# Git

***Роботу***

***виконав Марчук Роман***

***студент групи КСМ-83б***

**Git** — розподілена система керування версіями файлів та спільної роботи. Проект створив Лінус Торвальдс для керування розробкою ядра Linux, а сьогодні підтримується Джуніо Хамано. Git є однією з найефективніших, надійних і високопродуктивних систем керування версіями, що надає гнучкі засоби нелінійної розробки, що базуються на відгалуженні і злитті гілок. Для забезпечення цілісності історії та стійкості до змін заднім числом використовуються криптографічні методи, також можлива прив'язка цифрових підписів розробників до тегів і комітів.

Прикладами проектів, що використовують Git, є ядро Linux, Android, LibreOffice, Cairo, GNU Core Utilities, Mesa 3D, Wine, багато проектів з X.org, XMMS2, GStreamer, Debian DragonFly BSD, Perl, Eclipse, GNOME, KDE, Qt, Ruby on Rails, PostgreSQL, VideoLAN, PHP, One Laptop Per Child (OLPC), АБІС Koha, GNU LilyPond та ELinks і деякі дистрибутиви GNU/Linux (див. нижче).

Програма є вільною і випущена під ліцензією GNU GPL версії 2.

Система спроектована як набір програм, спеціально розроблених з врахуванням їхнього використання у скриптах. Це дозволяє зручно створювати спеціалізовані системи керування версіями на базі Git або користувацькі інтерфейси. Наприклад, Cogito є саме таким прикладом фронтенда до репозиторіїв Git. А StGit використовує Git для управління колекцією латок.

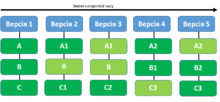
Система має ряд користувацьких інтерфейсів: наприклад, gitk та git-gui розповсюджуються з самим Git.

Віддалений доступ до репозиторіїв Git забезпечується git-демоном, SSH або HTTP сервером. TCP-сервіс git-daemon входить у дистрибутив Git і є разом з SSH найпоширенішим і надійним методом доступу. Метод доступу HTTP, незважаючи на ряд обмежень, дуже популярний в контрольованих мережах, тому що дозволяє використання існуючих конфігурацій мережевих фільтрів.



## Принцип роботи

**Збереження файлів**[

[](https://uk.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB:Git_image_filesystem2.png)

**Рис. 1 *Дані як зліпки стану проекту в часі***

Git, на відміну від Subversion і подібних до неї систем, не зберігає інформацію як список змін (патчів) для файлів. Замість цього Git зберігає дані набором зліпків. Кожного разу при фіксації поточної версії проекту Git зберігає зліпок того, як виглядають всі файли проекту. Але якщо файл не змінювався, то дається посилання на раніше збережений файл (див. рис. 1). Git схожий на своєрідну файлову систему з інструментами, які працюють поверх неї. Для кожного відстежуваного файлу Git зберігає розмір, час створення і останньої зміни. Ці дані зберігаються у файлі index, який знаходиться у теці .git Вся база даних Git зберігається в теці з назвою .git  
В Git файли можуть знаходитися в одному із 3-х станів: зафіксованому (файл вже збережено в локальній базі даних), зміненому (файл було змінено, але зміни не зафіксовано) і підготовленому (файли було змінено і відмічено для фіксації).

**Локальні операції**

Більшість дій можна виконувати на локальній файловій системі без використання інтернет підключення. Вся історія змін зберігається локально і при необхідності вивантажується у віддалений репозиторій. На відміну від Subversion, де без підключення до інтернету можна лише редагувати файли, але зберегти зміни в вашу базу даних неможливо (оскільки вона відключена від репозиторія). Будь-який коміт спочатку робиться локально, а потім вивантажується у віддалений репозиторій.

**Цілісність даних**

В своїй базі Git зберігає все по хешам файлів. Як хешуюча функція використовується SHA-1. Перед кожним збереженням файлів Git обчислює SHA-1 хеш файлу і отриманий хеш стає індексом файлу в Git. Використовуючи хеш Git легко відслідковує зміни в файлах.

