SCIENCES SORBONNE UNIVERSITÉ

Rapport de projet :

Réalisation d'un logiciel de gestion de versions

BATACHE Toufic et SIDDIQUE Zahra

Chargés de TD :
NOTARANTONIO Hadrien et PANDEY Ayush

LU2IN006 Structures de Données

17 avril 2023

Sommaire

Notice importante

Le code a été écrit, compilé, exécuté et testé sur un MacBook tournant sur MacOS Ventura. Nous sommes conscient du fait que certaines fonctionnalités peuvent ne pas fonctionner comme prévu sur d'autres machines. De plus, des fuites de mémoires sont présentes dans le code, que nous n'avons pas réussi à résoudre faute de temps. Cependant, nous confirmons que toutes les fonctions s'exécutent correctement et sans erreurs. En cas de besoin, nous pouvons fournir les logs des tests.

I - Introduction

Un logiciel de gestion de versions est un outil qui facilite le stockage, le suivi et la gestion de plusieurs versions d'un projet. Il permet d'accéder facilement à l'historique de toutes les modifications effectuées sur les fichiers, ce qui est particulièrement utile pour récupérer une version antérieure en cas de besoin. De plus, il est très bénéfiques pour le travail collaboratif car il permet de fusionner de manière intelligente différentes versions d'un même projet. Les logiciels de gestion de versions sont largement utilisés dans le développement logiciel pour faciliter le travail en équipe et conserver le code source relatif à différentes versions d'un même logiciel.

Le but de ce projet est créer un logiciel de gestion de versions en détaillant les différentes structures de données impliquées dans sa mise en œuvre. Le logiciel s'intitule **lit** — local information tracker. Plus précisément, les fonctionnalités suivantes sont celles qui ont été implémentés : la création d'enregistrements instantanés d'un projet, la navigation libre à travers les différentes versions, la construction et le maintien d'une arborescence des différentes versions du projet et la sauvegarde des changements qui ne sont pas inclus dans un instantané.

II - Structure du projet

Le logiciel est structuré de la manière suivante :

- Le fichier source `litCore.c` (avec son fichier d'en-tête `litCore.h`) qui contient tout le code source au cœur de lit.
- L'exécutable `lit` (dont la source sont les fichiers `lit.c` et du fichier d'en-tête `lit.h`) qui permet à l'utilisateur de communiquer avec le code source

alimentant lit à travers des commandes pouvant êtres entrées dans la ligne de commande.

- L'exécutable `litTest` qui permet d'exécuter les fonctions de test de lit. Les tests sont générés dans le répertoire `tests`.
- Le fichier Makefile qui permet de générer les deux exécutables mentionnés ci-dessus avec la commande `make all`.
- Tout les fichiers nécessaires au fonctionnement de lit sont présents dans le répertoire caché `.lit`.

Le fichier d'en-tête du code cœur de lit contient les structures de données nécessaires à son fonctionnement, notamment les structures :

- `List` qui est une liste doublement chaînée de `Cell`.
- `WorkTree` qui est une structure de données contenant des `WorkFile`.
- `Commit` qui est une `HashTable` qui contient des données hachées stockées dans `kvp` (key-value pair).

III - Description et explication du code

Les commandes utilisateur principales et leurs fonctions respectives dans le code sont les suivantes :

%> ./lit init — initialiser lit en appelant la fonction litInit()

%> ./lit branch <branch_name> — crée une branche qui s'appelle <branch_name> si elle n'existe pas déjà. Si la branche existe déjà, la commande doit afficher un message d'erreur

%> ./lit checkout-branch <branch_name> — réalise un déplacement sur la branche <branch_name>. Si cette branche n'existe pas, un message d'erreur est affiché

%> ./lit add <elem> [<elem2> <elem3> ...] — ajoute un ou plusieurs fichiers/répertoires à la zone de préparation

%> ./lit clear-add — vide la zone de préparation

%> ./lit commit <branch_name> [-m <message>] — effectue un commit sur une branche, avec ou sans message descriptif

%> ./lit merge <branch> <message> — permet de fusionner la branche <branche> avec la branche courante, en permettant de résoudre les conflits s'il en existe