Rapport

Serge Téhé, Louis Lafond

Octobre 2022 - Décembre 2022

Promotion 2024

PROJET SYSTEME

Responsable du projet : RACCOUCHOT MAIWEN

18 décembre 2022

Sommaire

1 Introduction

1.1 Objectif du projet

L'objectif de notre projet est de programmer une commande ftc qui se comporte comme la commande find. Le langage imposé est le langage C qui nous fournit les appels système adéquats pour la programmation de cette commande. Nous avons programmé plusieurs options de la commande find telles que -name, -date, -size, -mime, -ctc, -dir et quelques options optionnelles décrites dans la partie 2 du projet telles que -link, -ou, -perm.

1.2 Organisation

Pour travailler sur notre projet, nous décidons de nous partager les questions dans l'ordre des questions du sujet. Chaque semaine, chacun devrait faire 2 questions, par contre le plus rapide pouvait continuer les autres questions s'il avait du temps.

2 Fonctionnalités de base

2.1 Organisation des fichiers et Choix de conception

2.1.1 Organisation des fichiers

Dans chaque fichier questioni.c (2<=i<=9), nous avons codé les fonctions réalisant l'option de ladite question, généralement les fonctions sont de la forme findbyregex ou findbydate ou findbysize etc... Dans le fichier question1.c, nous avons codé une fonction parsecommand qui parse les arguments de la ligne de commande, détermine si le flag est dans la liste des options et exécute la fonction associée à chaque flag. Cette fonction gère aussi les erreurs. Enfin dans un fichier main.c, nous appelons la fonction parsecommande sur les arguments de la ligne de commande.

2.1.2 Choix de conception

La fonction de base qui permet de parcourir une arborescence est findallpath. Il s'agit d'une fonction récursive qui vérifie si l'élément courant de l'arborescence est un fichier ou un dossier. S'il s'agit d'un dossier on rappelle la fonction sur ce dossier. Toutes les autres fonctions de type findbyregex, findbydate, findbysize, findbyperm, findbyctc, findbymime, findbydir utilisent ce même principe. En effet lorsqu'on arrive à une feuille de l'arbre, on vérifie si ce fichier respecte le critère voulu, par exemple pour le flag -name : si le nom du fichier match le paramètre de la ligne de commande alors ce fichier est selectionné. Nous avons utilisé comme structure de données une liste chainée qui ajoute en tête le nom du fichier ou du dossier dépendant des questions. Ainsi pour chaque fonction exécutée, nous avons une liste chainée allouée correspondante qui contient les fichiers ou dossiers vérifiant les critères voulus, ce qui sera très utile pour les fonctions qui doivent matcher plusieurs critères passés en paramètre.

2.2 Fonction "fonction_ET" - Question4.c

La fonction "fonction_ET" doit permettre de gérer la présence de plusieurs flags en ligne de commande et en ne sélectionnant que les fichiers répondant à tous les critères passés. On sait que chaque fonction alloue une liste chainée qui contient des fichiers ou dossiers satisfaisant le critère voulu par ladite fonction ; Lorsque le flag "-ET" est détecté, il récupère toutes les options et paramètres en ligne de commande et fait l'intersection de toutes les listes chainées correspondantes puis on ajoute le résultat dans une nouvelle liste chainée.

2.3 Fonction "find_by_mime" - Question7.c

Cette fonction nécessite l'utilisation d'une librairie particulière, la librairie MegaMimes développée par Kobby Owen sous la license libre du MIT. Elle est accessible à l'adresse https ://github.com/kobbyowen/MegaMimes. Il s'est avéré qu'il existe très peu de libraires permettant de travailler sur les types mime de fichiers, donc nous avons décidé d'utiliser cette librairie qui semblait fournir de bons résultats. C'est une librairie qui permet d'obtenir le type mime d'un fichier à partir de son nom ou de son arborescence, mais aussi d'autres caractéristiques telles que sa taille, ou le nom mime.

3 Extensions

Nous avons pu developper 3 extensions -link, -ou et -perm.

3.1 Fonction "fonction_OU" - Question4.c

Cette fonction repose sur le même principe que la fonction "ET" vue précédemment, sauf que nous faisons un "ou" logique sur les listes chainées.

3.2 Fonction findbylink

Cette fonction repose exactement sur le même principe que les fonction findbyregex ou findbydate. Lorsqu'on arrive à une feuille de l'arbre on vérifie s'il s'agit d'un lien symbolique à l'aide de la structure dirent (on vérifie si le type est DT_LNK) et on l'empile dans la structure de données si oui. Sinon, s'il s'agit d'un dossier on rappelle la fonction sur le chemin du dossier et les autres paramètres.

