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 №3
дисциплины
«Основы кроссплатформенного программирования»
Вариант 6

Выполнил:
Сопов Максим Игоревич
2 курс, группа ИТС-б-о-23-1,
11.03.02
«Инфокоммуникационные технологии и системы связи»,
очная форма обучения

_____
(подпись)

Проверил:
Доцент департамента цифровых, робототехнических систем и электроники
Воронкин Р.А.

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2024 г.

Лабораторная работа 1.3 «Основы ветвления Git»

Цель работы: исследование базовых возможностей по работе с локальными и удаленными ветками Git.

Ссылка на GitHub: https://github.com/MaxITS-kurwa/labar3

Порядок выполнения работы:

1. Создан репозиторий на GitHub, три файла (1.txt, 2.txt, 3.txt)

2. Проинициализирован первый файл и сделан коммит с комментарием "добавлен 1.txt файл"

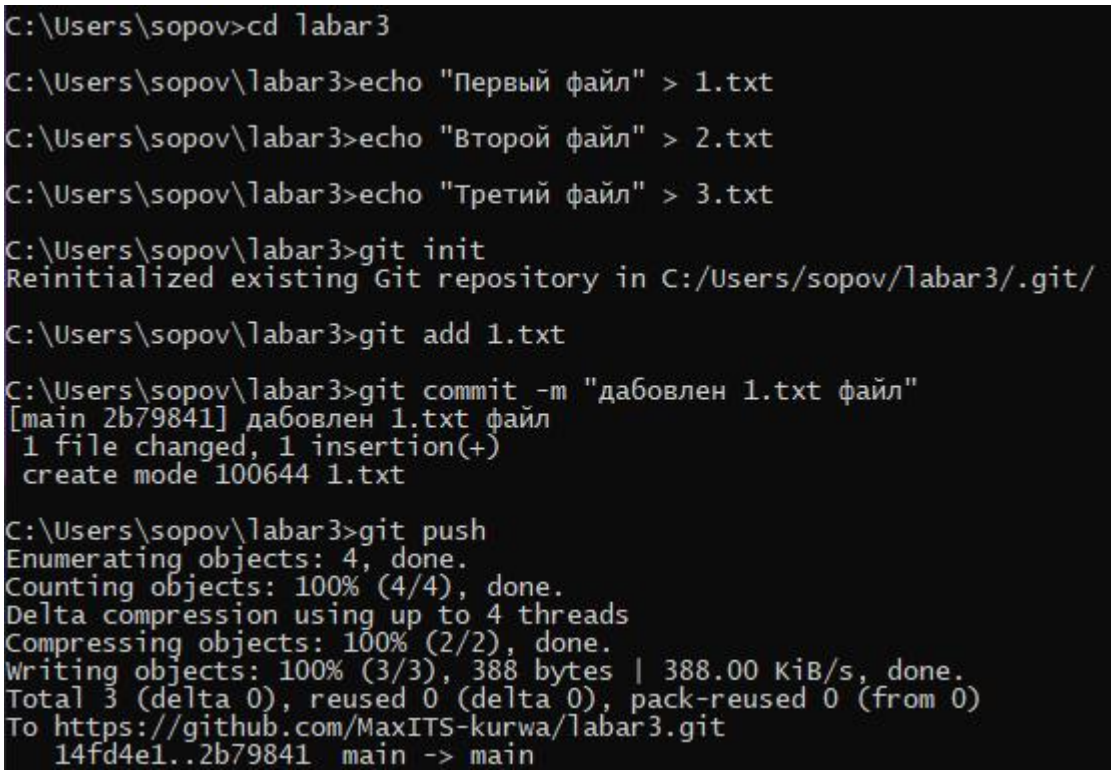

Рис. 1. Выполнение пункта 2, 3

3. Проинициализированы второй и третий файлы

4. Перезаписать уже сделанный коммит с новыми комментариями

```
C:\Users\sopov\labar3>git add 2.txt 3.txt

C:\Users\sopov\labar3>git commit -m "добавлен 2.txt и 3.txt"
[main bc6ed45] добавлен 2.txt и 3.txt
 2 files changed, 2 insertions(+)
 create mode 100644 2.txt
 create mode 100644 3.txt

C:\Users\sopov\labar3>git commit --amend -m "добавлен 2.txt и 3.txt"
[main 2b4d2b1] добавлен 2.txt и 3.txt
 Date: Thu Dec 26 20:25:02 2024 +0300
 2 files changed, 2 insertions(+)
 create mode 100644 2.txt
 create mode 100644 3.txt

C:\Users\sopov\labar3>git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 468 bytes | 156.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/MaxITS-kurwa/labar3.git
   2b79841..2b4d2b1  main -> main
```

