

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 1

Выполнил:
Бакулин Вадим Романович
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.

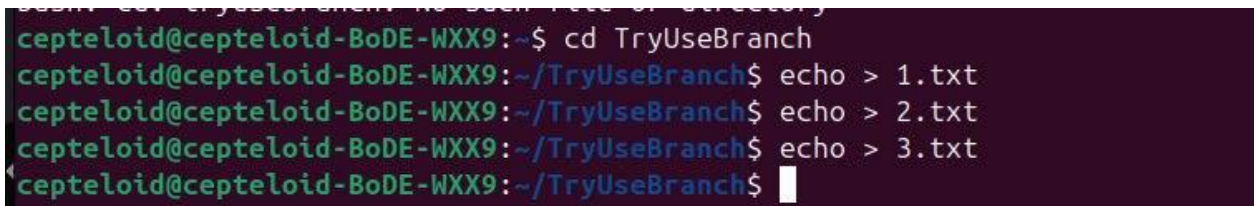
Лабораторная работа 1.3 «Основы ветвления Git»

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

Ссылка на GitHub: <https://github.com/zepteloid/TryUseBranch>

Ход работы:

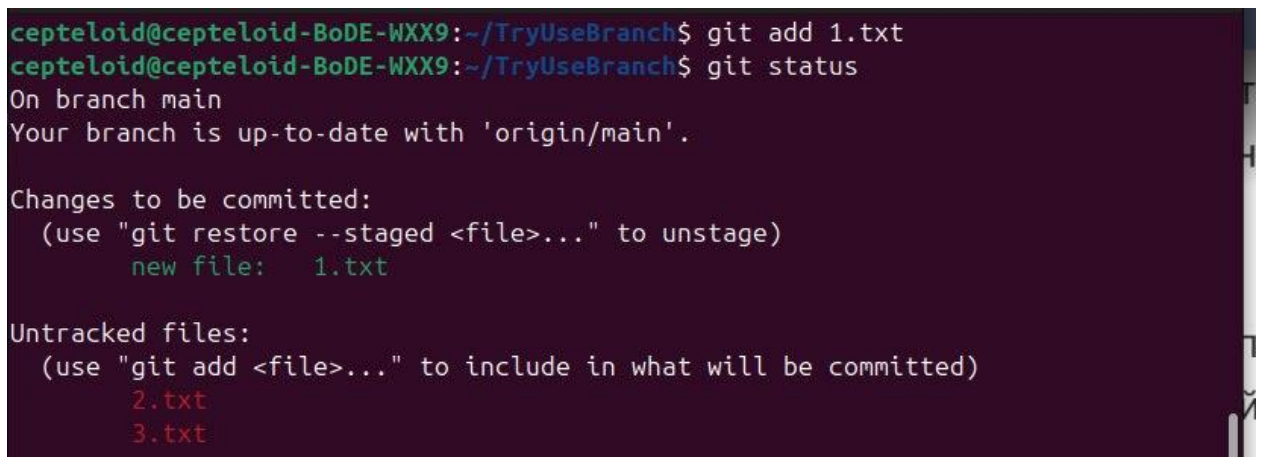
1. Создан общедоступный репозиторий на GitHub
2. Созданы три файла: 1.txt, 2.txt, 3.txt



```
zepteloid@zepteloid-BoDE-WXX9:~$ cd TryUseBranch
zepteloid@zepteloid-BoDE-WXX9:~/TryUseBranch$ echo > 1.txt
zepteloid@zepteloid-BoDE-WXX9:~/TryUseBranch$ echo > 2.txt
zepteloid@zepteloid-BoDE-WXX9:~/TryUseBranch$ echo > 3.txt
zepteloid@zepteloid-BoDE-WXX9:~/TryUseBranch$
```

Рисунок 1. Создание трех файлов

3. Проинициализирован первый файл и сделан коммит с комментарием "add 1.txt file"

