МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

„КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

НАВЧАЛЬНО-НАУКОВИЙ КОМПЛЕКС

„ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ”

# Лабораторна робота №6

## з курсу «Проектування інформаційних систем»

## Тема: *«Системи контролю версій SVN, GIT»*

Виконав:

студент IV курсу ІПСА

групи ДА-51

Крапивка Микита

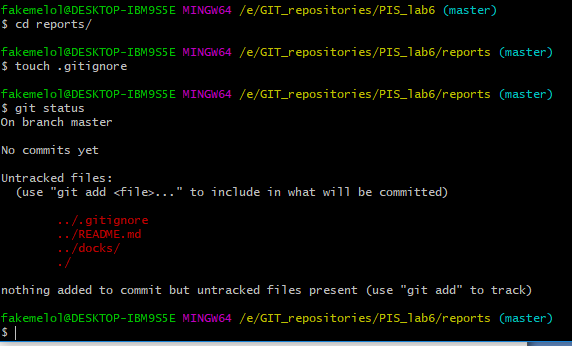
Київ – 2018

1. Мета роботи: за допомогою системи контролю версій завантажити коди програми у репозіторій. Відтворити типовий цикл розробки програмного забезпечення з використанням системи контролю версій.
2. Завдання:

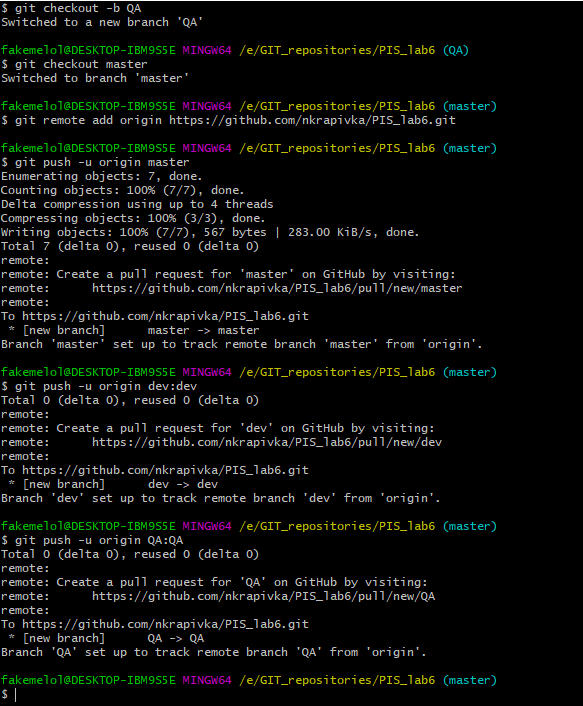
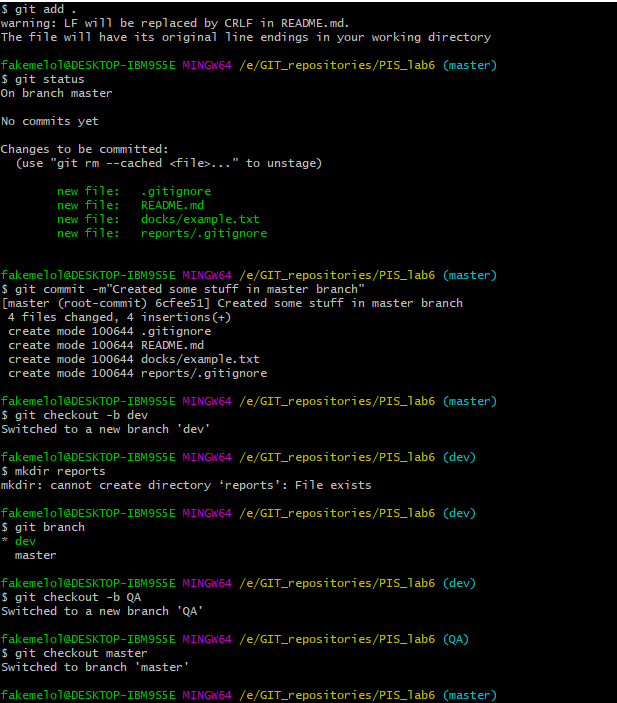
* Обрати безкоштовну систему репозиторія для системи контролю версіями, наприклад projectlocker, або інш.
* Встановити клієнтське безкоштовне програмне забезпечення для роботи с системою контролю версій (GIT, SVN clients).
* Протягом роботи над лабораторними роботами 2-6 використовувати систему контролю версіями.
* Описати цикл розробки програмного забезпечення з використанням системи контрою версій.

