

## Лабораторная работа №2

### «Система контроля версий Git. Часть 2»

**Цель работы:** научиться использовать операции отмены, исключать файлы, которые нет необходимости вести в системе контроля версий, а также научиться работать с удаленными репозиториями.

#### Практическая часть

##### 1. Операции отмены

В любой момент вам может потребоваться что-либо отменить. Рассмотрим несколько основных способов отмены сделанных изменений. Будьте осторожны, не все операции отмены в свою очередь можно отменить! Это одна из редких областей Git, где неверными действиями можно необратимо удалить результаты своей работы.

##### *Отмена индексации файла*

**\$ git reset HEAD <file>** – откат к последнему коммиту (здесь HEAD–указатель на текущий коммит. ). То есть, если что-то пошло не так, то можно отменить все изменения и откатиться к состоянию после последнего коммита.

**\$ git reset –hard HEAD** - самая мощная и самая опасная команда – откат к указанному коммиту с потерей всех изменений!

##### *Отмена изменений в файле*

**\$ git checkout** – извлечение файлов из истории коммитов.

#### Задание:

1. Создайте в папке вашей бригады папку *LR2* и в ней папку *task1* (непосредственно из под Windows, не используя GitBash).
2. Создайте внутри папки *task1* два файла с расширением .txt и добавьте в них текст.
3. Перейдите в папку *task1*, используя команду **cd <путь>**. Инициализируйте репозиторий.
4. Добавьте данные файлы в versionный контроль. Убедитесь в этом.

```

Acer@Acer552G MINGW64 /d/IT_in_PA/22VVV1/1_SPE/LR7/task1
$ cd d://IT_in_PA/22VVV1/1_SPE/LR7/task1
Acer@Acer552G MINGW64 /d/IT_in_PA/22VVV1/1_SPE/LR7/task1
$ git init
Initialized empty Git repository in D:/IT_in_PA/22VVV1/1_SPE/LR7/task1/.git/
Acer@Acer552G MINGW64 /d/IT_in_PA/22VVV1/1_SPE/LR7/task1 (master)
$ git add --all
Acer@Acer552G MINGW64 /d/IT_in_PA/22VVV1/1_SPE/LR7/task1 (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   text1.txt
        new file:   text2.txt

```

5. Сделайте коммит.

```

Acer@Acer552G MINGW64 /d/IT_in_PA/22VVV1/1_SPE/LR7/task1 (master)
$ git commit -m "new files"
[master (root-commit) 1913d7f] new files
 2 files changed, 2 insertions(+)
 create mode 100644 text1.txt
 create mode 100644 text2.txt

```

6. Внесите изменения в один из файлов, а другой файл удалите. Выведите текущий статус.

```

Acer@Acer552G MINGW64 /d/IT_in_PA/22VVV1/1_SPE/LR7/task1 (master)
$ git status
On branch master

Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   text1.txt
        deleted:    text2.txt

no changes added to commit (use "git add" and/or "git commit -a")

```

7. Выполните команду `$ git add` для обоих файлов, чтобы окончательно проиндексировать изменения.

```

Acer@Acer552G MINGW64 /d/IT_in_PA/22VVV1/1_SPE/LR7/task1 (master)
$ git add --all
Acer@Acer552G MINGW64 /d/IT_in_PA/22VVV1/1_SPE/LR7/task1 (master)
$ git status
On branch master

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   text1.txt
        deleted:    text2.txt

```

8. Отмените индексирование файлов. Убедитесь, в том, что все вернулось в состояние до индексации, проверив текущий статус.

```

Acer@Acer552G MINGW64 /d/IT_in_PA/22VVV1/1_SPE/LR7/task1 (master)
$ git reset HEAD
Unstaged changes after reset:
M    text1.txt
D    text2.txt

Acer@Acer552G MINGW64 /d/IT_in_PA/22VVV1/1_SPE/LR7/task1 (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   text1.txt
        deleted:    text2.txt

no changes added to commit (use "git add" and/or "git commit -a")

