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»

Выполнил:
Пугачев Кирилл Дмитриевич
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:
Доцент департамента цифровых, робототехнических систем и электроники
Воронкин Р.А.
_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2024 г.

ТЕМА: ОСНОВЫ ВЕТВЛЕНИЯ GIT

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

Ссылка на репозиторий: https://github.com/chillkirill/LABA3.git

Ход работы:

1. Создан общедоступный репозиторий на GitHub
2. Созданы три файла: 1.txt, 2.txt, 3.txt
3. Проинициализирован первый файл и сделан коммит с комментарием "add 1.txt file"

```
C:\Users\user>cd LABA3

C:\Users\user\LABA3>echo "первый файл" > 1.txt

C:\Users\user\LABA3>echo "второй файл" > 1.txt

C:\Users\user\LABA3>echo "третий файл" > 3.txt

C:\Users\user\LABA3>echo "второй файл" > 2.txt

C:\Users\user\LABA3>git init
Reinitialized existing Git repository in C:/Users/user/LABA3/.git/

C:\Users\user\LABA3>git add 1.txt

C:\Users\user\LABA3>git commit -m "add 1.txt file"
[main d58e114] add 1.txt file
 1 file changed, 1 insertion(+)
 create mode 100644 1.txt

C:\Users\user\LABA3>git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 353 bytes | 353.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/chillkirill/LABA3.git
   2525cb0..d58e114  main -> main
```

Рисунок 1. Выполнение пункта 2, 3

4. Проинициализированы второй и третий файлы
5. Перезаписать уже сделанный коммит с новыми комментариями

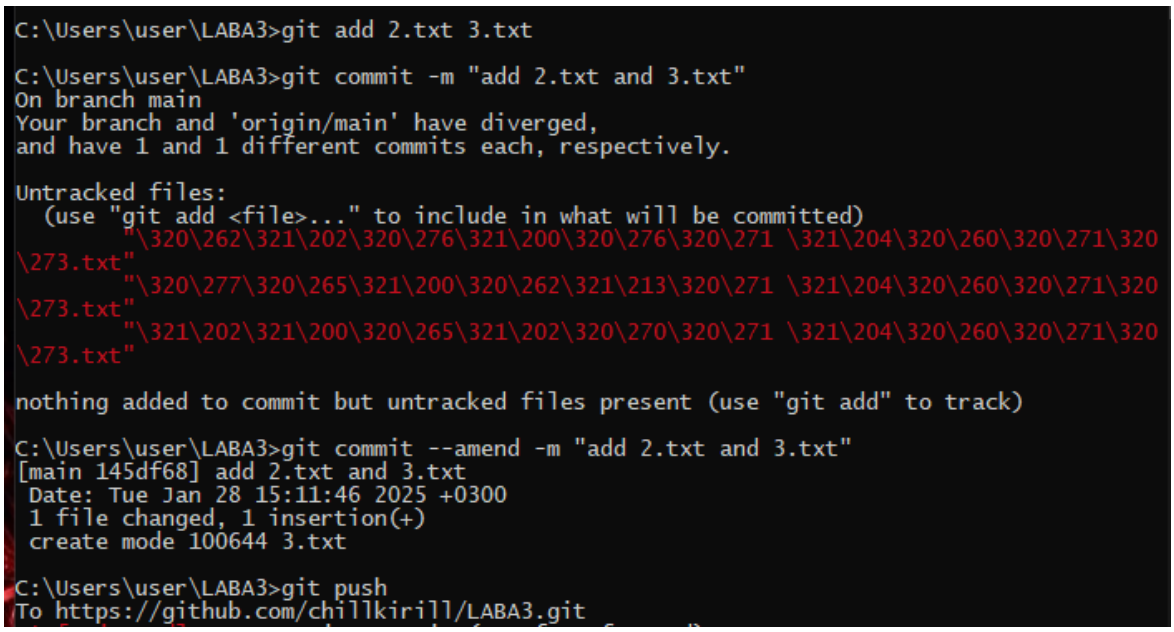

Рисунок 2. Выполнены пункты 4, 5

6. Создать новую ветку my_first_branch
7. Перейти на ветку и создать новый файл in_branch.txt

