

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт перспективной инженерии
Департамент цифровых, робототехнических систем и электроники

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №2
дисциплины
«Основы кроссплатформенного программирования»
Вариант 4

Выполнил:
Борцов Богдан Михайлович
2 курс, группа ИТС-б-о-23-1,
11.03.02 «Инфокоммуникационные
технологии и системы связи»,
направленность (профиль)
«Инфокоммуникационные системы и
сети»,
очная форма обучения

(подпись)

Проверил:
Воронкин Р.А.

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2024 г.

Тема: Исследование возможностей Git для работы с локальными репозиториями

Цель: исследовать базовые возможности системы контроля версий Git для работы с локальными репозиториями.

Порядок выполнения работы:


1. Изучил теоретический материал.
2. Ознакомился с методическими указаниями и выполнил их.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk ().*


Owner * / Repository name *


 REPONCFU / JLAB_2

✔ JLAB_2 is available.

Great repository names are short and memorable. Need inspiration? How about [congenial-guide](#) ?

Description (optional)

☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**
You choose who can see and commit to this repository.

Initialize this repository with:

☒ **Add a README file**
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: **Python**

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: **MIT License**

Рис. 1 – Создание репозитория, используя GitHub:

```
C:\Users\Elony\Desktop\Portable\PortableGit>git clone https://github.com/REPONCFU/JLAB_2.git
Cloning into 'JLAB_2'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (5/5), done.
```

Рис.2. – Клонирование репозитория на локальную машину

```
C:\Users\Elony\Desktop\Portable\PortableGit>cd JLAB_2

C:\Users\Elony\Desktop\Portable\PortableGit\JLAB_2>git log
commit 23e7dd7db1f303d6b7ac36168f12a93d2320803f (HEAD -> main)
Author: REPONCFU <stavbortsov@mail.ru>
Date: Sat Dec 7 19:17:14 2024 +0300

    Начало положено

commit 6eb013811a5955fd2cf2d3917bd3164fafebcf77 (origin/main, origin/HEAD)
Author: REPONCFU <stavbortsov@mail.ru>
Date: Sat Dec 7 12:20:59 2024 +0300

    Initial commit
```

Рис.3 – Команда git log

```
C:\Users\Elony\Desktop\Portable\PortableGit\JLAB_2>git log -p -2
commit 23e7dd7db1f303d6b7ac36168f12a93d2320803f (HEAD -> main)
Author: REPONCFU <stavbortsov@mail.ru>
Date: Sat Dec 7 19:17:14 2024 +0300

    Начало положено

diff --git a/.idea/.gitignore b/.idea/.gitignore
new file mode 100644
index 0000000..26d3352
--- /dev/null
+++ b/.idea/.gitignore
@@ -0,0 +1,3 @@
+# Default ignored files
+/shelf/
+/workspace.xml
diff --git a/.idea/.name b/.idea/.name
new file mode 100644
index 0000000..42061c0
--- /dev/null
+++ b/.idea/.name
@@ -0,0 +1 @@
+README.md
\ No newline at end of file
diff --git a/.idea/JLAB_2.iml b/.idea/JLAB_2.iml
new file mode 100644
index 0000000..d8b3f6c
--- /dev/null
+++ b/.idea/JLAB_2.iml
@@ -0,0 +1,8 @@
... skipping ...
commit 23e7dd7db1f303d6b7ac36168f12a93d2320803f (HEAD -> main)
Author: REPONCFU <stavbortsov@mail.ru>
Date: Sat Dec 7 19:17:14 2024 +0300

    Начало положено

diff --git a/.idea/.gitignore b/.idea/.gitignore
new file mode 100644
index 0000000..26d3352
--- /dev/null
+++ b/.idea/.gitignore
@@ -0,0 +1,3 @@
+# Default ignored files
```

Рис. 4 – Команда git log -p -2

```

C:\Users\Elony\Desktop\Portable\PortableGit\JLAB_2>git log --stat
commit 23e7dd7db1f303d6b7ac36168f12a93d2320803f (HEAD -> main)
Author: REPONCFU <stavbortsov@mail.ru>
Date: Sat Dec 7 19:17:14 2024 +0300

Начало положено

.idea/.gitignore | 3 +++
.idea/.name | 1 +
.idea/JLAB_2.iml | 8 +++++++
.idea/inspectionProfiles/profiles_settings.xml | 6 +++++
.idea/misc.xml | 7 ++++++
.idea/modules.xml | 8 +++++++
.idea/vcs.xml | 6 ++++++
README.md | 8 ++++++-
main.py | 12 ++++++++
9 files changed, 58 insertions(+), 1 deletion(-)

commit 6eb013811a5955fd2cf2d3917bd3164fafebcf77 (origin/main, origin/HEAD)
Author: REPONCFU <stavbortsov@mail.ru>
Date: Sat Dec 7 12:20:59 2024 +0300

Initial commit

.gitignore | 162 +++++
LICENSE | 21 ++++++
README.md | 1 +
3 files changed, 184 insertions(+)