```

9. Вернитесь в состояние до изменений, используя команду `$ git checkout`, и убедитесь, что в первом файле отменились изменения, а удалённый файл вернулся на место.

```

Acer@Acer552G MINGW64 /d/IT_in_PA/22VVV1/1_SPE/LR7/task1 (master)
$ git checkout text1.txt
Updated 1 path from the index

Acer@Acer552G MINGW64 /d/IT_in_PA/22VVV1/1_SPE/LR7/task1 (master)
$ git checkout text2.txt
Updated 1 path from the index

Acer@Acer552G MINGW64 /d/IT_in_PA/22VVV1/1_SPE/LR7/task1 (master)
$ git status
On branch master
nothing to commit, working tree clean

```

**Важно понимать**, что `git checkout <file>` — опасная команда. Все локальные изменения в файле пропадут. Git просто заменит его версией из последнего коммита. Ни в коем случае не используйте эту команду, если вы не уверены, что изменения в файле вам не нужны.

#### Самостоятельное задание:

1. Использование команды `$ git reset --hard HEAD`.

Снова внесите изменения в один из файлов, а другой удалите. Проиндексируйте изменения. Выведите текущий статус. Затем примените команду `git reset --hard HEAD` и проверьте полученный статус. Сделайте выводы.

2. Откат не на последний коммит.

Дважды внесите изменения в один файл и дважды зафиксируйте эти изменения с комментариями. Выведите лог. Выполните откат по номеру (несколько цифр, позволяющих идентифицировать его однозначно) предпоследнего коммита, используя команду `git reset --hard`. Проверьте файл, убедитесь, что всё вернулось к состоянию предпоследнего коммита.

*Примечание:*

Если лог не помещается на экране, то программа будет выводить его по частям и после первой части ждать нажатия любой клавиши. Вверх-вниз его можно проматывать стрелками на клавиатуре. Для выхода из этого режима следует нажать «q»+enter.

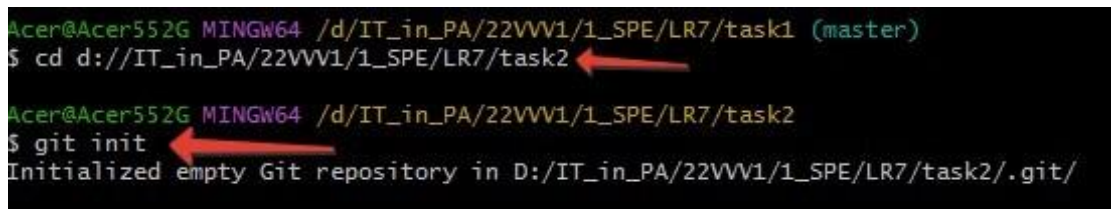
## 2. Игнорирование файлов

Зачастую, имеется группа файлов, которые не только нет необходимости автоматически добавлять в репозиторий, но и видеть в списках неотслеживаемых. К таким файлам обычно относятся автоматически генерируемые файлы (различные логи, результаты сборки программ и т.п.). В таком случае, необходимо создать файл .gitignore с перечислением шаблонов соответствующих таким файлам.

Файл .gitignore – это обычный текстовый файл, который содержит список всех указанных файлов и папок проекта, которые Git должен игнорировать и не отслеживать.

### Задание:

1. Создайте в папке вашей бригады папку LR2 и в ней папку task2 (непосредственно из под Windows, не используя GitBash). Инициализируйте репозиторий.



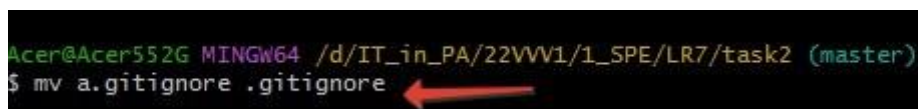
```
Acer@Acer552G MINGW64 /d/IT_in_PA/22VVV1/1_SPE/LR7/task1 (master)
$ cd d://IT_in_PA/22VVV1/1_SPE/LR7/task2
Acer@Acer552G MINGW64 /d/IT_in_PA/22VVV1/1_SPE/LR7/task2
$ git init
Initialized empty Git repository in D:/IT_in_PA/22VVV1/1_SPE/LR7/task2/.git/
```

2. Создайте в этом репозитории файл «.gitignore» для постоянного исключения из числа файлов с отслеживаемыми изменениями временные и автоматически создаваемые файлы компиляторов и компоновщиков.

