены изменений в репозитории существует команда: `git reset`, которая переместит HEAD и текущую ветку обратно туда, куда вы укажете, отказавшись от любых коммитов, которые могут быть оставлены позади. Данное действие уже показано на рисунке 1.11.