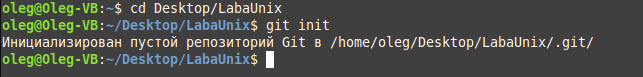
Петров Олег ПИ19-2

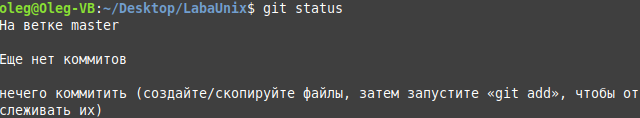
Лабораторная работа №1.2

1. Выберите тематику программы, которую собираетесь написать. Создайте для нее рабочую директорию



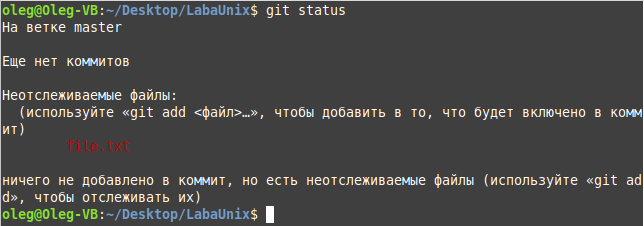
2. Инициализируйте в рабочей директории репозиторий при помощи команды git init.

3. Выполните в репозитории команду git status. Проинтерпретируйте полученное сообщение.

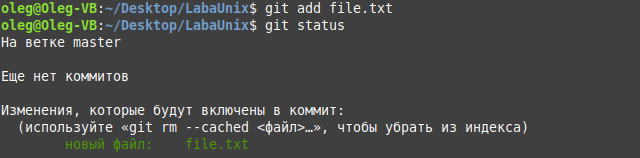


4. Создайте файл для исходного текста программы. Выполните команду git status.

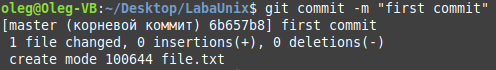




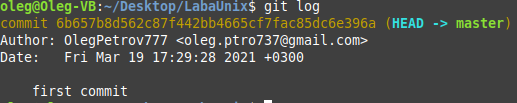
5. Добавьте созданный файл под версионный контроль при помощи команды git add. Еще раз выполните git status.



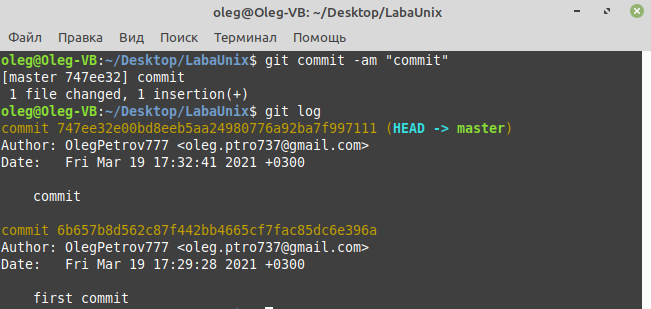
6. Сделайте начальный коммит при помощи команды git commit с опцией -m.



7. Сделайте еще несколько коммитов. Выполните команду git log для просмотра истории коммитов.

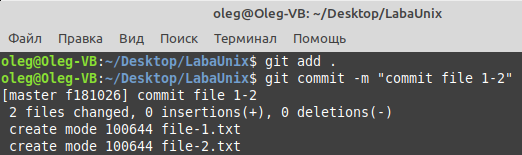


8. Сделайте так, чтобы при коммите измененные файлы автоматически добавлялись в коммит.

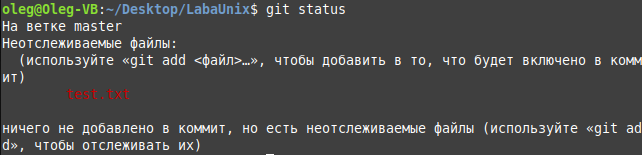


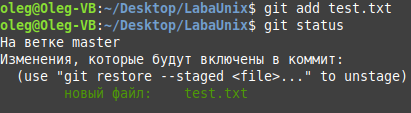
9. Добавьте еще несколько файлов с исходным текстом программы.

10. Добавьте все новые файлы под версионный контроль одной командой.



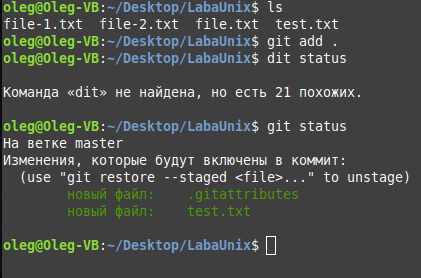
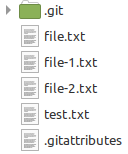
11. На всех стадиях работы пользуйтесь командой git status.



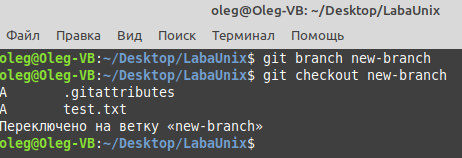


12. Инициализируйте в рабочей директории виртуальное окружение (Если вы пишите не на Python, то можете инициализировать какой-либо программный фреймворк, либо начать работать в IDE, которая создает скрытую папку с настройками в рабочем каталоге).

13. Добавьте созданную служебную папку в файл .gitignore. Проверьте, что они не добавляются в репозитории при добавлении новых файлов с исходным кодом.



14. Создайте новую тематическую ветку git branch. Перейдите в нее с помощью git checkout. Выведите на экран список всех веток.



15. Сделайте несколько коммитов в основную и тематическую ветки.



16. Слейте изменения в основную ветку с помощью git merge. Если произошел конфликт слияния, разрешите его и завершите слияние с помощью git commit.