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.

```
C:\Users\Elony\Desktop\Portable\PortableGit\JLAB_3>git add 1.txt

C:\Users\Elony\Desktop\Portable\PortableGit\JLAB_3>git commit -m "add 1.txt file"
[main e328f8a] add 1.txt file
 1 file changed, 1 insertion(+)
 create mode 100644 1.txt
```

Рис. 1 – Выполнил пункт 4

```
C:\Users\Elony\Desktop\Portable\PortableGit\JLAB_3>git add 2.txt 3.txt

C:\Users\Elony\Desktop\Portable\PortableGit\JLAB_3>git commit --amend -m "add 2.txt and 3.txt"
[main 2d409ab] add 2.txt and 3.txt
 Date: Sun Dec 29 19:44:57 2024 +0300
 3 files changed, 3 insertions(+)
 create mode 100644 1.txt
 create mode 100644 2.txt
 create mode 100644 3.txt
```

Рис.2. – Выполнил пункты 5 и 6

```
C:\Users\Elony\Desktop\Portable\PortableGit\JLAB_3>git branch my_first_branch

C:\Users\Elony\Desktop\Portable\PortableGit\JLAB_3>git checkout my_first_branch
Switched to branch 'my_first_branch'

C:\Users\Elony\Desktop\Portable\PortableGit\JLAB_3>echo "File created in branch" > in_branch.txt

C:\Users\Elony\Desktop\Portable\PortableGit\JLAB_3>git add in_branch.txt

C:\Users\Elony\Desktop\Portable\PortableGit\JLAB_3>git commit -m "add in_branch.txt"
[my_first_branch a6bf381] add in_branch.txt
 1 file changed, 1 insertion(+)
 create mode 100644 in_branch.txt
```

Рис.3 – Выполнил пункты 7 и 8

```
C:\Users\Elony\Desktop\Portable\PortableGit\JLAB_3>git merge my_first_branch
Updating 2d409ab..a6bf381
Fast-forward
 in_branch.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 in_branch.txt

C:\Users\Elony\Desktop\Portable\PortableGit\JLAB_3>git merge new_branch
Updating a6bf381..0dda3e4
Fast-forward
 1.txt | 1 +
 1 file changed, 1 insertion(+)

C:\Users\Elony\Desktop\Portable\PortableGit\JLAB_3>git branch -d my_first_branch
Deleted branch my_first_branch (was a6bf381).

C:\Users\Elony\Desktop\Portable\PortableGit\JLAB_3>git branch -d new_branch
Deleted branch new_branch (was 0dda3e4).
```

Рис. 4 – Выполнил пункты 9 - 13

```
C:\Users\Elony\Desktop\Portable\PortableGit\JLAB_3>git branch branch_1

C:\Users\Elony\Desktop\Portable\PortableGit\JLAB_3>git branch branch_2

C:\Users\Elony\Desktop\Portable\PortableGit\JLAB_3>git checkout branch_1
Switched to branch 'branch_1'

C:\Users\Elony\Desktop\Portable\PortableGit\JLAB_3>echo "fix in the 1.txt" > 1.txt

C:\Users\Elony\Desktop\Portable\PortableGit\JLAB_3>echo "fix in the 3.txt" > 3.txt

C:\Users\Elony\Desktop\Portable\PortableGit\JLAB_3>git add 1.txt 3.txt

C:\Users\Elony\Desktop\Portable\PortableGit\JLAB_3>git commit -m "fix in 1.txt and 3.txt on branch_1"
[branch_1 23473c5] fix in 1.txt and 3.txt on branch_1
 2 files changed, 2 insertions(+), 3 deletions(-)

C:\Users\Elony\Desktop\Portable\PortableGit\JLAB_3>git checkout branch_2
Switched to branch 'branch_2'

C:\Users\Elony\Desktop\Portable\PortableGit\JLAB_3>echo "My fix in the 1.txt" > 1.txt

C:\Users\Elony\Desktop\Portable\PortableGit\JLAB_3>echo "My fix in the 3.txt" > 3.txt

C:\Users\Elony\Desktop\Portable\PortableGit\JLAB_3>git add 1.txt 3.txt

C:\Users\Elony\Desktop\Portable\PortableGit\JLAB_3>git commit -m "My fix in 1.txt and 3.txt on branch_2"
[branch_2 3859ee0] My fix in 1.txt and 3.txt on branch_2
 2 files changed, 2 insertions(+), 3 deletions(-)

C:\Users\Elony\Desktop\Portable\PortableGit\JLAB_3>git checkout branch_1
Switched to branch 'branch_1'

C:\Users\Elony\Desktop\Portable\PortableGit\JLAB_3>git merge branch_2
Auto-merging 1.txt
CONFLICT (content): Merge conflict in 1.txt
Auto-merging 3.txt
CONFLICT (content): Merge conflict in 3.txt
Automatic merge failed; fix conflicts and then commit the result.
```