| Имя        | Тип | Размер  |
|------------|-----|---------|
| [.]        |     | <Папка> |
| [.git]     |     | <Папка> |
| .gitignore |     | 0       |

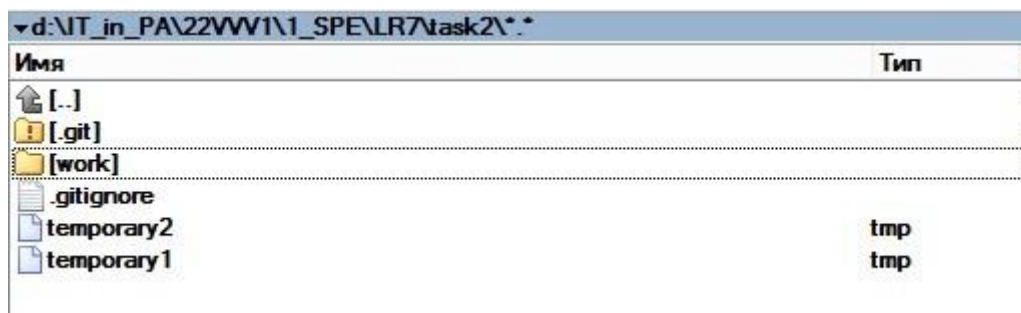
Если при создании файла с именем «.gitignore» средствами контекстного меню мыши произошел отказ системы Windows сохранить файл без указания имени перед точкой, то в этом случае сохраните его с произвольным именем, например, «a.gitignore», а затем в командной строке Git Bash переименуйте его командой mv, предварительно сделав папку с файлом текущей (например, mv .gitignore.txt .gitignore)

| Имя    | Тип       | Размер  |
|--------|-----------|---------|
| [.]    |           | <Папка> |
| [.git] |           | <Папка> |
| a      | gitignore | 0       |



```
Acer@Acer552G MINGW64 /d/IT_in_PA/22VVV1/1_SPE/LR7/task2 (master)
$ mv a.gitignore .gitignore
```

2. Создайте в своем репозитории папку work, в которую добавьте любые файлы лабораторных работ, и 2 файла с расширением \*.tmp.



3. Проверьте текущий статус.

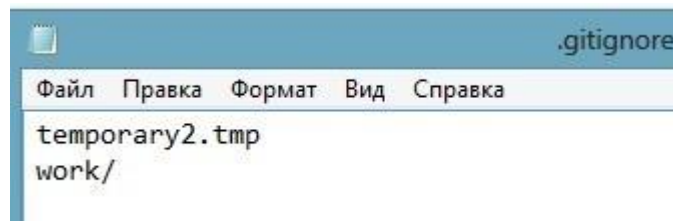
```
Acer@Acer552G MINGW64 /d/IT_in_PA/22VVV1/1_SPE/LR7/task2 (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore
        temporary1.tmp
        temporary2.tmp
        work/

nothing added to commit but untracked files present (use "git add" to track)
```

4. Создайте в файле «.gitignore» список файлов и папок, отслеживать которые не нужно, например



5. Добавьте под версию контроль. gitignore и сделайте коммит. Посмотрите статус и сделайте вывод.

### Самостоятельное задание:

1. Измените по своему усмотрению шаблоны игнорирования файлов и папок (измените список шаблонов в файле «.gitignore»).

Используйте следующие шаблоны:

- Если вы хотите игнорировать все файлы с определенным именем, то вам нужно написать точное имя файла, например, *temporary2.tmp*. В таком случае вам не нужно указывать полный путь к конкретному файлу. Этот шаблон будет игнорировать все файлы с таким именем, расположенные в любой папке проекта.

- Если вы хотите игнорировать файл, который расположен в корневом каталоге, то вы должны написать следующее: */temporary2.tmp*.

- Если вы хотите игнорировать файл, который расположен в папке *test* корневого каталога, вы должны сделать следующее: */test/ temporary2.tmp*.

