

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота № 1

з дисципліни «Теорія розробки програмного забезпечення»

Тема: «Системи контролю версій. Розподілена система контролю версій «Git»»

Виконав:

студент групи ІА-32
Бечке Олексій Ігорович

Перевірив:

Мягкий Михайло
Юрійович

Київ 2025

Зміст

Вступ.....	2
Теоретичні відомості	2
Хід роботи	4
Висновок	7

Вступ

Тема: Системи контролю версій. Розподілена система контролю версій «Git».

Мета: Навчитися виконувати основні операції в роботі з децентралізованими системами контролю версій на прикладі роботи з сучасною системою Git.

Системи контролю версій використовуються для збереження історії змін файлів у проєктах та забезпечують можливість відновлення попередніх версій. Розподілені системи, такі як Git, дозволяють працювати з локальною копією репозиторію та синхронізувати зміни з віддаленим сервером. У роботі виконано практичне ознайомлення з основними командами Git: ініціалізацією репозиторію, створенням та перемиканням між гілками, додаванням змін до staging area, комітами, злиттям гілок та вирішенням конфліктів.

Теоретичні відомості

Системи контролю версій (СКВ) – це програмні засоби, які дозволяють відслідковувати зміни файлів у проєктах, зберігати історію змін та при необхідності повертатися до попередніх версій. СКВ забезпечують контроль за розробкою програмного забезпечення, дозволяють працювати команді над спільним проєктом і запобігають втраті даних.

Види СКВ

1. Централізовані СКВ

- Мають один центральний сервер, на якому зберігається повна історія проєкту.
- Клієнти звертаються до серверу для отримання останніх версій файлів і внесення змін.
- Приклад: Subversion (SVN), CVS.
- Недоліки: якщо сервер недоступний, робота команди ускладнюється.

2. Розподілені СКВ

- Кожен учасник проєкту має повну локальну копію репозиторію з історією змін.
- Дозволяють автономну роботу та синхронізацію з іншими копіями при наявності з'єднання.
- Приклад: Git, Mercurial.
- Переваги: можливість роботи офлайн, безпечне збереження історії на кожній локальній копії, легке злиття змін.

Git – це сучасна розподілена система контролю версій, яка широко використовується у програмній розробці. Основними особливостями Git є швидкість роботи, ефективне керування гілками та злиттями, можливість роботи офлайн та повна локальна історія змін.

Основні концепції Git

- **Репозиторій (repository)** – сховище проєкту, що містить всі файли та історію змін.
- **Коміт (commit)** – збереження змін у репозиторії з повідомленням, яке описує внесені правки.
- **Гілка (branch)** – паралельна лінія розробки, що дозволяє працювати над окремими функціями без впливу на основну гілку.
- **Merge** – об'єднання змін з однієї гілки в іншу.
- **Staging area** – проміжна область для підготовки змін перед комітом.

Основні команди Git

- `git init` – створення нового Git-репозиторію.
- `git clone <url>` – копіювання віддаленого репозиторію на локальний комп'ютер.
- `git add <файл>` або `git add .` – додавання змін у staging area.
- `git commit -m "повідомлення"` – створення коміту з внесеними змінами.
- `git status` – перегляд стану файлів та репозиторію.
- `git branch` – перегляд локальних гілок або створення нової гілки.
- `git checkout <гілка>` – перемикання на іншу гілку.
- `git merge <гілка>` – злиття іншої гілки у поточну.
- `git log` – перегляд історії комітів.
- `git pull` – отримання останніх змін з віддаленого репозиторію та їх злиття з локальною гілкою.
- `git push` – відправка локальних комітів у віддалений репозиторій.
- `git cherry-pick <хеш>` – перенесення окремого коміту з однієї гілки в іншу.
- `git rebase <гілка>` – перенесення комітів однієї гілки на вершину іншої для лінійної історії.

