МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

імені ІГОРЯ СІКОРСЬКОГО»

ЗВІТ

ПРО ЛАБОРАТОРНУ РОБОТУ №1

ТЕМА: «Системи контроля версій. Розподілена система контролю версій «Git».»

Виконала:

Студентка групи ІА-34

Потейчук С.А.

Перевірив:

Мягкий М.Ю.

Київ 2025

Зміст

Теоретичні відомості

Система управління версіями – програмне забезпечення яке призначено допомогти команді розробників керувати змінами в вихідному коді під час роботи. Система керування версіями дозволяє додавати зміни в файлах в репозиторій і таким чином після кожної фіксації змін мани нову ревізію файлів. Це дозволяє повертатися до попередніх версій коду для аналізу внесених змін або пошуку, які зміни привели до появи помилки. Таким чином можна знайти хто, коли і які зміни зробив в коді, а також чому ці зміни були зроблені.

Такі системи найбільш широко використовуються при розробці програмного забезпечення для зберігання вихідних кодів програми, що розробляється. Однак вони можуть з успіхом застосовуватися і в інших областях, в яких ведеться робота з великою кількістю електронних документів, що безперервно змінюються. Зокрема, системи керування версіями застосовуються у САПР, зазвичай у складі систем керування даними про виріб (PDM). Керування версіями використовується у інструментах конфігураційного керування.

Ранній етап

Найпершою системою контролю версій була система «скопіювати і вставити», коли більшість проєктів просто копіювалася з місця на місце зі зміною назва (проєкт_1; проєкт_новий; проєкт_найновіший і т.д.), як правило у вигляді zip архіву або подібних (arj, tar). Звичайно, такі маніпуляції над файловою системою навряд чи можна назвати хоч скільки повноцінною системою контролю версій (або системою взагалі). Для вирішення цих проблем 1982 року з'являється RCS.

Однією з основних нововведень RCS було використання дельт для зберігання змін (тобто зберігаються ті рядки, які змінилися, а не весь файл). Однак він мав низку недоліків.

Етап централізованих систем

На початку 90-х почалася епоха централізованих систем контролю версій. У цей період розробники почали переходити до централізованих систем, що дозволяли працювати кільком користувачам одночасно через сервер

Одна із перших найпопулярніших систем (і досі використовувана) система контролю версій – CVS. Цю епоху можна охарактеризувати досить сформованим уявленням про системи контролю версій, їх можливості, появою центральних репозиторіїв (та синхронізації дій команди).

SVN – у порівнянні з CVS це був наступний крок. Надійна та швидкодіюча систему контролю версій, яка зараз розробляється в рамках проєкту Apache Software Foundation. Вона реалізована за технологією клієнт-сервер та відрізняється неймовірною простотою – дві кнопки (commit, update). Порівняно з CVS, це удосконалена централізована система з кращим управлінням комітами та резервними копіями.

Етап децентралізації

У 1992 році з'явився один з основних представників світу систем розподіленого контролю версій. ClearCase був однозначно попереду свого часу і для багатьох він досі є однією з найпотужніших систем контролю версій будь коли створених.

Гіт є системою розподіленого контролю версій, коли кожен розробник має власний репозиторій, куди він вносить зміни. Далі система гіт синхронізує репозиторії із центральним репозиторієм. Це дозволяє проводити роботу незалежно від центрального репозиторію (на відміну від SVN, коли версіонування передбачало наявність зв'язку з центральним сервером), перекладає складності ведення гілок та склеювання змін більше на плечі системи, ніж розробників та ін.

Mercurial був створений як і Git після оголошення про те, що BitKeeper більше не буде безкоштовним для всіх. Багато в чому схожий на Git, Mercurial також використовує ідею наборів змін, але на відміну від Git, зберігає їх у не у вигляді вузла в графі, а вигляді плоского набору файлів і папок, званих revlog.

Етап хмарних платформ

У сучасну епоху акцент робиться на інтеграції систем контролю версій із хмарними платформами та автоматизації розробки. І в більшості випадків такою системою контролю версій є Git.