Рис. 5 – Выполнил пункты 9 - 17

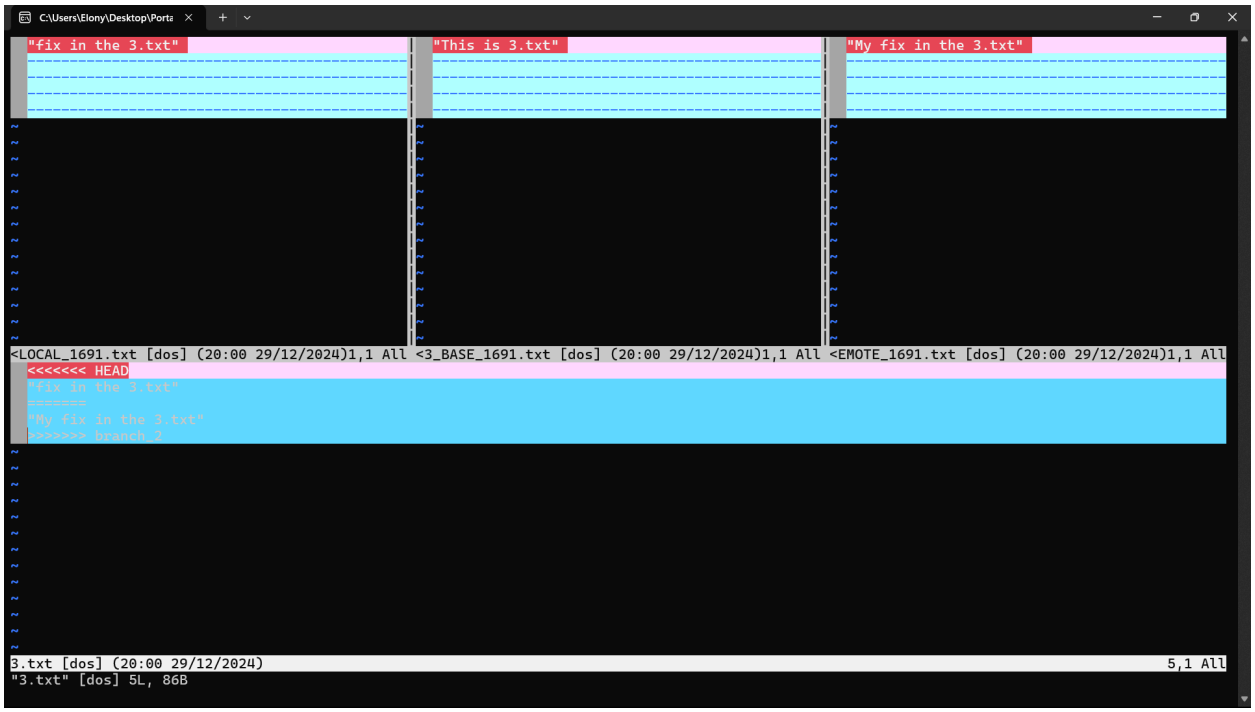

Рис. 6 – Решаю конфликт 3.txt используя команду git mergetool

```
C:\Users\Elony\Desktop\Portable\PortableGit\JLAB_3>git branch branch_3

C:\Users\Elony\Desktop\Portable\PortableGit\JLAB_3>git checkout branch_3
Switched to branch 'branch_3'

C:\Users\Elony\Desktop\Portable\PortableGit\JLAB_3>echo "the final fantasy in the 4.txt file" >> 2.txt

C:\Users\Elony\Desktop\Portable\PortableGit\JLAB_3>git add 2.txt

C:\Users\Elony\Desktop\Portable\PortableGit\JLAB_3>git commit -m "add final fantasy row to 2.txt"
[branch_3 8ae15aa] add final fantasy row to 2.txt
 1 file changed, 1 insertion(+)

C:\Users\Elony\Desktop\Portable\PortableGit\JLAB_3>git push -u origin branch_3
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 324 bytes | 324.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'branch_3' on GitHub by visiting:
remote:      https://github.com/REPONCFU/JLAB_3/pull/new/branch_3
remote:
To https://github.com/REPONCFU/JLAB_3.git
 * [new branch]      branch_3 -> branch_3
branch 'branch_3' set up to track 'origin/branch_3'.

C:\Users\Elony\Desktop\Portable\PortableGit\JLAB_3>git branch -f main branch_2

C:\Users\Elony\Desktop\Portable\PortableGit\JLAB_3>git push origin main
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/REPONCFU/JLAB_3.git
   1f0c038..3859ee0  main -> main

C:\Users\Elony\Desktop\Portable\PortableGit\JLAB_3>git push origin branch_2
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'branch_2' on GitHub by visiting:
remote:      https://github.com/REPONCFU/JLAB_3/pull/new/branch_2
remote:
To https://github.com/REPONCFU/JLAB_3.git
 * [new branch]      branch_2 -> branch_2
```

Рис. 7 – Выполнил пункты 18 - 24

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. Что такое ветка?