3.3 Fonction findbyperm

Pour la fonction perm, nous avons utilisé le champ mode de la structure stat qui retourne une valeur entière en base 10 que nous avons converti en base 8 et récupérer les 3 derniers chiffres qui correspondent aux valeurs octales des permissions attribuées respectivement au propriétaire, au groupe et aux autres. On parcourt ainsi l'arborescence et on vérifie si les permissions du fichier courant correspondent à celles entrées en ligne de commande.

4 Difficultés et résolution

Nous avons rencontré certaines difficultés comme le fait que nous ayons des résultats différents en fonction des terminaux. Ce problème nous a pris assez de temps parce que c'était la première fois que cela nous arrivait et c'était vraiment étrange. Nous avons vu monsieur Bouthier qui nous a dit que cela est lié à la gestion de la mémoire. Ainsi nous avons vérifier qu'il n'y a aucune fuite mémoire mais le problème persistait. Après nous nous sommes rendus compte que cela provenait des appels systèmes. En effet, on oublait de faire un closedir du dossier dans les fonctions.

Une autre difficulté est survenue lorsque nous avons voulu tester nos différentes fonctions, en particulier avec la gestion de la mémoire lors de l'utilisation des listes chainées. Nous avons plusieurs fois obtenu des problèmes de "Segmentation Fault" dûs à certaines particularités des tests, c'était le cas avec la fonction mime qui utilise une bibliothèque externe pour reconnaître le type mime d'un fichier. Seulement, si le type mime du fichier passé en argument n'est pas reconnu par la librairie, un problème de mémoire apparaît lors de l'appel de la fonction, et c'est une particularité à laquelle on ne s'attendait pas et à laquelle on s'est adapté.

5 Conclusion

5.1 Conclusion Louis Lafond

Je suis arrivé à TELECOM Nancy cette année, donc directement en deuxième année pour suivre mon cursus de double-diplôme depuis l'ENSGSI. Ce projet m'a donc fait découvrir et apprendre le langage C. En arrivant, le fonction et l'organisation nécessaire pour rendre un projet dans l'informatique m'étaient inconnus. J'ai dû apprendre à utiliser le GitLab, trouver et apprendre à utiliser un IDE qui me convienne, en bref, j'ai dû rattraper mon retard et apprendre ce qui avait été enseigné en première année concernant l'informatique en peu de temps. Ce sont parfois de choses qui peuvent sembler ridicules, comme lorsque j'ai compris comment "pull" et "push" sur GitLab, mais nécessaires. Pour cela, je suis très reconnaissant envers Yohou qui m'a toujours aidé : il a été suffisamment patient pour m'expliquer et m'aider à avancer. Le langage C n'est pas le plus intuitif, et je me suis plongé dans la compréhension du fonctionnement de la mémoire pour écrire mes programmmes. A propos de programmation, j'ai l'impression d'avoir été plutôt lent et je remercie encore Yohou d'avoir accepté mon retard et de m'avoir soutenu ! Heureusement pour l'avancée du projet, il est à l'aise avec le langage C et la programmation en général, il m'a beaucoup appris. J'ai aimé travailler sur ce projet, j'ai l'impression d'avoir progressé en programmation, non seulement au niveau technique mais aussi au niveau de la mise en place d'un projet en général, choses qui m'étaient inconnues jusqu'à maintenant.

5.2 Conclusion Yohou Téhé

Ce projet m'a permi d'avoir plus d'expériences en C à travers la connaissance des appels systèmes et les gestions d'erreurs que je n'avais jamais rencontrées auparavant. Aussi il m'a permi d'apprendre à mieux gérer mon temps, vu la multitude de projets qu'on devrait faire simultanément.

6 Annexe

Ci-dessous, un tableau récapitulatif du nombre d'heures approximatives passées sur les différentes étapes du projet.

Tâches	Louis	Yohou
Question1	6h	1h
Question2	0h	1h
Question3	0h	2h
Question4	7h	4h
Question5	0h	4h
Question6	0h	3h
Question7	18h	10mins
Question8	0h	5h
Question9	0h	1h
Question10-11	2h	4h
Extensions	0h	7h
Résolution de problèmes de mémoire et tests	6h	10h
rédaction du rapport	4h	3h