Хід роботи

1. Створюємо локальний репозиторій.

```
psa@DESKTOP-LTIHJ5F MINGW64 ~
$ mkdir test1

psa@DESKTOP-LTIHJ5F MINGW64 ~
$ cd test1/

psa@DESKTOP-LTIHJ5F MINGW64 ~/test1
$ git init
Initialized empty Git repository in C:/Users/psa/test1/.git/

psa@DESKTOP-LTIHJ5F MINGW64 ~/test1 (master)
$ git branch

psa@DESKTOP-LTIHJ5F MINGW64 ~/test1 (master)
$ git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)
```

2. Ініціалізація локального репозиторію.

```
psa@DESKTOP-LTIHJ5F MINGW64 ~/test1 (master)
$ git commit -m "init" --allow-empty
[master (root-commit) cd359d1] init

psa@DESKTOP-LTIHJ5F MINGW64 ~/test1 (master)
$ git branch
* master
```

3. Створення трьох гілок.

```
psa@DESKTOP-LTIHJ5F MINGW64 ~/test1 (master)
$ git branch
* master

psa@DESKTOP-LTIHJ5F MINGW64 ~/test1 (master)
$ git branch br1

psa@DESKTOP-LTIHJ5F MINGW64 ~/test1 (master)
$ git branch
  br1
* master

psa@DESKTOP-LTIHJ5F MINGW64 ~/test1 (master)
$ git checkout -b br2
Switched to a new branch 'br2'

psa@DESKTOP-LTIHJ5F MINGW64 ~/test1 (br2)
$ git checkout master
Switched to branch 'master'

psa@DESKTOP-LTIHJ5F MINGW64 ~/test1 (master)
$ git branch
  br1
  br2
* master

psa@DESKTOP-LTIHJ5F MINGW64 ~/test1 (master)
$ git switch -c br3
Switched to a new branch 'br3'

psa@DESKTOP-LTIHJ5F MINGW64 ~/test1 (br3)
$ git branch
  br1
  br2
* br3
  master
```

4. Створення трьох файлів.

```
psa@DESKTOP-LTIHJ5F MINGW64 ~/test1 (br3)
$ git checkout master
Switched to branch 'master'

psa@DESKTOP-LTIHJ5F MINGW64 ~/test1 (master)
$ echo "1" > f1.txt

psa@DESKTOP-LTIHJ5F MINGW64 ~/test1 (master)
$ echo "2" > f2.txt

psa@DESKTOP-LTIHJ5F MINGW64 ~/test1 (master)
$ echo "3" > f3.txt
```

5. Фіксація файлів на різних гілках.

```
psa@DESKTOP-LTIHJ5F MINGW64 ~/test1 (master)
$ git checkout br1
Switched to branch 'br1'

psa@DESKTOP-LTIHJ5F MINGW64 ~/test1 (br1)
$ git add f1.txt
warning: in the working copy of 'f1.txt', LF will be replaced by CRLF the next time Git touches it

psa@DESKTOP-LTIHJ5F MINGW64 ~/test1 (br1)
$ git commit -m "add f1.txt"
[br1 287cd73] add f1.txt
 1 file changed, 1 insertion(+)
 create mode 100644 f1.txt

psa@DESKTOP-LTIHJ5F MINGW64 ~/test1 (br1)
$ git checkout br2
Switched to branch 'br2'

psa@DESKTOP-LTIHJ5F MINGW64 ~/test1 (br2)
$ git add f2.txt
warning: in the working copy of 'f2.txt', LF will be replaced by CRLF the next time Git touches it

psa@DESKTOP-LTIHJ5F MINGW64 ~/test1 (br2)
$ git commit -m "add f2.txt"
[br2 d7f403b] add f2.txt
 1 file changed, 1 insertion(+)
 create mode 100644 f2.txt

psa@DESKTOP-LTIHJ5F MINGW64 ~/test1 (br2)
$ git checkout br3
Switched to branch 'br3'

psa@DESKTOP-LTIHJ5F MINGW64 ~/test1 (br3)
$ git add f3.txt
warning: in the working copy of 'f3.txt', LF will be replaced by CRLF the next time Git touches it

psa@DESKTOP-LTIHJ5F MINGW64 ~/test1 (br3)
$ git commit -m "add f3.txt"
[br3 839a515] add f3.txt
 1 file changed, 1 insertion(+)
 create mode 100644 f3.txt
```