- Если вы хотите игнорировать весь каталог со всем его содержимым, вам нужно указать имя каталога со слешем в конце: *test/*. Если вы напишете просто имя каталога без слеша, то этот шаблон будет соответствовать как любым файлам, так и любым каталогам с таким именем.

- Если вы хотите игнорировать все файлы и каталоги, имя которых начинается с *img*, для этого вам необходимо указать имя, а затем селектор подстановочного символа *\**: *img\**

- Если вы хотите игнорировать все файлы с определенным расширением, то вам необходимо будет использовать селектор подстановочного знака *\**, за которым последует расширение файла, например, *\*.tmp*. Этот шаблон будет соответствовать любому файлу с расширением *.tmp*, расположенному в любой папке проекта.

- Если вы хотите исключить конкретный файл с расширением *.tmp*, который в противном случае был бы проигнорирован, как и все остальные, то нужно будет воспользоваться шаблоном с отрицанием (с восклицательным знаком): *!temporary2.tmp*. В результате все файлы, оканчивающиеся на *.tmp* будут игнорироваться, кроме файла *temporary2.tmp*. Данный шаблон не будет работать, если вы игнорируете весь каталог

2. Сделайте коммит.

3. Продемонстрируйте результаты и сделайте выводы.

### 3. Удалённые репозитории

Git – это распределенная система, это значит, что у нас может быть много репозиториев. Они могут быть распределены – находиться на разных компьютерах внутри локальной сети, в разных местах одного компьютера или где-то далеко в интернете. Распределенность Git заключается в том, что он умеет определенным образом связывать эти репозитории между собой.

В некоторых СКВ система репозиториев клиент-серверная, но не в Git. Здесь все репозитории равноправны между собой. Репозитории делятся на 2 типа – локальные (у нас на компьютере) и удалённые (где-то ещё). Далее рассмотрим связывание локальных репозиториев, и будем рассматривать только ветвь *master*.

#### Задание:

1. Создайте в папке *LR2* новую папку *task3*, а внутри этой папки две папки *NewRep1* и *NewRep2*. Инициализируйте данные репозитории командой **git init**.

```

Acer@Acer552G MINGW64 /d/IT_in_PA/22VVV1/1_SPE/LR7/task3 (master)
$ cd d://IT_in_PA/22VVV1/1_SPE/LR7/task3/NewRep1|
Acer@Acer552G MINGW64 /d/IT_in_PA/22VVV1/1_SPE/LR7/task3/NewRep1 (master)
$ git init
Initialized empty Git repository in D:/IT_in_PA/22VVV1/1_SPE/LR7/task3/NewRep1/.git/
Acer@Acer552G MINGW64 /d/IT_in_PA/22VVV1/1_SPE/LR7/task3/NewRep1 (master)
$ cd d://IT_in_PA/22VVV1/1_SPE/LR7/task3/NewRep2
Acer@Acer552G MINGW64 /d/IT_in_PA/22VVV1/1_SPE/LR7/task3/NewRep2 (master)
$ git init
Initialized empty Git repository in D:/IT_in_PA/22VVV1/1_SPE/LR7/task3/NewRep2/.git/
Acer@Acer552G MINGW64 /d/IT_in_PA/22VVV1/1_SPE/LR7/task3/NewRep2 (master)
$

```

2. Добавьте в репозиторий *NewRep2* файл *test1.txt* с каким-либо текстом, проверьте статус, сделайте коммит.

```

Acer@Acer552G MINGW64 /d/IT_in_PA/22VVV1/1_SPE/LR7/task3/NewRep2 (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    test1.txt

nothing added to commit but untracked files present (use "git add" to track)
Acer@Acer552G MINGW64 /d/IT_in_PA/22VVV1/1_SPE/LR7/task3/NewRep2 (master)
$ git add test1.txt
Acer@Acer552G MINGW64 /d/IT_in_PA/22VVV1/1_SPE/LR7/task3/NewRep2 (master)
$ git commit -m "New file"
[master (root-commit) 163a713] New file
 1 file changed, 1 insertion(+)
 create mode 100644 test1.txt

```

3. Сделайте текущим первый репозиторий.

```

Acer@Acer552G MINGW64 /d/IT_in_PA/22VVV1/1_SPE/LR7/task3 (master)
$ cd d://IT_in_PA/22VVV1/1_SPE/LR7/task3/NewRep1