**Галуження *(гілки*)**

Галуження — це розмежування від основної лінії розробки. Git дозволяє створити декілька гілок і перемикатися між ними. Це корисно, оскільки дозволяє працювати декільком розробникам над своїм функціоналом не заважаючи іншим і не псуючи основну гілку. За замовчуванням, Git створює гілку з назвою master. Гілка в Git просто являє собою вказівник на одну із фіксацій. При кожній новій фіксації гілка в Git рухається автоматично (тобто перемикається на фіксацію). Гілка є простим файлом, який містить 40 символів контрольної суми SHA-1 фіксації. Створення нової гілки дуже швидке, оскільки це однаково запису в файл 41 байта (40 символів + символ нового рядка).

**Зливання та перебазовування даних**

Git підтримує два способи для інтеграції змін з гілки в гілку: merge (зливання) та rebase (перебазування). Основна різниця полягає в тому, що rebase запам'ятовує фіксації у вигляді патчів, перемотує гілку і застосовує патчі у вигляді фіксацій на відміну від merge, який зливає дві гілки в одну.

## **Приклади**

***Створення локального репозиторію***

Додаємо всі файли

*$ cd <path\_project> #Шлях до проекта (наприклад: $ cd с:/www/git). Переходимо в існуючу папку, в якій буде розміщений проект*

*$ git init #Ініціалізація репозиторію. Створює в каталозі Project каталог .git і в ньому всі необхідні файли репозиторію*

*$ git add . #Додаємо всі файли, які наявні в каталозі Project, під версійний контроль*

*$ git commit -m "Коментар" #Фіксуємо зміни*

***Клонувати проект з віддаленого сервера в локальний каталог***

*$ git clone <remote\_path\_project> #Проект буде клоновано з віддаленого сервера у поточний локальний каталог (наприклад, https://github.com/git/git.git)*

***Отримати останні зміни проекту з віддаленого сервера***

*$ git pull #Всі файли будуть оновлені до останньої версії з віддаленого сервера*

***Перегляд історії змін***

*$ git log #Відображає всі зміни, зроблені в поточній гілці проекта в хронологічному порядку*

***Видалення файлів***

***Видалення з індексу***

*$ git rm --cached README.txt #Видалення файлу README.txt з індексу, тобто він фізично залишається, але зміни в ньому не відслідковуються*

***Видалення з репозиторію***

*$ git rm README.txt #Видалення файлу README.txt з робочого каталогу та індексу !!! Використовувати уважно!*

***Рекурсивне видалення***

*$ git rm --cached -r . #Видалення з індексу всіх файлів в поточному каталозі, включаючи вкладені каталоги*

***Робота з декількома гілками***

***Створення нової гілки***

*$ git branch new\_branch # Створення нової гілки з назвою new\_branch*

*$ git checkout new\_branch # Переключення на гілку new\_branch*

***Перегляд всіх гілок***

*$ git branch # Переглянути всі гілки*

# Веб-інтерфейси

* ***gitweb—*** написаний на Perl.Більшість наведених нижче великих публічних git-репозиторіїв його й застосовує.
* ***wit*** — написаний на Ruby/eRuby ([англ.](https://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D1%96%D0%B9%D1%81%D1%8C%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0) *Christian Meder*)
* ***gitarella*** — Ruby (Diego Pettenò)
* ***git-php***— PHP ([англ.](https://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D1%96%D0%B9%D1%81%D1%8C%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0) *Zack Bartel*)
* ***cgit*** — написаний на C ([англ.](https://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D1%96%D0%B9%D1%81%D1%8C%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0) *Lars Hjemli*)
* ***cogito*** — інтерфейс з командами спроектованими для користувачів звиклих до CVS та SVN ([англ.](https://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D1%96%D0%B9%D1%81%D1%8C%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0) *Petr Baudis*)
* ***StGit*** — написана на Python система керування колекцією латок ([англ.](https://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D1%96%D0%B9%D1%81%D1%8C%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0) *Catalin Marinas*)

## **Обмін змінами з іншими системами керування версіями**

## ***CVS*** — імпорт і експорт, емуляція CVS-сервера, в стандартному постачанні

* ***Subversion*** — імпорт і експорт (частково), в стандартному постачанні
* .tar.gz, .tar.bz2 (серії версійних файлів) — імпорт і експорт, в стандартному постачанні
* ***Perforce*** — імпорт і експорт (частково), в стандартному постачанні і в архівах Git Mailing List
* ***Mercuria***l — імпорт і експорт, в стандартному постачанні і за допомогою зовнішньої програми (Tailor)

Крім того, завдяки своїй структурі даних та форматам даних обміну Git дозволяє порівняно просто і ефективно реалізувати обмін даними практично зі всіма існуючими системами керування версіями.