6. Виводимо історію на екран.

```
psa@DESKTOP-LTIHJ5F MINGW64 ~/test1 (br3)
$ git log
commit 839a5155628c39efd4c7f0d837adb23aa92cb0f8 (HEAD -> br3)
Author: anuri-el <sofia.poteychuk@gmail.com>
Date:   Sat Sep 13 17:17:46 2025 +0300

    add f3.txt

commit cd359d1af684b1573da5555c90f9ab0ac0398ef7 (master)
Author: anuri-el <sofia.poteychuk@gmail.com>
Date:   Sat Sep 13 17:13:47 2025 +0300

    init

psa@DESKTOP-LTIHJ5F MINGW64 ~/test1 (br3)
$ git log --all --graph
* commit 839a5155628c39efd4c7f0d837adb23aa92cb0f8 (HEAD -> br3)
| Author: anuri-el <sofia.poteychuk@gmail.com>
| Date:   Sat Sep 13 17:17:46 2025 +0300
|
|     add f3.txt
|
| * commit d7f403bd8cbfaabba52b2ed81fc31e9572f8a499 (br2)
|/  Author: anuri-el <sofia.poteychuk@gmail.com>
|   Date:   Sat Sep 13 17:17:29 2025 +0300
|
|       add f2.txt
|
| * commit 287cd7341585e71797fbf31e7324140373e599e4 (br1)
|/  Author: anuri-el <sofia.poteychuk@gmail.com>
|   Date:   Sat Sep 13 17:17:09 2025 +0300
|
|       add f1.txt
|
* commit cd359d1af684b1573da5555c90f9ab0ac0398ef7 (master)
  Author: anuri-el <sofia.poteychuk@gmail.com>
  Date:   Sat Sep 13 17:13:47 2025 +0300

      init
```

7. Злиття br3 і b2.

```
psa@DESKTOP-LTIHJ5F MINGW64 ~/test1 (br3)
$ git merge b2
merge: b2 - not something we can merge

psa@DESKTOP-LTIHJ5F MINGW64 ~/test1 (br3)
$ git merge br2
Merge made by the 'ort' strategy.
 f2.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 f2.txt

psa@DESKTOP-LTIHJ5F MINGW64 ~/test1 (br3)
$ git log --all --graph
*   commit 00a39d043b80d4a2a775d74081a747ccf2e21a2f (HEAD -> br3)
|\  Merge: 839a515 d7f403b
| | Author: anuri-el <sofia.poteychuk@gmail.com>
| | Date:   Sat Sep 13 17:20:04 2025 +0300
| |
| |     Merge branch 'br2' into br3
| |
| * commit d7f403bd8cbfaabba52b2ed81fc31e9572f8a499 (br2)
| | Author: anuri-el <sofia.poteychuk@gmail.com>
| | Date:   Sat Sep 13 17:17:29 2025 +0300
| |
| |     add f2.txt
| |
* | commit 839a5155628c39efd4c7f0d837adb23aa92cb0f8
|/  Author: anuri-el <sofia.poteychuk@gmail.com>
|   Date:   Sat Sep 13 17:17:46 2025 +0300
|
|       add f3.txt
|
| * commit 287cd7341585e71797fbf31e7324140373e599e4 (br1)
|/  Author: anuri-el <sofia.poteychuk@gmail.com>
|   Date:   Sat Sep 13 17:17:09 2025 +0300
|
|       add f1.txt
|
* commit cd359d1af684b1573da5555c90f9ab0ac0398ef7 (master)
  Author: anuri-el <sofia.poteychuk@gmail.com>
  Date:   Sat Sep 13 17:13:47 2025 +0300

      init
```

Висновки

У ході виконання лабораторної роботи я ознайомилась з основними можливостями системи керування версіями Git та на практиці відпрацювала їх застосування.

Було створено локальний репозиторій, додано та зафіксовано файли. Для закріплення навичок роботи з гілками було створено додаткові гілки, виконано у них зміни та злило. У результаті виконання завдання я закріпила знання щодо створення та ведення репозиторіїв, а також отримала практичні навички, необхідні для ефективної командної розробки програмних проєктів.