## Контрольна робота № 1

## Використання комітів та гілок для керування різними версіями файлів за допомогою СКВ Git

### Теоретичні відомості

В процесі виконання лабораторної роботи рекомендується використовувати такі команди (в круглих дужках вказується момент часу в лекції Романа Романовського, а в кутових – сторінка з книги ProGit, де висвітлюються відповідні команди чи параметри):

**Загальні налаштування:**

1. Встановлення імені користувача та адреси його електронної пошти:
    - **git config --global user .name <ім'я без лапок>**;
    - **git config --global user.email <адреса email>** (system – діє для всіх користувачів; global – діє для всіх проектів обраного користувача, без них – тільки для активного проекту <22>).
2. Перевірка встановлених параметрів:
    - **git config -–list (q-вихід з режиму перегляду) (1:27:05).**
    - **git config user.name – видає ім'я користувача. Інші параметри – аналогічно.**

**Робота з файлами і каталогами:**

1. cd <disk>:, cd <dir>/ – перейти на вказаний диск чи в папку.
2. ls – вивести вміст поточного каталогу, при чому файли і каталоги виглядають по різному, –l – вивід в стовбець, -a – вивід з прихованими файлами, -la – разом. dir – вивести всі об'єкти в однаковому форматі.
3. | - результати команди зліва передати команді справа. Наприклад, ls | grep <текст без лапок> - виводить лише файли, в назві яких є потрібний текст.
4. cat <ім'я> - вивести вміст вказаного файла на екран (1:33:30).
5. explorer . – відкрити поточний каталог в провіднику.
6. mkdir <dir> – створити каталог.
7. touch <file> – створити файл. Якщо в підкаталозі, то sub/<file> (зворотній слеш). Назви з пробілами потрібно брати в подвійні лапки.
8. echo <текст> > <file> – вивести текст в файл . Дописати: >>. Текст може бути без лапок.
9. cp <звідки> <куди> – скопіювати файл.
10. mv <звідки> <куди> - перейменувати/перемістити файл (1:05:15).
11. rm – видалити файл, rm –r – видалити каталог, rm –rf – видалити без підтвердження. Якщо не вказати каталог – то може очистити весь диск (55:00).
12. clear – очистити консоль.

**Робота з комітами репозиторію:**

1. git cat-file –t <hash> – вивести тип файла; -p – вивести дані про коміт.

**Робота з локальним репозиторієм:**

1. **git init - створити новий репозиторій. При цьому створюється прихований каталог git з визначеною структурою (56:00).**
2. **git status – видає несинхронізовані файли (58:08),** -s – скорочений перегляд.
3. **git add <назва> - переміщує файл до підготовлених для commit. git add *.txt (59:28), git add * - переміщує всі видимі файли, git add . – переміщує також приховані файли (1:03:10), git add –f <каталог>/ - додати до синхронізації каталог, навіть якщо він – ігнорований об'єкт (1:09:10).**