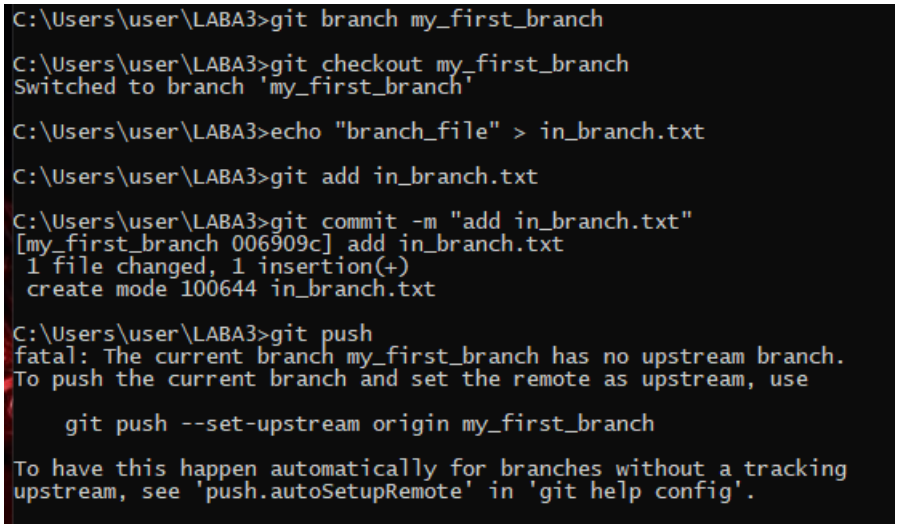

Рисунок 3. Выполнены пункты 6, 7

8. Вернуться на ветку master
9. Создать и сразу перейти на ветку new_branch
10. Сделать изменения в файле 1.txt

```
C:\Users\user\LABA3>git checkout master
error: pathspec 'master' did not match any file(s) known to git

C:\Users\user\LABA3>git checkout -b new.branch
Switched to a new branch 'new.branch'

C:\Users\user\LABA3>echo "new row in the 1.txt file" >> 1.txt

C:\Users\user\LABA3>git add 1.txt

C:\Users\user\LABA3>git commit -m "added new row to 1.txt"
[new.branch 0ec3db4] added new row to 1.txt
 1 file changed, 1 insertion(+)

C:\Users\user\LABA3>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'.
```

Рисунок 4. Выполнены пункты 8, 9, 10

11. Слить ветки master и my_first_branch в new_branch
12. Удалить ветку my_first_branch

```
C:\Users\user\LABA3>git checkout new.branch
Already on 'new.branch'

C:\Users\user\LABA3>git merge master
merge: master - not something we can merge

C:\Users\user\LABA3>git merge my_first_branch
Already up to date.

C:\Users\user\LABA3>git branch -d my_first_branch
Deleted branch my_first_branch (was 006909c).
```

Рисунок 5. Выполнены пункты 11, 12

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

```
C:\Users\user\LABA3>git branch branch_1

C:\Users\user\LABA3>git branch branch_2

C:\Users\user\LABA3>git checkout branch_1
Switched to branch 'branch_1'

C:\Users\user\LABA3>echo "fix in the 1.txt" > 1.txt

C:\Users\user\LABA3>echo "fix in the 3.txt" > 3.txt

C:\Users\user\LABA3>git add 1.txt 3.txt

C:\Users\user\LABA3>git commit -m "fix in 1.txt and 3.txt"
[branch_1 f518667] fix in 1.txt and 3.txt
 2 files changed, 2 insertions(+), 3 deletions(-)
```

Рисунок 6. Выполнены пункты 13, 14

15. Повторить изменения для branch_2

```
C:\Users\user\LABA3>git checkout branch_2
Switched to branch 'branch_2'

C:\Users\user\LABA3>echo "my fix in the 1.txt" > 1.txt

C:\Users\user\LABA3>echo "my fix in the 3.txt" > 3.txt

C:\Users\user\LABA3>git add 1.txt 3.txt

C:\Users\user\LABA3>git commit -m "fix in 1.txt and 3.txt"
[branch_2 5d7f1aa] fix in 1.txt and 3.txt
 2 files changed, 2 insertions(+), 3 deletions(-)

C:\Users\user\LABA3>git push
fatal: The current branch branch_2 has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin branch_2

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

Рисунок 7. Выполнен пункт 15

16. Слить ветку branch_2 в branch_1

17. Отправить ветку branch_1 на GitHub

```
C:\Users\user\LABA3>git checkout branch_1
Switched to branch 'branch_1'

C:\Users\user\LABA3>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\user\LABA3>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\user\LABA3>git remote add origin https://github.com/chillkirill/LABA3.git
error: remote origin already exists.

C:\Users\user\LABA3>git push origin branch_1
Enumerating objects: 15, done.
Counting objects: 100% (15/15), done.
Delta compression using up to 8 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (13/13), 1.05 KiB | 1.05 MiB/s, done.
Total 13 (delta 4), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (4/4), completed with 1 local object.
remote:
remote: Create a pull request for 'branch_1' on GitHub by visiting:
remote:      https://github.com/chillkirill/LABA3/pull/new/branch_1
remote:
To https://github.com/chillkirill/LABA3.git
 * [new branch]      branch_1 -> branch_1
```

Рисунок 8. Выполнены пункты 16, 17

18. Создать ветку branch_3

19. Добавить файл 4.txt

```
C:\Users\user\LABA3>git checkout -b branch_3
Switched to a new branch 'branch_3'

C:\Users\user\LABA3>echo "the final 4.txt file" > 4.txt

C:\Users\user\LABA3>git add 4.txt

C:\Users\user\LABA3>git commit -m "added 4.txt"
U       1.txt
U       3.txt
error: Committing is not possible because you have unmerged files.
```

Рисунок 9. Выполнены пункты 18, 19

20. Выполнить перемещение ветки master на ветку branch_2

21.

```
C:\Users\user\LABA3>git checkout branch_2
1.txt: needs merge
3.txt: needs merge
error: you need to resolve your current index first

C:\Users\user\LABA3>git branch -M branch_2 master
```

Рисунок 10. Выполнен пункт 20

Ответы на 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

(Принудительно: 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**.**