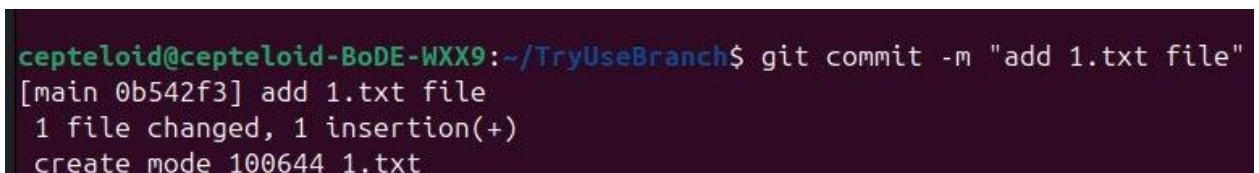


```
zepteloid@zepteloid-BoDE-WXX9:~/TryUseBranch$ git add 1.txt
zepteloid@zepteloid-BoDE-WXX9:~/TryUseBranch$ git status
On branch main
Your branch is up-to-date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   1.txt

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

Рисунок 2. Проинициализирован первый файл



```
zepteloid@zepteloid-BoDE-WXX9:~/TryUseBranch$ git commit -m "add 1.txt file"
[main 0b542f3] add 1.txt file
1 file changed, 1 insertion(+)
create mode 100644 1.txt
```

Рисунок 3. Коммит первого файла

4. Проинициализированы второй и третий файлы

```

cepteloid@cepteloid-BoDE-WXX9:~/TryUseBranch$ git add 2.txt 3.txt
cepteloid@cepteloid-BoDE-WXX9:~/TryUseBranch$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   2.txt
        new file:   3.txt

```

Рисунок 4. Инициализация второго и третьего файла

5. Перезаписать уже сделанный коммит с новыми комментариями

```

cepteloid@cepteloid-BoDE-WXX9:~/TryUseBranch$ git commit --amend -m "add 2.txt and 3.txt"
[main c810067] add 2.txt and 3.txt
Date: Fri Dec 13 10:34:02 2024 +0300
3 files changed, 3 insertions(+)
create mode 100644 1.txt
create mode 100644 2.txt
create mode 100644 3.txt
cepteloid@cepteloid-BoDE-WXX9:~/TryUseBranch$ git log --oneline
c810067 (HEAD -> main) add 2.txt and 3.txt
a2a71e4 (origin/main, origin/HEAD) Initial commit
cepteloid@cepteloid-BoDE-WXX9:~/TryUseBranch$

```

Рисунок 5. Перезапись существующего коммита

6. Создана новая ветка my_first_branch и создан и закоммичен новый текстовый файл

```

cepteloid@cepteloid-BoDE-WXX9:~/TryUseBranch$ git branch my_first_branch
cepteloid@cepteloid-BoDE-WXX9:~/TryUseBranch$ git log --oneline
c810067 (HEAD -> main, my_first_branch) add 2.txt and 3.txt
a2a71e4 (origin/main, origin/HEAD) Initial commit
cepteloid@cepteloid-BoDE-WXX9:~/TryUseBranch$ git checkout my_first_branch
Switched to branch 'my_first_branch'
cepteloid@cepteloid-BoDE-WXX9:~/TryUseBranch$ echo > in_branch.txt
cepteloid@cepteloid-BoDE-WXX9:~/TryUseBranch$ git status
On branch my_first_branch
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        in_branch.txt

nothing added to commit but untracked files present (use "git add" to track)
cepteloid@cepteloid-BoDE-WXX9:~/TryUseBranch$ git add in_branch.txt
cepteloid@cepteloid-BoDE-WXX9:~/TryUseBranch$ git status
On branch my_first_branch
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   in_branch.txt

cepteloid@cepteloid-BoDE-WXX9:~/TryUseBranch$ git commit -m "create file in_branch.txt on branch my_first_branch"
[my_first_branch 09bb253] create file in_branch.txt on branch my_first_branch
1 file changed, 1 insertion(+)
create mode 100644 in_branch.txt
cepteloid@cepteloid-BoDE-WXX9:~/TryUseBranch$

```

Рисунок 6. Создание новой ветки и коммит файла на этой ветке

7. Вернуться на ветку master

8. Создать и сразу перейти на ветку new_branch

