Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №1

«Системи контролю версій. Розподілена система контролю версій Git»

Виконав:
студент групи ІА–32
Лось Ярослав

Київ 2025

Вступ

У процесі розробки програмного забезпечення виникає необхідність у керуванні змінами коду, особливо при роботі в команді. Для цього використовуються системи управління версіями (Version Control Systems, VCS), які дозволяють зберігати історію змін, повертатися до попередніх версій та координувати роботу кількох розробників. Сучасні системи такого типу забезпечують зручність, надійність і автоматизацію процесів розробки, що робить їх невід’ємним інструментом програміста.

Зміст

Теоретичні відомості

Системи управління версіями можна умовно поділити на кілька етапів розвитку:

- Ранній етап – копіювання проєктів вручну у вигляді архівів. Першою реальною системою вважається RCS (1982), яка зберігала зміни у вигляді дельт, але працювала лише з текстовими файлами.
- Централізовані системи – у 90-х з'явилися CVS та SVN, які мали центральний репозиторій і дозволяли кільком користувачам працювати одночасно. Недоліком було складне управління гілками.
- Децентралізовані системи – у 2005 році створено Git та Mercurial, що забезпечили швидкість, надійність і незалежність від центрального сервера. Git став найбільш популярною системою завдяки зручній роботі з гілками та розподіленому підходу.
- Хмарні платформи – сучасний етап розвитку (GitHub, GitLab, Bitbucket), що забезпечує інтеграцію з DevOps-процесами, автоматизацію (CI/CD) та зручну командну співпрацю.

Git є розподіленою системою контролю версій, у якій кожен розробник має власний локальний репозиторій. Основні команди:

- git clone – клонування репозиторію;
- git commit – фіксація змін;
- git push та git pull – синхронізація з віддаленим репозиторієм;
- git branch, git merge – створення та злиття гілок.

Таким чином, Git дозволяє ефективно організувати роботу над проєктами, забезпечуючи контроль змін та спрощуючи співпрацю між розробниками.

Хід роботи

Створимо Git репозиторій

```
C:\Sandbox\Lab1>git init
Initialized empty Git repository in C:/Sandbox/Lab1/.git/
```

Створимо текстовий файл і зробимо коміт

```
C:\Sandbox\Lab1>echo "Some text" > text.txt

C:\Sandbox\Lab1>git add text.txt

C:\Sandbox\Lab1>git commit -m "Initial commit"
[master (root-commit) acd6de2] Initial commit
 1 file changed, 1 insertion(+)
 create mode 100644 text.txt
```

Створимо першу гілку за допомогою git branch, перейдемо в неї, змінимо файл і зробимо коміт

```
C:\Sandbox\Lab1>git branch f1
```

```
C:\Sandbox\Lab1>git checkout f1
Switched to branch 'f1'

C:\Sandbox\Lab1>echo "f1" >> text.txt

C:\Sandbox\Lab1>git add text.txt

C:\Sandbox\Lab1>git commit -m "f1"
[f1 aa5bb81] f1
 1 file changed, 1 insertion(+)
```

Створимо другу гілку за допомогою git checkout -b, змінимо текстовий файл і зробимо коміт

```
C:\Sandbox\Lab1>git checkout -b f2
Switched to a new branch 'f2'

C:\Sandbox\Lab1>echo "f2" >> text.txt

C:\Sandbox\Lab1>git add text.txt

C:\Sandbox\Lab1>git commit -m "f2"
[f2 bfa9fb2] f2
 1 file changed, 1 insertion(+)
```

Спробуємо провести merge наших двох гілок

```
C:\Sandbox\Lab1>git merge f2
Auto-merging text.txt
CONFLICT (content): Merge conflict in text.txt
Automatic merge failed; fix conflicts and then commit the result.
```

В результаті виник конфлікт, адже ми змінювали один файл в двох гілках.

Вирішимо цей конфлікт в самому файлі і зробимо коміт.

```
C:\Sandbox\Lab1>git add text.txt

C:\Sandbox\Lab1>git commit -m "final"
[f1 beb0f7d] final
```

В результаті отримаємо наступний граф комітів

```
C:\Sandbox\Lab1>git log --all --graph
*   commit beb0f7dd8b9bbe2d5aaeee557a9012d5d0966c8b (f1)
|\  Merge: aa5bb81 bfa9fb2
| | Author: Yaroslav Los <yarroslavlos@gmail.com>
| | Date:   Mon Sep 22 14:07:02 2025 +0300
| |
| |     final
| |
| * commit bfa9fb26dd786600195338fe35cfaf3113fded57 (f2)
| | Author: Yaroslav Los <yarroslavlos@gmail.com>
| | Date:   Mon Sep 22 14:05:07 2025 +0300
| |
| |     f2
| |
* | commit aa5bb81bd5950975af79077da2682174ec4c30e1
|/  Author: Yaroslav Los <yarroslavlos@gmail.com>
|   Date:   Mon Sep 22 14:05:49 2025 +0300
|
|       f1
|
* commit acd6de205769832cc202891ad4411022a1889345 (HEAD -> master)
  Author: Yaroslav Los <yarroslavlos@gmail.com>
  Date:   Mon Sep 22 14:02:36 2025 +0300

      Initial commit
```

Висновки

Протягом виконання лабораторної роботи я навчився виконувати основні операції з децентралізованими системами контролю версій на прикладі Git. Також я закріпив навички створення репозиторію, додавання та фіксації змін, роботи з гілками, а також об’єднання результатів роботи через команду merge. Окрему увагу приділив вирішенню конфліктів, які виникають під час злиття гілок.

Таким чином, я отримав практичне розуміння того, як Git дозволяє організувати зручну і надійну роботу з вихідним кодом, відслідковувати історію змін та забезпечувати командну взаємодію у процесі розробки.