Информационные технологии (часть 2)

1. Требования к оформлению отчета

Отчет должен быть оформлен по СТП. Все рисунки должны быть читабельными. На каждом рисунке должна быть видна ФАМИЛИЯ СТУДЕНТА. Обязательно выделяем команды на рисунках в терминале!

2. Объединение веток (часть 2)

Введение

Целью данной лабораторной работы является научиться сливать две отличающиеся ветки для переноса изменений обратно в одну ветку.

Составить список используемых команд в терминале Linux

Команда	Значение	Расшифровать ключи

Задание к работе

1. Создайте папку командой `mkdir` и перейдите в каталог:

```
mkdir GIT
cd GIT
```

2. Проинициализируйте пустой репозиторий в GIT:

```
git init
```

3. Создайте файл командой `echo` и просмотрите содержимое файла:

```
echo '#Лабораторная работа №6' >> file1.md
cat file1.md
```

4. Проиндексируйте файл, просмотрите основную информацию о состоянии репозитория, зафиксируйте файл:

```
git add file1.md
git status
git commit -m "Поясните изменения"
```

5. Создайте первую ветку с помощью команды:

```
git checkout -b ФИО_one
```

6. Создайте вторую ветку с помощью команды:

```
git branch ФИО_two
```

7. Для просмотра имеющихся в репозитории веток нужно ввести эту команду без аргументов:

```
git branch
```

8. Перейдите на ветку `master.` Произведите слияние ветки `ФИО_one` к ветке `master`:

```
git merge ФИО_one
```

9. Удалите первую ветку `ФИО_one`:

```
git branch -d ФИО_one
```

10. Просмотрите имеющиеся ветки (`git branch`). Просмотрите содержимое файла:

```
cat file1.md
```

11. Добавьте заголовок 2-го уровня (произвольный текст).

```
echo '##Цель работы – научиться сливать две отличающиеся ветки' >> file1.md
```

12. Просмотрите содержимое файла через редактор `gedit`:

```
gedit file1.md
```

13. Проиндексируйте файл, просмотрите основную информацию о состоянии репозитория, зафиксируйте файл.

14. Просмотрите историю коммитов в графическом формате:

```
git log --all --oneline --graph
```

15. Просмотр текущих веток `(git branch)`, переход на вторую ветку `(git checkout ФИО_two).` Просмотр содержимого ветки командой `ls.`

16. Добавьте заголовок 3-го уровня (произвольный текст).

```
echo '###Изучаем новые текстовые редакторы: nano, gedit, vi' >> file1.md
```

17. Просмотрим содержимое файла с помощью редактора `vi:`

```
vi file1.md
```

18. Проиндексируйте файл, зафиксируйте файл.

19. Конечный формат лога:

```
git log --pretty=format:"%h %ad | %s%d [%an]" --date=short
```

20. Перейдите на ветку `master.` Измените файл в редакторе `nano` (произвольная текст). Проиндексируйте файл, зафиксируйте файл.

```
git checkout master

nano file1.md

git add file1.md

git commit -m "Поясните изменения"
```

21. Перейдите на ветку ФИО_two, измените файл в редакторе nano (произвольный текст). Проиндексируйте файл, зафиксируйте файл.

```
git checkout ФИО_two

nano file1.md

git add file1.md

git commit -m "Поясните изменения"
```

22. Перейдите на ветку master, произведите слияние ветки ФИО_two к ветке master. В результате возникнет конфликт слияния.

```
git checkout master

git merge ФИО_two
```

В случае возникновения конфликтов `git` заносит в создаваемый при объединении коммит файл, содержащий текст обоих версий. Начало конфликтного фрагмента помечается строкой, начинающиеся с символов `<<<<<<<` и содержащей имя первой ветки, а заканчивается строкой, начинающиеся с символов `>>>>>>>` и содержащей имя вливаемой ветки. Версии из каждой ветки разделяются строкой с символами =======. Такой файл получает статус не объединенный (unmerged).

23. Откройте файл в редакторе и разрешите конфликт, оставив оба изменения. Проиндексируйте файл. Зафиксируйте изменения.

```
nano file1.md

git add file1.md

git commit -m "Разрешен конфликт слияния веток"
```

24. Просмотрите историю коммитов в графическом формате, чтобы убедиться в наличии обеих веток и слияния между ними.

```
git log --all --oneline --graph
```

25. Измените содержимое файла и добавьте коммит в ветку master (`git checkout master`). Повторите этот пункт.

26. Установите указатель новой ветки на коммит с первым изменением файла:

```
git reset [режим] commit
```

В различных сценариях использования может потребоваться выполнять не полный набора этих действий, что задается режимом команды:

`--soft` – выполняет только первое действие;

`--mixed` (режим по-умолчанию) – выполняет 1 и 2 действия;

`--hard` – выполняет все действия.

27. Просмотрите историю коммитов в графическом формате, чтобы убедиться в наличии обеих веток и слияния между ними.

```
git log --all --oneline --graph
```