Рис. 2. Выполнены пункты 4, 5

5. Создать новую ветку my_first_branch

6. Перейти на ветку и создать новый файл in_branch.txt

```
C:\Users\sopov\labar3>git branch my_first_branch

C:\Users\sopov\labar3>git checkout my_first_branch
Switched to branch 'my_first_branch'

C:\Users\sopov\labar3>echo "branch_file" > in_branch.txt

C:\Users\sopov\labar3>git add in_branch.txt

C:\Users\sopov\labar3>git commit -m "добавлен in_branch.txt"
[my_first_branch 85db718] добавлен in_branch.txt
 1 file changed, 1 insertion(+)
 create mode 100644 in_branch.txt
```

Рис. 3. Выполнены пункты 6, 7

7. Вернуться на ветку master

8. Создать и сразу перейти на ветку new_branch

9. Сделать изменения в файле 1.txt

```
C:\Users\sopov\labar3>git checkout master
error: pathspec 'master' did not match any file(s) known to git

C:\Users\sopov\labar3>git checkout -b new.branch
Switched to a new branch 'new.branch'

C:\Users\sopov\labar3>echo "новая строчка в 1.txt файле" >> 1.txt

C:\Users\sopov\labar3>git add 1.txt

C:\Users\sopov\labar3>git commit -m "добавлена новая строчка в 1.txt"
[new.branch db96cea] добавлена новая строчка в 1.txt
 1 file changed, 1 insertion(+)

C:\Users\sopov\labar3>git push
fatal: The current branch new.branch has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin new.branch

To have this happen automatically for branches without a tracking
upstream, see 'push.autoSetupRemote' in 'git help config'.
```

Рис. 4. Выполнены пункты 8, 9, 10

10. Слить ветки master и my_first_branch в new_branch

11. Удалить ветку my_first_branch

```
C:\Users\sopov\labar3>git checkout new.branch
Already on 'new.branch'

C:\Users\sopov\labar3>git merge master
merge: master - not something we can merge

C:\Users\sopov\labar3>git merge my_first_branch
Already up to date.

C:\Users\sopov\labar3>git branch -d my_first_branch
Deleted branch my_first_branch (was 85db718).
```

Рис. 5. Выполнены пункты 10, 11

12. Создать ветки branch_1 и branch_2
13. Изменить файлы 1.txt и 3.txt в ветке branch_1

```
C:\Users\sopov\labar3>git branch branch_1

C:\Users\sopov\labar3>git branch branch_2

C:\Users\sopov\labar3>git checkout branch_1
Switched to branch 'branch_1'

C:\Users\sopov\labar3>echo "исправления в 1.txt" > 1.txt

C:\Users\sopov\labar3>echo "исправления в 3.txt" > 3.txt

C:\Users\sopov\labar3>git add 1.txt 3.txt

C:\Users\sopov\labar3>git commit -m "исправления в 1.txt и 3.txt"
[branch_1 f64dc30] исправления в 1.txt и 3.txt
 2 files changed, 2 insertions(+), 3 deletions(-)
```

Рис. 6. Выполнены пункты 12, 13

14.Повторить изменения для branch_2

```
C:\Users\c\third-rep>git checkout brunch_2
error: pathspec 'brunch_2' did not match any file(s) known to git

C:\Users\c\third-rep>echo "my fix in the 1.txt" > 1.txt

C:\Users\c\third-rep>echo "my fix in the 3.txt" > 3.txt

C:\Users\c\third-rep>git add 1.txt 3.txt

C:\Users\c\third-rep>git commit -m "my fix in 1.txt and 3.txt"
[new.branch f124169] my fix in 1.txt and 3.txt
 2 files changed, 2 insertions(+), 2 deletions(-)
```

Рис. 7. Выполнен пункт 14

15. Слить ветку branch_2 в branch_1
16. Отправить ветку branch_1 на GitHub

```
C:\Users\sopov\labar3>git checkout branch_1
Switched to branch 'branch_1'

C:\Users\sopov\labar3>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.

C:\Users\sopov\labar3>git push
fatal: The current branch branch_1 has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin branch_1

To have this happen automatically for branches without a tracking
upstream, see 'push.autoSetupRemote' in 'git help config'.


C:\Users\sopov\labar3>git remote add origin https://github.com/MaxITS-kurwa/labar3.git
error: remote origin already exists.

C:\Users\sopov\labar3>git push origin branch_1
Enumerating objects: 13, done.
Counting objects: 100% (13/13), done.
Delta compression using up to 4 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (10/10), 1.06 KiB | 542.00 KiB/s, done.
Total 10 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (3/3), completed with 1 local object.
remote:
remote: Create a pull request for 'branch_1' on GitHub by visiting:
remote:      https://github.com/MaxITS-kurwa/labar3/pull/new/branch_1
remote:
To https://github.com/MaxITS-kurwa/labar3.git
 * [new branch]      branch_1 -> branch_1
```