Хід роботи

Ініціалізація нового порожнього Git-репозиторію у поточній папці.

```
D:\КПІ\3курс\ТППЗ\lab1>git init
```

```
Initialized empty Git repository in  
D:/КПІ/3курс/ТППЗ/lab1/.git/
```

Створення файлу inner.txt

Додавання всіх файлів у поточній папці та підпапках у staging area для підготовки до коміту.

```
D:\КПІ\3курс\ТППЗ\lab1>git add .
```

Створення першого коміту з файлами, які були додані у staging area, з повідомленням "Initial commit".

```
D:\КПІ\3курс\ТППЗ\lab1>git commit -m "Initial commit"
```

```
[master (root-commit) de0cafc] Initial commit  
1 file changed, 0 insertions(+), 0 deletions(-)  
create mode 100644 inner.txt
```

Створення нової локальної гілки з назвою new_branch на основі поточного коміту.

```
D:\КПІ\3курс\ТППЗ\lab1>git branch new_branch
```

Створення нової гілки another_branch і одразу перемикання на неї.

```
D:\КПІ\3курс\ТППЗ\lab1>git checkout -b another_branch  
Switched to a new branch 'another_branch'
```

Перегляд списку всіх локальних гілок і відмітка поточної.

```
D:\КПІ\3курс\ТППЗ\lab1>git branch
```

```
* another_branch  
master  
new_branch
```

Додавання змін до файлу inner.txt (text1)

Додавання всіх змін у робочій директорії до staging area для підготовки до коміту.

```
D:\КПІ\3курс\ТППЗ\lab1>git add .
```

Створення коміту з повідомленням "Commit1" у поточній гілці another_branch.

```
D:\КПІ\3курс\ТППЗ\lab1>git commit -m "Commit1"
```

```
[another_branch bb43cd3] Commit1
```

```
1 file changed, 1 insertion(+)
```

Перемикання на гілку new_branch.

```
D:\КПІ\3курс\ТППЗ\lab1>git checkout new_branch
```

```
Switched to branch 'new_branch'
```

Додавання змін до файлу inner.txt (text2)

Додавання всіх змін у робочій директорії до staging area для підготовки до коміту у гілці new_branch.

```
D:\КПІ\3курс\ТППЗ\lab1>git add .
```

Створення коміту з повідомленням "Commit2" у гілці new_branch.

```
D:\КПІ\3курс\ТППЗ\lab1>git commit -m "Commit2"
```

```
[new_branch af72ba6] Commit2
```

```
1 file changed, 1 insertion(+)
```

Спроба злити гілку another_branch у поточну гілку new_branch; виник конфлікт у файлі inner.txt.

```
D:\КПІ\3курс\ТППЗ\lab1>git merge another_branch
```

```
Auto-merging inner.txt
```

```
CONFLICT (content): Merge conflict in inner.txt
```

```
Automatic merge failed; fix conflicts and then commit the result.
```

Ручне вирішення конфліктів у файлі inner.txt (text1 text2)

Додавання змін після ручного вирішення конфліктів у staging area для завершення злиття

```
D:\КПІ\Зкурс\ТППЗ\lab1>git add .
```

Завершення злиття після вирішення конфліктів, створення merge commit.

```
D:\КПІ\Зкурс\ТППЗ\lab1>git merge --continue
```

```
[new_branch 067dbda] Merge branch 'another_branch' into new_branch
```

Вивід історії комітів у графічному вигляді, показує всі гілки, merge commit і структуру злиття.

```
D:\КПІ\Зкурс\ТППЗ\lab1>git log --all --graph
```

```
*    commit 067dbda6d1693c960b6c29fa55023e9fc57abb00 (HEAD  
-> new_branch)
```

```
|\ Merge: af72ba6 bb43cd3
```

```
| | Author: Oleksiy Bechke <bechke@dlit.dp.ua>
```

```
| | Date:   Mon Sep 22 10:15:04 2025 +0300
```

```
| |
```

```
| | Merge branch 'another_branch' into new_branch
```

```
| |
```

```
| * commit bb43cd3f2ea01a8768904d980b4f43faa6e45462  
(another_branch)
```

```
| | Author: Oleksiy Bechke <bechke@dlit.dp.ua>
```

```
| | Date:   Mon Sep 22 10:13:30 2025 +0300
```

```
| |
```

```
| | Commit1
```

```
| |
```

```
* | commit af72ba66817147bbc7275f48cf0c29ef717750d8
```

```
|\ Author: Oleksiy Bechke <bechke@dlit.dp.ua>
```

```
| Date:   Mon Sep 22 10:14:11 2025 +0300
```

```
|  
|      Commit2  
|  
* commit de0cafc64f751055a937834f5c98e96adf66149a  
(master)  
  Author: Oleksiy Bechke <bechke@dlit.dp.ua>  
  Date:   Mon Sep 22 09:26:38 2025 +0300  
  
      Initial commit  
  
D:\КПІ\3курс\ТПЗ\lab1>
```

Висновок

Під час виконання лабораторної роботи я навчився працювати з розподіленою системою контролю версій git.