```

Рис. 5 – Команда git log --stat

```

C:\Users\Elony\Desktop\Portable\PortableGit\JLAB_2>git log --pretty=oneline
23e7dd7db1f303d6b7ac36168f12a93d2320803f (HEAD -> main) Начало положено
6eb013811a5955fd2cf2d3917bd3164fafebcf77 (origin/main, origin/HEAD) Initial commit

```

Рис. 6 – Команда git log --pretty=oneline

```

C:\Users\Elony\Desktop\Portable\PortableGit\JLAB_2>git log --pretty=format:"%h - %an, %ar : %s"
23e7dd7 - REPONCFU, 3 weeks ago : Начало положено
6eb0138 - REPONCFU, 3 weeks ago : Initial commit

```

Рис. 7 – Команда git log --pretty=format:"%h - %an, %ar : %s"

```

C:\Users\Elony\Desktop\Portable\PortableGit\JLAB_2>git log --pretty=format:"%h %s" --graph
* 23e7dd7 Начало положено
* 6eb0138 Initial commit

```

Рис. 8 – Команда git log --pretty=format:"%h %s" --graph

```

C:\Users\Elony\Desktop\Portable\PortableGit\JLAB_2>git log --since=3.weeks

C:\Users\Elony\Desktop\Portable\PortableGit\JLAB_2>git log --since=4.weeks
commit 23e7dd7db1f303d6b7ac36168f12a93d2320803f (HEAD -> main)
Author: REPONCFU <stavbortsov@mail.ru>
Date: Sat Dec 7 19:17:14 2024 +0300

Начало положено

commit 6eb013811a5955fd2cf2d3917bd3164fafebcf77 (origin/main, origin/HEAD)
Author: REPONCFU <stavbortsov@mail.ru>
Date: Sat Dec 7 12:20:59 2024 +0300

Initial commit

```

Рис. 9 – Команда git log --since=4.weeks


```
C:\Users\Elony\Desktop\Portable\PortableGit\JLAB_2>git remote -v
origin https://github.com/REPONCFU/JLAB_2.git (fetch)
origin https://github.com/REPONCFU/JLAB_2.git (push)
```

Рис. 13 – Команда git remote -v

```
C:\Users\Elony\Desktop\Portable\PortableGit\JLAB_2>git remote add pb https://github.com/paulboone/ticgit
C:\Users\Elony\Desktop\Portable\PortableGit\JLAB_2>git remote -v
origin https://github.com/REPONCFU/JLAB_2.git (fetch)
origin https://github.com/REPONCFU/JLAB_2.git (push)
pb      https://github.com/paulboone/ticgit (fetch)
pb      https://github.com/paulboone/ticgit (push)
```

Рис. 14 – Команда git remote add

```
C:\Users\Elony\Desktop\Portable\PortableGit\JLAB_2>git remote rename pb paul
C:\Users\Elony\Desktop\Portable\PortableGit\JLAB_2>git remote
origin
paul
```

Рис. 15 – Команда git remote rename

```
C:\Users\Elony\Desktop\Portable\PortableGit\JLAB_2>git tag -a v1.4 -m "my version 1.4"
C:\Users\Elony\Desktop\Portable\PortableGit\JLAB_2>git tag
v1.4
C:\Users\Elony\Desktop\Portable\PortableGit\JLAB_2>git show v1.4
tag v1.4
Tagger: REPONCFU <stavbortsov@mail.ru>
Date: Sun Dec 29 18:22:37 2024 +0300

my version 1.4

commit 23e7dd7db1f303d6b7ac36168f12a93d2320803f (HEAD -> main, tag: v1.4)
Author: REPONCFU <stavbortsov@mail.ru>
Date: Sat Dec 7 19:17:14 2024 +0300

Начало положено

diff --git a/.idea/.gitignore b/.idea/.gitignore
new file mode 100644
index 0000000..26d3352
```

Рис. 16 – Команды связанные с тегами

```

C:\Users\Elony\Desktop\Portable\PortableGit\JLAB_2>git checkout v1.4
A      c
Note: switching to 'v1.4'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

    git switch -c <new-branch-name>

Or undo this operation with:

    git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at 23e7dd7 Начало положено