4. **git commit –m <повідомлення>– відправити файли в локальний репозиторій з вказаним коментарем. Можна без –m <повідомлення> (59:50), --amend – переписати останній коміт, -a – створити коміт з усіх відстежуваних файлів в обхід індекса <39> (якщо файл ще не відстежується, то спочатку має бути add).**
5. **git mv** <звідки> <куди> – перейменовує в індексі <41>.
6. touch .gitignore – створюється файл, вміст якого ігнорується від синхронізації (1:06:40). У цьому файлі вводяться назви файлів, маски файлів, **назви каталогів вказуються з /**, які не піддаються для commit (1.07.24)
7. notepad .gitignore – відредагувати файл ігнорування синхронізації.
8. **git rm – видалити файл з індексу та каталогу, помітившии для вилучення з наступного коміту, --cached – видалити з індексу, повернувши в основний каталог <40>.**
9. **git diff – вивести зміни поточного каталогу відносно індекса, git diff master – показує зміни поточної версії відносно комітів (1:24:17), git diff –staged – вивести зміни індекса відносно останнього коміту <35>.**
10. **git show [коміт]** – вивести зміни, зроблені вказаним комітом.
11. **git restore <file> – відновити файл з індекса в активний каталог.**
12. **git reset – відновити стан індекса з активного коміту , --hard – відновити індекс та робочий каталог <267>, --soft – перемістити лише «голову» <265>, <коміт> – перейти до вказаного коміту <264>, HEAD <file> – вилучити файл з переліку індексованих (відмінити його запис в коміт) <49>.**
13. **git checkout [коміт] <file>** – відновити файл з поточного чи вказаного коміту в активний каталог.
14. **git revert [коміт]** – відмінити зміни, внесені вказаним комітом.
15. **git log – показати історію всіх commit в зворотній часовій послідовності (1:10:20)**, номер коміту при посиланнях може складатися з кількох до 40 символів, -p – показує зміни відносно попереднього коміту, -<N> - обмежує кількість комітів для виведення <43>, --graph – показує галуження гілок, --stat – показує скорочену статистику по змінах в комітах.
16. **git branch** – переглянути наявні гілки; –m <oldBranch> <newBranch> – перейменовує гілку; -d <branch> – видаляє гілку; <branch> <tag> – створює нову гілку у вказаному тегом стані.
17. **git checkout –b <branch>** – створити нову гілку з вказаною назвою.
18. **git checkout <branch>** – перейти до гілки *branch* (фактично, відновити стан коміту, якому вона відповідає).
19. **git merge <branch>** – злити активну гілку з вказаною.
20. **history** – переглянути всі попередні команди (1:09:51).
21. **git help <команда >** - вивести розгорнуту довідку про команду, <команда> -h – вивести скорочену версію довідки.

## Хід роботи

Виконайте наведені нижче завдання. Відповідні команди Git **з тією ж нумерацією** наведіть у звіті. В місцях, позначених ✍, вставте знімки програми Git Bash. Після виконання завдань дайте у файлі звіту письмові відповіді на контрольні питання.

1. Завантажте та встановіть на своєму ПК Git (не GitHub) версії не нижче 2.25 (краще з сайту https://gitforwindows.org/).
2. Завантажте консольну програму Git Bash.
3. Визначте версію встановленої програми.
4. Встановіть ім’я користувача та пароль для ідентифікації комітів.
5. Переконайтеся, що ваші параметри встановлено.

6. Створіть каталог для виконання завдань лабораторної роботи та зайдіить у неї.
7. Створіть в папці лабораторної локальний репозиторій.
8. Створіть текстовий файл **MyFamily**. Введіть в нього своє ПІБ. Виведіть вміст цього файлу в консоль.
9. Створіть перший коміт з цим файлом та запам'ятайте його ідентифікатор.
10. Створіть гілку **Father**.
11. Доповніть файл **MyFamily** ПІБ батька та його батьків. Виведіть вміст цього файлу в консоль.
12. Створіть другий коміт з цим файлом та запам'ятайте його ідентифікатор.
13. Створіть гілку **Mother**.
14. Перейдіть в цій гілці до першого коміту як в індексі, так і в активному каталозі.
15. Доповніть файл **MyFamily** ПІБ мами та її батьків. Виведіть вміст цього файлу в консоль.
16. Створіть третій коміт з цим файлом та запам'ятайте його ідентифікатор.
17. Поверніться в гілку **master**.
18. Перейдіть в цій гілці до першого коміту як в індексі, так і в активному каталозі.
19. Злийте файл цього коміту з гілкою **Father**. Чи виник конфлікт при злитті? Виведіть вміст цього файлу в консоль.
20. Злийте файл цього коміту з гілкою **Mother**. Відредагуйте файл **MyFamily**, в якому виник конфлікт. Виведіть вміст цього файлу в консоль.
21. Створіть четвертий коміт з цим файлом. Виведіть інформацію про всі коміти.

## Контрольні питання

1. Які три ієрархічні структури контролює Git? Коли формується кожна з цих структур?
2. Чим відрізняються дії команд **git commit, git commit –amend** та **git commit –a**?
3. Яка команда створює нову гілку? Переходить в нову гілку? Створює і переходить в нову гілку?
4. Де видаляють файли команди **rm** та **git rm**?
5. Які дії потрібно виконати після усунення конфліктів у вмістах файлів?

Звіт за результатами виконання контрольної роботи здайте через Moodle, а якщо це не вдасться – збережіть його на віддаленому диску, надайте доступ та надішліть посилання на цей файл в коментарях до контрольної роботи через Moodle.