```

4. Добавьте к текущему репозиторию ссылку на удалённый (находящийся в другом месте) под каким-то именем (назовём его **rep2**), командой **git remote add**.

```

Acer@Acer552G MINGW64 /d/IT_in_PA/22VVV1/1_SPE/LR7/task3/NewRep1 (master)
$ git remote add rep2 "../NewRep2"

```

5. Просмотрите список прикрепленных к репозиторию удалённых репозиторий командой **git remote**. Ключ **-v** позволяет выводить не только название, но ещё и адрес удалённого репозитория. В скобках указаны допустимые команды скачивания (fetch) и отправки изменений (push).

```

Acer@Acer552G MINGW64 /d/IT_in_PA/22VVV1/1_SPE/LR7/task3/NewRep1 (master)
$ git remote
rep2
Acer@Acer552G MINGW64 /d/IT_in_PA/22VVV1/1_SPE/LR7/task3/NewRep1 (master)
$ git remote -v
rep2    ../NewRep2 (fetch)
rep2    ../NewRep2 (push)

```

6. Просмотрите удаленный репозиторий с помощью команды `git remote show`

<имя>

```
Acer@Acer552G MINGW64 /d/IT_in_PA/22VVV1/1_SPE/LR7/task3/NewRep1 (master)
$ git remote show rep2
* remote rep2
  Fetch URL: ../NewRep2
  Push URL: ../NewRep2
  HEAD branch: master
  Remote branch:
    master new (next fetch will store in remotes/rep2)
```

7. Выполните команду `git fetch rep2`. Эта команда возьмет информацию обо всех изменениях в удалённом репозитории и перенесёт в текущий.

Обратите внимание, что будет перенесена только информация об изменениях, а не они сами!!! `fetch` не меняет файлы!!!

```
Acer@Acer552G MINGW64 /d/IT_in_PA/22VVV1/1_SPE/LR7/task3/NewRep1 (master)
$ git fetch rep2
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 192 bytes | 0 bytes/s, done.
From ../NewRep2
* [new branch]      master      -> rep2/master
```

Выведите содержимое репозитория командой `ls`. Вы увидите, что он не изменился. В нем по-прежнему ничего нет. Однако, теперь мы знаем всё об удалённом репозитории. После того, как мы сделали `git fetch`, мы получили в том числе список веток удалённого репозитория. В данный момент у нас есть одна ветка – `master`. А в удаленном репозитории может быть много разных веток. Для того, чтобы работать с удаленными изменениями, нам нужно связать ветки между собой. В самом простом случае – связать две ветки `master` между собой.

8. Выполните команду `git checkout --track rep2/master`. Git должен создать связь текущей ветки данного репозитория с ветвью `master` репозитория `rep2`.

```
Acer@Acer552G MINGW64 /d/IT_in_PA/22VVV1/1_SPE/LR7/task3/NewRep1 (master)
$ git checkout --track rep2/master
Already on 'master'
branch 'master' set up to track 'rep2/master'.
```

В ответ Git пишет, что ветка `master` установлена для отслеживания удалённой ветки `master` из `rep2`.

9. Выполните команды `ls` и `git status`. Сделайте выводы

10. Получите изменения командой `git pull`.

- Если содержимое локального репозитория одинаково с удалённым, вывод будет таким:

```
$ git pull
Already up to date.
```

- Если были изменения, вывод будет примерно таким:

```
Acer@Acer552G MINGW64 /d/IT_in_PA/22VVV1/1_SPE/LR7/task3/NewRep1 (master)
$ git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 227 bytes | 1024 bytes/s, done.
From ../NewRep2
   163a713..4c6e424 master    -> rep2/master
Updating 163a713..4c6e424
Fast-forward
 test1.txt | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
```

10. Удалите ссылку на удаленный репозиторий **rep2**, используя команду **git remote rm rep2**.
11. Просмотрите список прикрепленных к репозиторию удалённых репозиториях. Сделайте выводы.

### Требования к отчету:

Титульный лист

Все введенные команды, ответы системы, пояснения, что было сделано между пунктами (например, отредактирован файл).