```

Рис. 17 – Команда git checkout

```

C:\Users\Elony\Desktop\Portable\PortableGit\JLAB_2>git log --graph --pretty=oneline --abbrev-commit
* 8fb07f3 (HEAD, tag: v3, tag: 0.2, origin/main, origin/HEAD) rich для улучшенного пользовательского интерфейса
* 9cebce2 Добавлено сохранение результатов в файл
* 131bea5 Добавлен интерактивный режим с циклом
* 85da9d9 Добавлена поддержка нескольких алгоритмов хеширования
* 4f36a26 Добавлена базовая реализация хеширования с использованием SHA-256
* ef13ff0 (tag: v0.1) Инициализация репозитория: добавлен README.md
* 23e7dd7 (tag: v1.4, main) Начало положено
* 6eb0138 Initial commit

```

Рис. 18 – Команда git log

```

commit 8fb07f3d59bd89d174b95aeb3eb2dccf46c10081 (HEAD, tag: v3, tag: 0.2, origin/main, origin/HEAD)
Author: REPONCFU <stavbortsov@mail.ru>
Date: Sun Dec 29 19:02:32 2024 +0300

    rich для улучшенного пользовательского интерфейса

diff --git a/main.py b/main.py
index 3b9a873..6273df2 100644
--- a/main.py
+++ b/main.py
@@ -1,4 +1,9 @@
 import hashlib
+from rich.console import Console
+from rich.panel import Panel
+from rich.prompt import Prompt
+
+console = Console()

def hash_message(message, algorithm="sha256"):
@@ -14,9 +19,10 @@ def hash_message(message, algorithm="sha256"):
     if algorithm not in algorithms:
         raise ValueError(f"Неподдерживаемый алгоритм: {algorithm}")

-    hasher = algorithms[algorithm]()
-    hasher.update(message.encode('utf-8'))
-    return hasher.hexdigest()
+    with console.status("[bold green]Хеширование выполняется...", spinner="dots"):
+        hasher = algorithms[algorithm]()
+
+

```

Рис. 19 – Команда git show HEAD

```

commit 9cebce2e6c80baf2df453bf37574f02e2dca164b
Author: REPONCFU <stavbortsov@mail.ru>
Date: Sun Dec 29 18:56:36 2024 +0300

    Добавлено сохранение результатов в файл

diff --git a/main.py b/main.py
index da8b48a..3b9a873 100644
--- a/main.py
+++ b/main.py
@@ -19,17 +19,28 @@ def hash_message(message, algorithm="sha256"):
     return hasher.hexdigest()

+def save_to_file(filename, data):
+    """
+    Сохранение данных в файл.
+    """
+    with open(filename, "a") as file:
+        file.write(data + "\n")
+
+
+if __name__ == "__main__":
+    print("Добро пожаловать в хешер!")
+    print("Поддерживаемые алгоритмы: sha256, sha512, md5")
+    output_file = "hash_results.txt"

    while True:
        message = input("\nВведите сообщение для хеширования (или 'выход' для завершения): ")
:|

```

Рис. 20 – Команда git show HEAD~1

```

@@ -1,4 +1,9 @@
import hashlib
+from rich.console import Console
+from rich.panel import Panel
+from rich.prompt import Prompt
+
+console = Console()

def hash_message(message, algorithm="sha256"):
@@ -14,9 +19,10 @@ def hash_message(message, algorithm="sha256"):
    if algorithm not in algorithms:
        raise ValueError(f"Неподдерживаемый алгоритм: {algorithm}")

-    hasher = algorithms[algorithm]()
-    hasher.update(message.encode('utf-8'))
-    return hasher.hexdigest()
+    with console.status("[bold green]Хеширование выполняется...", spinner="dots"):
+        hasher = algorithms[algorithm]()
+        hasher.update(message.encode('utf-8'))
+        return hasher.hexdigest()

def save_to_file(filename, data):
@@ -28,19 +34,23 @@ def save_to_file(filename, data):

if __name__ == "__main__":
-    print("Добро пожаловать в хешер!")
:|

```

Рис. 21 – Команда git show 8fb07f3

После удаления содержимого файла `main.py` (шаг 10.1) и выполнения команды `git checkout - main.py` (шаг 10.2), файл успешно восстановился до последнего зафиксированного состояния. При повторном удалении содержимого и коммите (шаг 10.3) файл стал пустым, и это изменение было зафиксировано. После выполнения команды `git reset --hard HEAD~1` (шаг 10.4) репозиторий откатился к предыдущей версии, а файл `main.py` полностью восстановил свой исходный код. Эти действия подтвердили возможность восстановления отдельных файлов и отката репозитория к заданному состоянию.

