Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий.

Специальность Программная инженерия.

1 курс, 10 группа, 1 подгруппа

Основы Программной инженерии

Выполнил: Лавринович Егор Владимирович

Лабораторная работа №10

Системы контроля версий. Работа с ветками в Git

|  |  |
| --- | --- |
| 3 |  |
| 4 |  |
| 6 |  |
| 7 |  |
| 8 |  |
| 9 |  |

**1. Как используется команда git log?**

Команда git log отображает историю коммитов текущего репозитория.  
Примеры использования:

* git log: отображает полную историю коммитов.
* git log --oneline: выводит коммиты в кратком формате (одна строка на коммит).
* git log --graph: визуализирует дерево веток.
* git log -p: показывает подробности изменений (diff) для каждого коммита.
* git log --since="2 weeks ago": выводит только коммиты за последние 2 недели.

**2. Что такое ветвление?**

Ветвление в Git — это процесс создания независимой линии разработки. Каждая ветка является указателем на коммит, с которого начинается её история. Основная цель ветвления — изоляция изменений, что позволяет работать над новыми функциями или исправлениями ошибок, не затрагивая основную ветку.

**3. Как используется команда git checkout?**

Команда git checkout используется для:

* Переключения между ветками:
* git checkout branch\_name
* Переключения на определённый коммит (в режим "detached HEAD"):
* git checkout commit\_hash
* Восстановления изменений из конкретного файла:
* git checkout HEAD -- file\_name

**Примечание:** В современных версиях Git для переключения веток предпочтительно использовать git switch.

**4. Что такое указатель HEAD?**

HEAD — это указатель на текущую ветку или коммит. Когда вы переключаетесь между ветками, HEAD указывает на последний коммит этой ветки. Если переключиться на конкретный коммит, HEAD становится "отсоединённым" (detached HEAD).

**5. Продемонстрируйте создание новых веток в Git**

1. Создание ветки:
2. git branch branch\_name
3. Создание и переключение на новую ветку:
4. git checkout -b branch\_name
5. (Современный способ) Использование switch:
6. git switch -c branch\_name

**6. Какая команда осуществляет переключение веток?**

* Использование checkout:
* git checkout branch\_name
* Современный способ (рекомендуется):
* git switch branch\_name

**7. Для чего используется команда merge?**

Команда git merge объединяет изменения из одной ветки в другую.  
Пример:

git merge branch\_name

При этом текущая ветка получит изменения из указанной ветки branch\_name.

**8. Какая команда позволяет отобразить историю репозитория?**

Команда git log отображает историю изменений (см. пункт 1). Альтернативы:

* git reflog: отображает историю изменений указателя HEAD.
* git log --oneline --all --graph: удобный формат для анализа веток.

**9. Как выполнить слияние изменений из одной ветки в другую?**

1. Переключитесь на ветку, куда хотите влить изменения:
2. git switch main
3. Выполните слияние:
4. git merge feature\_branch

**10. Какие проблемы могут быть при слиянии и как они разрешаются?**

Основная проблема при слиянии — **конфликты**, возникающие, если изменения в двух ветках затрагивают один и тот же участок кода.

**Разрешение конфликтов:**

1. Git указывает файлы с конфликтами.
2. Откройте эти файлы, найдите конфликтные места (обозначены <<<<<<<, =======, >>>>>>>) и вручную разрешите их.
3. Добавьте изменения в индекс:
4. git add file\_name
5. Завершите слияние:
6. git commit

**Совет:** Для визуального разрешения конфликтов удобно использовать инструменты, такие как VS Code, P4Merge или KDiff3.