1. Результати роботи:

**

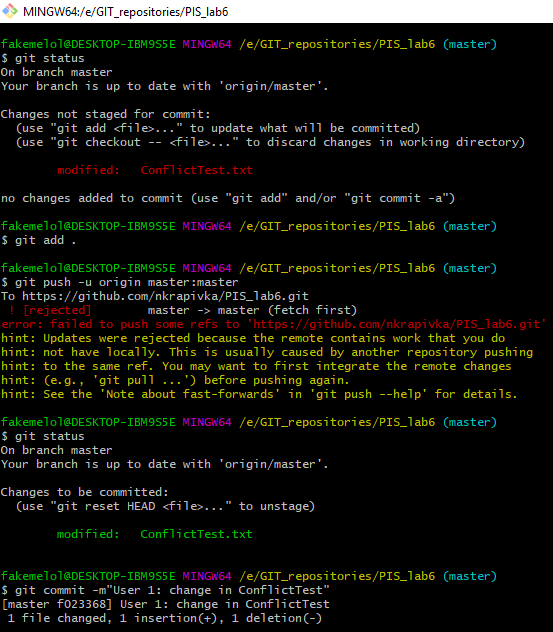
**

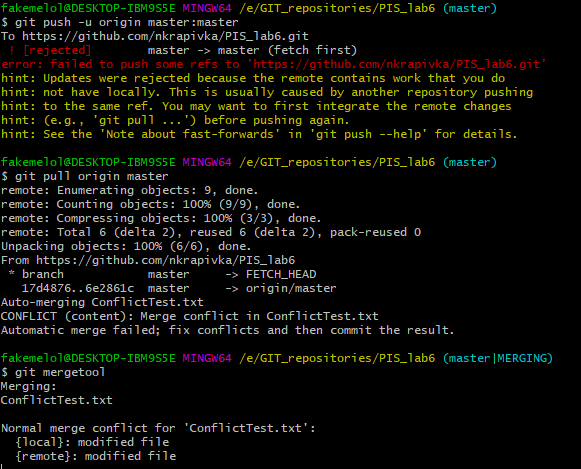
Створення локального репозиторію

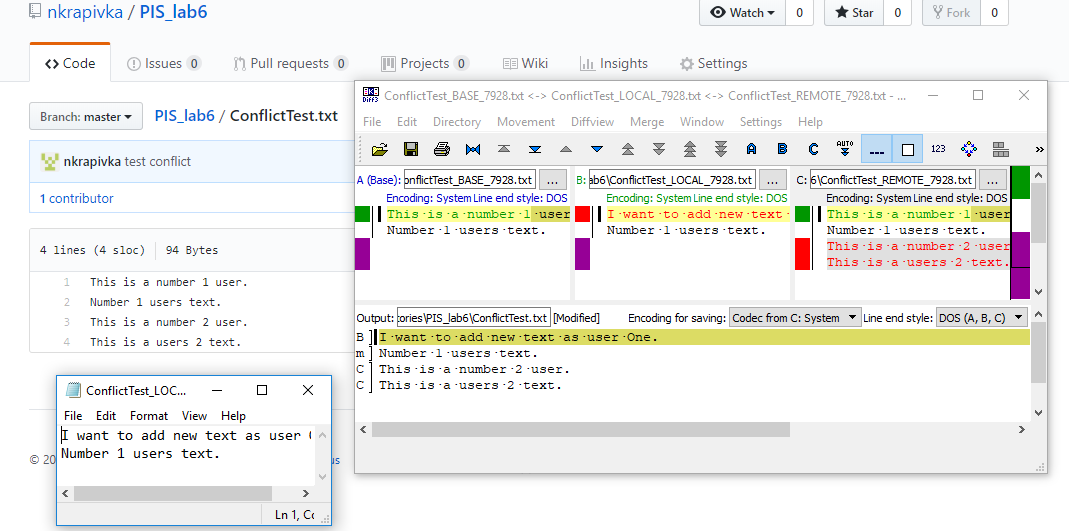
**

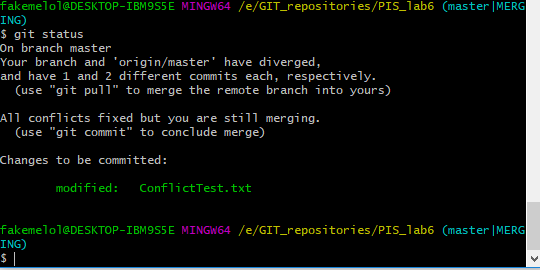
Створення нових гiлок, commit змiн, push from local to remote.