Рис. 25 – Вывод по шагу 10

3. Ответил на контрольные вопросы.
4. Написал вывод на основе выполненной работы.
5. Сохранил отчет в PDF формате
6. Добавил отчет в папку DOC
7. Отправил локальные изменения в репозиторий GitHub.

Ответы на контрольные вопросы:

1. Как выполнить историю коммитов в Git? Какие существуют дополнительные опции для просмотра истории коммитов?

Команда `git log` позволяет просматривать историю коммитов. Дополнительные опции:

- `--graph` — вывод истории в виде графа.
- `--pretty=<format>` — настройка отображения (oneline, short, medium, full).
- `--abbrev-commit` — сокращение хэшей коммитов.
- `--since/--until` — фильтрация по дате.
- `--author` — фильтрация по автору.

2. Как ограничить вывод при просмотре истории коммитов?

Используются опции:

- `-n <число>` — вывод указанного количества последних коммитов (например, `git log -n 5`).
- `--grep=<строка>` — поиск коммитов с определенной строкой в сообщении.
- `--since/--until` — фильтрация по дате (например, `git log --since="2024-01-01"`).

3. Как внести изменения в уже сделанный коммит?

Для изменения последнего коммита:

1. Внести изменения в файлы.
2. Выполнить `git add <файл>`.
3. Использовать `git commit --amend`.

Это откроет редактор, где можно изменить сообщение коммита.

4. Как отменить индексацию файла в Git?

Используется команда:

```
git reset <файл>
```

Она удаляет файл из области индексации, оставляя его изменения в рабочей директории.

5. Как отменить изменения в файле?

Для отмены несохраненных изменений:

```
git checkout -- <файл>
```

Для отмены изменений, уже добавленных в индекс:

```
git reset <файл>
```

6. Что такое удаленный репозиторий Git?

Удаленный репозиторий — это хранилище, расположенное на сервере,

доступное через интернет или локальную сеть. Он используется для совместной работы над проектом.

7. Как выполнить просмотр удаленных репозиториях данного локального репозитория?

```
git remote -v
```

Она выводит список удаленных репозиториях с их URL-адресами.

8. Как добавить удаленный репозиторий для данного локального репозитория?

Команда:

```
git remote add <имя> <URL>
```

Например:

```
git remote add origin https://github.com/user/repo.git
```

9. Как выполнить отправку/получение изменений с удаленного репозитория?

Отправка изменений:

```
git push <удаленный_репозиторий> <ветка>
```

Получение изменений:

```
git pull <удаленный_репозиторий> <ветка>
```

10. Как выполнить просмотр удаленного репозитория?

Команда:

```
git remote show <имя>
```

Она выводит подробную информацию об удаленном репозитории.

11. Каково назначение тэгов Git?

Теги используются для маркировки определенных коммитов, например, для пометки версий релизов.

12. Как осуществляется работа с тэгами Git?

Создание тега:

```
git tag <имя_тега>
```

Создание аннотированного тега:

```
git tag -a <имя_тега> -m "Сообщение"
```

Отправка тегов на удаленный репозиторий:

```
git push origin <имя_тега>
```

13. Назначение флага --prune в командах git fetch и git push

Флаг **--prune** используется для удаления ссылок на удаленные ветки, которые больше не существуют на сервере:

- При **git fetch** очищает локальные ссылки на удаленные ветки.
- При **git push** удаляет ветки на удаленном сервере, которые были удалены локально.

Пример:

```
git push --prune git@example.com:/new-location.git
```

Эта команда синхронизирует удаленный репозиторий, удаляя ветки, которые отсутствуют в локальном репозитории.

Вывод:

В ходе лабораторной работы были освоены ключевые команды и операции Git. История коммитов успешно просматривалась с использованием `git log --graph --pretty=oneline`, что позволило визуализировать ветвления и изменения с сокращенными хэшами. После удаления содержимого файла `main.py` команда `git checkout -- main.py` восстановила его состояние, а откат через `git reset --hard HEAD~1` вернул репозиторий к предыдущей версии, подтвердив надежность инструментов восстановления. Аннотированные теги, созданные через `git tag -a v1.0 -m "Release 1.0"`, были успешно отправлены в удаленный репозиторий. Для зеркалирования использовалась команда `git fetch --prune`, которая очистила устаревшие ветки, а `git push --prune` синхронизировала локальные и удаленные ветки. Работа подтвердила, что Git обеспечивает контроль версий, эффективное восстановление данных и удобное управление удаленными репозиториями.