Projet : LP25

LP25 – 2021 – P20

DURR Théo -TC04
CLOAREC Florian -TC04
DURAND Sulyvan -TC04
FANTOU Guillaume -TC04

Suiveur UTBM
LASSABE Frédéric
BAALA Oumaya

utbm
université de technologie
Belfort-Montbéliard

INTRODUCTION

Nous allons vous expliquer comment nous avons abouti à un système expert dans notre projet à l'issue de notre Unité de Valeur LO21.

Tout d'abord nous allons expliquer rapidement ce qu'est un système expert. Cela permet de déduire des conclusions à partir d'une base de faits et d'une base de connaissance qui sont analysées par un moteur d'inférence. La base de connaissance est constituée de règles composées d'une prémisse et d'une conclusion.

Si la prémisse est vraie alors la conclusion le devient aussi et on peut ajouter cette dernière à la base de faits. La prémisse quant à elle se compose de propositions , à noter que la conclusion est aussi une proposition.

Dans le cadre de l'UV LP25 (Initiation à linux et C système) nous devons réaliser un projet. Ce dernier consiste à scanner récursivement un répertoire afin d'en extraire son arborescence qui sera en premier lieu stocké dans la mémoire avant d'être stocké dans un fichier après avoir effectué un tri.

SOMMAIRE

I- Description des concepts et des choix associées

Lors que nous avons commencé ce projet nous nous sommes mis d'accord sur l'IDE à utiliser. Nous avons alors choisis d'utiliser Visual Studio Code pour sa flexibilité d'utilisation. Comme fait depuis le début du cours en c, nous avons utilisé git afin de travailler efficacement. Nous avons préféré utiliser GitHub au lieu de GitLab car l'utilisation du premier est mieux maitrisée. A côté de cela nous avons installé un Windows Subsystem for Linux (WSL). Ceci a été très utile pour utiliser les fonctions linux dans avec notre environnement Windows sans devoir créer de machines virtuelles.

Nous nous sommes ensuite chacun répartis différentes parties pour gagner du temps et ne pas créer de conflits lorsque l'on push sur une branche ou bien lors d'un merge avec Git.

II – Main

La principale utilisation du main va être de passer en paramètre des différentes fonctions les options mises au lancement du programme. Pour cela on utilise getopt puis un switch qui stocke les bonnes valeurs des noms du dossier analysé et du fichier de sauvegarde ainsi que la présence du md5 dans la sauvegarde finale ou non.

Cette partie a nécessité la maîtrise de la fonction getopt que nous avions vu en TD, il a suffi d'un exemple et de quelques recherches complémentaires au cours pour utiliser cette fonction.

III – Scan

L'objectif de la partie scan est de construire la structure de données s_directory et de la remplir avec les données de l'arborescence que nous cherchons à sauvegarder. Pour ce faire la structure de donnée pour représenter un dossier (s_directory) et pour représenter un fichier (s_file) nous ont été donnée avec le sujet. Ce sont des structures récursives faite pour être utilisées au sein d'une liste chaînée. Pour l'implémentation de la fonction process_dir j'ai utilisé les outils finir par dirent afin de parcourir tout le contenu du dossier et en fonction traiter chacun des élément si c'est un dossier en faisant un appel récursif à process_dir et si c'est un fichier en appelant process_file. Les valeurs de retour de ces deux fonctions sont alors ajoutées à la liste chainé associée. Pour la fonction process_file j'ai fait appelle à la fonction stat qui permet de récupérer toutes les informations nécessaire sur le fichier. De plus je fais aussi appel à la fonction process_md5sum faite par mon camarade afin de construire la structure. A la fin de l'exécution de cette partie, la fonction process_dir renvoie un pointeur sur la structure dûment créé et c'est cette structure qui contient en elle les pointeur vers tous les élément contenue par les sous dossier et sous fichier ainsi que toute leur information.

Ce projet m'a beaucoup apporté en termes de méthode de développement. En effet, j'ai l'habitude de travailler sous windows, mais pour les besoin de ce projet j'ai développé sur wsl ce qui m'a donc permis de profiter à la fois de la compilation et de l'exécution sous linux tout en utilisant un éditeur de code en interface graphique. Cela m'a aussi permis de comprendre en appliquant le fonctionnement précis des fonctions que j'ai dû utiliser car j'ai été amené à lire beaucoup de documentation sur internet ainsi que certains articles du livre "Le langage C norme

ANSI". Grâce à ce projet j'ai l'impression de maîtriser beaucoup plus ce langage et d'être capable de résoudre un large panel de problèmes qui pourrait m'être posé dans le monde professionnel.

IV – Save

Pour ma part, ce projet m'aura permis de renforcer mes capacités à répondre à une problématique, à effectuer un travail de groupe ainsi que de renforcer mon expérience en programmation C. J'ai rencontré durant ce semestre des difficultés avec les pointeurs ou encore l'allocation dynamique. J'avais également besoin de me perfectionner en compilation et en manipulation des fichiers. Ce projet de LP25 m'aura aidé à combler ces lacunes. Il aura été très intéressant d'un point de vue théorique.