Ветка в Git — это указатель на конкретный коммит, который позволяет изолировать изменения. Используется для разработки новых функций, исправления ошибок и экспериментов без влияния на основную ветку.

2. Что такое HEAD?

HEAD — это указатель на текущую ветку или коммит, который используется в данный момент. Обычно указывает на последний коммит текущей ветки.

3. Способы создания веток:

Локальная ветка:
git branch <имя_ветки>

Создание и переключение:
git checkout -b <имя_ветки>

Удаленная ветка (через push):
git push origin <имя_ветки>

4. Как узнать текущую ветку?

Используется команда:

git branch

Текущая ветка отмечена символом *.

5. Как переключаться между ветками?

Команда для переключения:

git checkout <имя_ветки>

Или для создания и одновременного переключения:

git checkout -b <имя_ветки>

6. Что такое удаленная ветка?

Удаленная ветка — это ветка, которая хранится на сервере (например, GitHub, GitLab) и используется для совместной работы. Такие ветки имеют префикс remotes/origin/.

7. Что такое ветка отслеживания?

Ветка отслеживания — это локальная ветка, связанная с удалённой веткой. Она автоматически синхронизируется с удалённой при выполнении команд git pull и git push.

8. Как создать ветку отслеживания?

Для создания отслеживающей ветки:

git checkout --track origin/<имя_ветки>

Или при отправке локальной ветки на удалённый репозиторий:

git push -u origin <имя_ветки>

9. Как отправить изменения из локальной ветки в удаленную ветку?

Команда для отправки:

git push origin <имя_ветки>

Если локальная ветка уже отслеживает удалённую, достаточно:

git push

10. В чем отличие команд git fetch и git pull?

- **git fetch**: Загружает изменения из удалённого репозитория, но не сливает их с текущей веткой.
- **git pull**: Выполняет git fetch, а затем автоматически сливает изменения с текущей веткой.

11. Как удалить локальную и удаленную ветки?

Удалить локальную ветку:
git branch -d <имя_ветки>

- (Принудительное удаление: git branch -D <имя_ветки>)

Удалить удаленную ветку:
git push origin --delete <имя_ветки>

12. Git-flow: основные типы веток и работа с ними

Модель **git-flow** подразумевает несколько типов веток:

- **master/main**: Содержит стабильные релизы.
- **develop**: Основная ветка для разработки.
- **feature**: Создаётся для разработки новой функциональности.
- **release**: Подготовка релиза (тестирование, исправление ошибок).
- **hotfix**: Для экстренных исправлений в стабильной версии.

Работа с ветками:

- Новые функции разрабатываются в ветках feature/*, которые создаются из develop.
- После завершения работы слияние происходит обратно в develop.

- Готовый релиз переходит из release/* в master и develop.

Недостатки:

- Подходит не для всех проектов: избыточен для маленьких или часто изменяемых репозиториев.
- Требует дополнительного контроля версий и синхронизации между разработчиками.

13. Инструменты работы с ветками в GUI (пример: Sourcetree)

Sourcetree предоставляет следующие функции:

- Визуальное отображение всех веток, их слияний и различий.
- Удобное переключение между ветками.
- Создание новых локальных и удалённых веток.
- Удаление веток с локального и удалённого репозиториев.
- Автоматическое разрешение конфликтов с помощью встроенных инструментов.

Эти инструменты упрощают управление ветками и интеграцию с удалёнными репозиториями для пользователей, предпочитающих GUI.

Вывод:

В ходе лабораторной работы была изучена работа с ветками в Git, включая создание локальных и удалённых веток, настройку веток отслеживания и их синхронизацию. Были освоены команды для переключения между ветками, их слияния и разрешения конфликтов как вручную, так и с помощью утилиты git mergetool. Изучена разница между командами git fetch и git pull, что показало преимущества контроля при использовании git fetch. Работа с моделью ветвления **git-flow** продемонстрировала её преимущества для крупных проектов, а также выявила её избыточность для небольших задач.