Рис. 8. Выполнены пункты 15, 16

17. Создать ветку branch_3
18. Добавить файл 4.txt

```
C:\Users\sopov\labar3>git checkout -b branch_3
Switched to a new branch 'branch_3'

C:\Users\sopov\labar3>echo "крайний 4.txt файл" > 4.txt

C:\Users\sopov\labar3>git add 4.txt
C:\Users\sopov\labar3>git commit -m "добавлен 4.txt"
[branch_3 916af95] добавлен 4.txt
 3 files changed, 9 insertions(+)
 create mode 100644 4.txt
```

Рис. 9. Выполнены пункты 17, 18

19. Выполнить перемещение ветки master на ветку branch_2

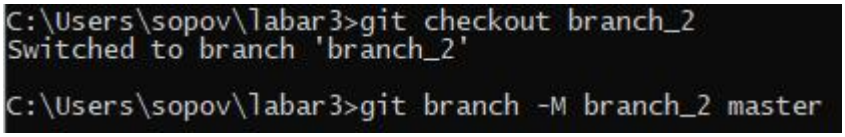

Рис. 10. Выполнен пункт 19

Контрольные вопросы:

1. Что такое ветка?

Ветка (branch) — это независимая линия разработки в репозитории Git. Она позволяет вносить изменения в код, не затрагивая основную ветку (обычно main или master). Это удобно для разработки новых функций, исправления ошибок или экспериментов.

2. Что такое HEAD?

HEAD — это указатель на текущую ветку или коммит, с которым вы работаете в данный момент. Обычно HEAD указывает на последний коммит текущей ветки.

3. Способы создания веток

- **Через командную строку:**
- git branch имя_ветки
- **Создание и переключение:**
- git checkout -b имя_ветки
- **С использованием GUI-инструментов** (например, GitKraken, SourceTree, Visual Studio Code).

4. Как узнать текущую ветку?

- В командной строке:
- git branch

Текущая ветка будет выделена звездочкой *.

- Через GUI-инструменты (например, название текущей ветки отображается в интерфейсе).

5. Как переключаться между ветками?

- Переключение на существующую ветку:

- git checkout имя_ветки
- Начиная с Git 2.23:
- git switch имя_ветки

6. Что такое удаленная ветка?

Удалённая ветка — это ветка, которая хранится на удалённом репозитории (например, на GitHub). Обычно её используют для совместной работы.

7. Что такое ветка отслеживания?

Ветка отслеживания (tracking branch) — это локальная ветка, связанная с удалённой веткой. Любые изменения, отправленные из локальной ветки, будут отражаться в удалённой ветке.

8. Как создать ветку отслеживания?

1. При клонировании удалённой ветки:
2. git checkout -b локальная_ветка origin/удалённая_ветка
3. При создании отслеживания для существующей ветки:
4. git branch --set-upstream-to=origin/удалённая_ветка локальная_ветка

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

1. Свяжите ветку с удалённой:
2. git push -u origin имя_ветки
3. Отправьте изменения:
4. git push

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

- **git fetch**: Загружает изменения из удалённого репозитория, но не сливает их с локальными изменениями.
- **git pull**: Загружает изменения и сразу сливает их с текущей локальной веткой.

11. Как удалить локальную и удаленную ветки?

- Удалить локальную ветку:
- git branch -d имя_ветки
- Удалить удалённую ветку:

- git push origin --delete имя_ветки

12. Изучить модель ветвления git-flow

Основные типы веток в git-flow:

- **Main** — основная ветка для стабильных релизов.
- **Develop** — основная ветка для разработки.
- **Feature** — ветки для новых функций.
- **Release** — ветки для подготовки релиза.
- **Hotfix** — ветки для исправления ошибок в продакшене.

Организация работы:

1. Новая функциональность разрабатывается в ветке feature.
2. После завершения работы сливается в develop.
3. Подготовка к релизу ведётся в ветке release.
4. Для срочных исправлений используются ветки hotfix, которые сливаются в main и develop.

Недостатки git-flow:

- Сложность в мелких проектах.
- Трудности с интеграцией в CI/CD.
- Увеличение времени на управление ветками.

Вывод: в ходе выполнения данной работы были изучены базовые и расширенные возможности работы с ветками в системе контроля версий Git. Были разобраны следующие аспекты: создание веток и их основные преимущества, такие как изоляция изменений и удобство в командной разработке. Понятие HEAD как указателя на текущий коммит или ветку. Методы создания и переключения между ветками, включая использование команд git branch, git checkout и git switch. Работа с удалёнными ветками и настройка веток отслеживания, что позволяет синхронизировать локальные и удалённые изменения. Различия между командами git fetch и git pull, что важно для корректного взаимодействия с удалёнными репозиториями. Удаление локальных и удалённых веток, а также слияние изменений**.**