Certaines difficultés rencontrées notamment des erreurs d'allocations l'ont rendu assez corsé sur la fin. Je ne regrette cependant pas d'avoir eu l'opportunité d'effectuer ce travail très intéressant et instructif. La grande compétence et l'organisation de mon équipe a conforté les efforts effectués sur la fin lorsque la date butoir approchait.

V – Tree

Les fonctions créées dans Tree servent à stocker la hiérarchie en mémoire avant de la mettre dans un fichier. Pour cela on a utilisé les connaissances acquises en LO21 pour faire des ajouts en queue des listes, contenues dans les structures données, pour respectivement les fichiers et les dossiers.

Le nettoyage des structures s_file se fait en itératif mais pour vider la mémoire des structures s_directory on utilise un mélange d'itératif pour vider la liste et de récursif pour vider le contenu de chaque dossier.

Cette partie reposant principalement sur la connaissance des listes, je n'ai pas beaucoup appris vu que j'ai fait LO21 où l'on se concentre sur cet aspect du langage C.

VI – MD5

La partie concernant le codage MD5 a été réalisée à l'aide de la Bibliothèque openssl/md5. Pour rappel, le MD5 sert à vérifier si un fichier a été modifié grâce à une empreinte qu'il suffit de comparer. Dans notre projet la partie MD5 sera utilisée lorsque l'option -s sera donné avec la commande.

Globalement une partie pas très longue mais où on a besoin de comprendre comment marche la bibliothèque. On a des fonctions qui sont très puissantes mais c'est au départ assez perturbant car j'ai eu l'impression de ne rien contrôler. Mais après quelque temps à trouver la

documentation liée à la bibliothèque et à comprendre ce que j'envoie aux fonctions et ce qu'elles me retournent, la tâche s'est plutôt déroulée correctement.

VII – Test et commentaire sur le résultat

Nous allons vous montrer un test du programme avec la commande suivante :

```
sulyvan@DESKTOP-PJAVQGH:~/LP25_project$ ./bin/main -s -i .
*************program start*************

Directory tree written in ./save
*************program stop*************
```

Et en allant dans le fichier save nous trouvons :

```
0   2021-06-15 22:06:54 ./
    1   2021-06-15 15:38:27 245 d02345cf5a699e818f55af5c46c7231e    ./.gitignore
    1   2021-06-09 13:59:59 35149   1ebbd3e34237af26da5dc08a4e440464    ./LICENSE
    1   2021-06-09 13:59:59 14  29ed756cf86bfed50643746a0d192929    ./README.md
    1   2021-06-15 20:47:24 367 f399976c708c5ef9414a35ccf58ea693    ./CMakeLists.txt
    1   2021-06-15 20:47:24 549 9f1abc6ff2bc6c67156a7af6afb633d7    ./Makefile
    0   2021-06-15 20:47:24 ./src/
        1   2021-06-15 20:47:24 748 c4d3458dbdbf08a98b1183a5dbc09b88    ./src/utils.h
        1   2021-06-15 20:47:24 719 d0e628950fc12160ceada984547e55ad    ./src/md5sum.c
        1   2021-06-15 15:38:27 892 36e698bb8787dc07951b6e6cccd07a3b    ./src/scan.h
        1   2021-06-15 20:47:24 3712    c6384035af6af107d2202cfd1967378f    ./src/scan.c
        1   2021-06-15 20:47:24 1424    f5f40fa6d85327f7687f165a2d006c6c    ./src/tree.h
        1   2021-06-15 20:47:24 1547    3029c50e27090a7eae8d4e4628811a49    ./src/types.h
        1   2021-06-15 20:47:24 1513    232153f1588d9e1ab43242b3edde047e    ./src/tree.c
        1   2021-06-15 15:38:27 1027    137dfee5d0defd3afe68c5ae9e8d72a0    ./src/utils.c
        1   2021-06-15 20:47:24 544 deac6ce0db6d381b1450b59904299053    ./src/md5sum.h
        1   2021-06-15 20:47:24 1543    36159777b1199b007bf3521592ac8195    ./src/save.h
        1   2021-06-15 20:47:24 3864    60dd1fc1bb32bbbaa4774038b6687fdb    ./src/save.c
        1   2021-06-15 20:47:24 749 27aa1bb39691569ebd84115778c12031    ./src/include.h
        1   2021-06-15 20:47:24 1832    6d4f19673cc9e981458c8d24f775fee4    ./src/main.c

```

Ce qui est bien le résultat attendu avec la date puis la taille du fichier puis la somme MD5 et enfin le chemin pour aller au fichier. On voit aussi l'arborescence avec les tabulations.

DURR Théo
CLOAREC Florian
DURAND Sulyvan
FANTOU Guillaume

Projet : LP25 – A20

Résumé

Ce projet a pour but de créer un système expert en algorithme puis de le coder en langage C. Ce dernier termine l'UV LO21 dans le cadre du Tronc Commun 03.

utbm
université de technologie
Belfort-Montbéliard