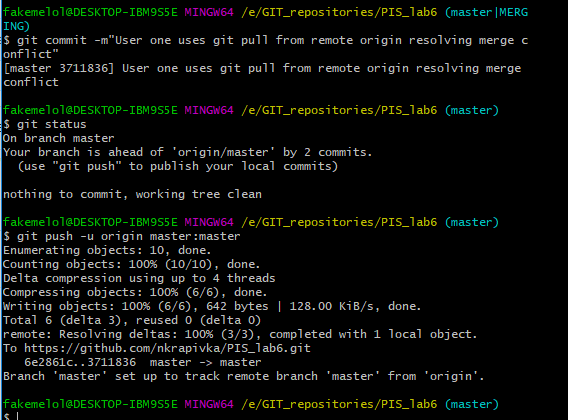
Пробуем решить конфликт: Допустим, над проектом работает несколько человек, и все вносят свои коммиты на удаленный репозиторий. К примеру два человека занимались разработкой одного и того же метода, но вносили разные изменения, в таком случае можно наблюдать такую картину:

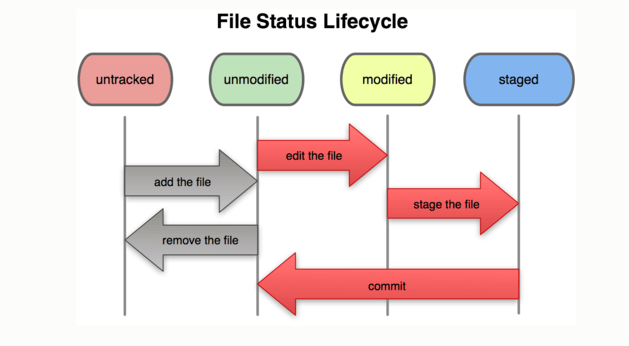
**  
Здесь пользователь номер один сделал изменения в одном из файлов.

**

После успешного коммита пользователь номер 1 пытается сделать Push на remote репозиторий, но получает отказ, так как другой пользователь тоже сделал коммит и выполнил Push ранее. Чтобы двигаться дальше, пользователь номер один должен сначала выкачать изменения второго.  
При попытке выполнения git pull наблюдаем конфликт, который пытаемся разрешить:  
**

**

После вызова mergetool – разрешили конфликт. Осталось выполнить коммит и выполнить push уже готовой новой версии на remote репозиторий.  
**

**

1. Правило користування репозиторієм проекту:
   1. Для розробки інформаційної системи було обрано модель розгалуження Issue Branch Workflow.
   2. Для кожної нової задачі розробник створює нову гілку під назвою, яка відповідає назві тікета у системі проектного трекінгу. Гілка створюється від останнього комміту гілки master. Для кожної задачі створюється своя гілка, і кожна гілка відводиться лише під одне завдання.
   3. Коли задача виконана, розробник зливає гілку задачі в dev-гілку.
   4. Після того, як задача була успішно протестована у dev-гілці, вона заливається розробником у master-гілку.
   5. У випадку виникнення конфліктів злиття, які не вдається вирішити самому, потрібно звернутися до іншого розробника, відповідального за коміти, що конфліктують.

[Комманды git bash](https://git-scm.com/book/ru/v2/Appendix-C%3A-%D0%9A%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4%D1%8B-Git-%D0%9E%D1%81%D0%BD%D0%BE%D0%B2%D0%BD%D1%8B%D0%B5-%D0%BA%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4%D1%8B)

Висновки

Під час виконання даної лабораторної роботи було створено локальний Git – репозиторій, а також віддалений репозиторій на GitHub. Було створено і синхронізовано дві додаткові гілки (крім master): dev (в якій буде вестись розробка інформаційної системи) та test (в якій вона буде тестуватися). Було створено декілька комітів та відіслано протокол лабораторної роботи на віддалений репозиторій.