```
cepteloid@cepteloid-BoDE-WXX9:~/TryUseBranch$ git checkout -b new_branch
Switched to a new branch 'new_branch'
```

Рисунок 7. Создание и переход на новую ветку

9. Слить ветки master и my_first_branch в new_branch
10. Удалить ветку my_first_branch

```
cepteloid@cepteloid-BoDE-WXX9:~/TryUseBranch$ git branch -d my_first_branch
Deleted branch my_first_branch (was 09bb253).
cepteloid@cepteloid-BoDE-WXX9:~/TryUseBranch$ git branch -d new_branch
Deleted branch new_branch (was e500968).
cepteloid@cepteloid-BoDE-WXX9:~/TryUseBranch$
```

Рисунок 8. Локальное удаление веток

11. Создать ветки branch_1 и branch_2

```
cepteloid@cepteloid-BoDE-WXX9:~/TryUseBranch$ git branch branch_1
cepteloid@cepteloid-BoDE-WXX9:~/TryUseBranch$ git branch branch_2
```

Рисунок 9. Создание веток

12. Изменить файлы 1.txt и 3.txt в ветке branch_1 и ветке branch_2 и слить две ветки:

```
cepteloid@cepteloid-BoDE-WXX9:~/TryUseBranch$ 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.
```

Рисунок 10. Слияние ветки 1 и ветки 2

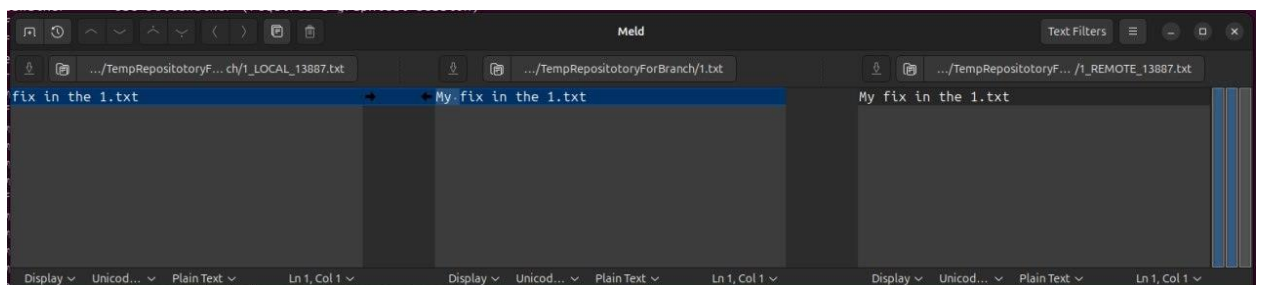


Рисунок 11. Решение конфликта слияния с помощью инструмента meld

13. Отправить ветку branch_1 на GitHub
14. Создать ветку branch_3 с помощью GitHub

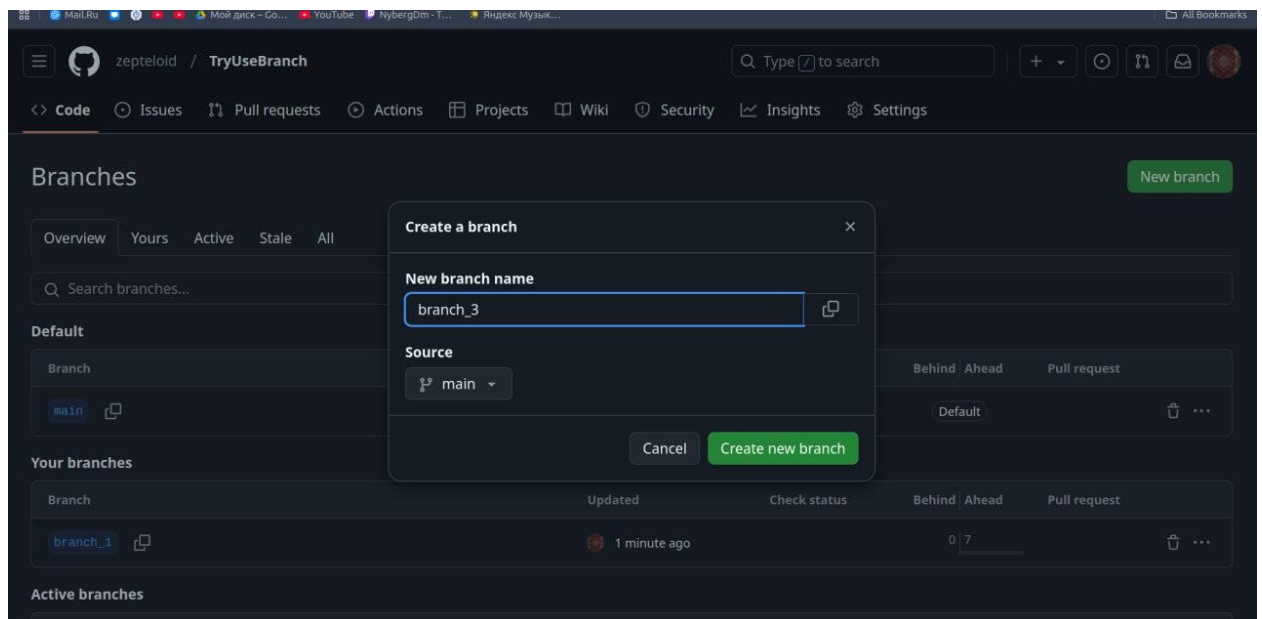


Рисунок 12. Создание ветки с помощью github

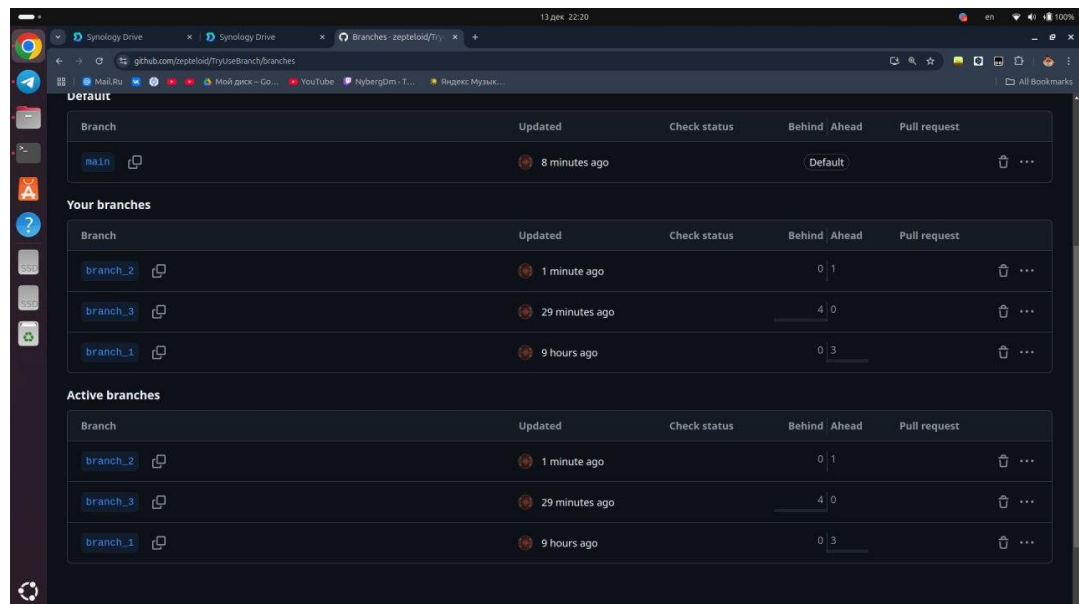


Рисунок 13. Ветки проекта

15. Создать локальную ветку отслеживания

```
cepteloid@cepteloid-BoDE-WXX9:~/TryUseBranch$
cepteloid@cepteloid-BoDE-WXX9:~/TryUseBranch$ git checkout --track origin/branch_3
branch 'branch_3' set up to track 'origin/branch_3'.
Switched to a new branch 'branch_3'
cepteloid@cepteloid-BoDE-WXX9:~/TryUseBranch$
```

Рисунок 9. Локальная ветка отслеживания

16. Выполнить перемещение ветки master на ветку branch_2 и отправил ветку branch_2 на удаленный сервер

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 удалённую 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.