Projet de Réseau-Système
2022/2023

Rapport de projet

---

18 Décembre 2022

Noé-Laurent Laurent
Edgar PERONNIN

Responsable de module :
Maiwenn RACOUCHOT

Table des matières

1 Introduction **3**

2 Choix de conception **3**

3 Extensions développées **3**

4 Difficultés rencontrées **3**

5 Répartition du travail **4**

6 Conclusion **4**

1 Introduction

Ce projet a pour objectif de réaliser un programme qui permette de retrouver à partir d'un point de départ donné, un fichier dans une arborescence à la manière de la commande find. Celui-ci doit être intégralement réalisé en langage C et prendra en compte de nombreux paramètres qui permettront de rechercher un fichier selon les critères voulus. La commande s'exécutera de la manière suivante dans le cas général : ./ftc POINT_DE_DEPART -CRITERE_DE_RECHERCHE PARAMETRE

2 Choix de conception

Nous avons décidé d'utiliser des variables globales FLAG afin de récupérer les options présentes dans la ligne de commande. Une fonction récursives s'occupe ensuite de parcourir toute l'arborescence de fichier et chaque test de la ligne de commande est effectué sur chaque fichier, un test étant géré par une fonction.
Par ailleurs, le programme utilise plusieurs structures de données données dans des librairies :

- La structure dirent qui permet de gérer le parcours de dossiers et de fichier. Nous l'utilisons pour reconnaître le nom et le type des fichiers et des dossiers.
- La structure regex qui permet de gérer les expressions régulières pour -name et -ctc.
- La structure stat qui permet d'accèder à certaines informations d'un fichier telles que la date de dernier accès pour -date ou les permissions accordées pour -perm.

Ces trois structures de données viennent de trois librairies C : dirent.h, sys/stat.h et regex.h .

3 Extensions développées

Nous avons pu implémenter les extensions suivantes :

- -color : Il colore en vert les dossiers du path et en rouge le fichier.
- Les mots clés de -date : Il permet de mettre des mots tels que "today" ou "yesterday" en argument de -date.
- -perm : Il retourne les fichiers ayant les permission spécifiées en argument de cette option.
- -ou : Il permet de retourner les fichiers vérifiant une des options spécifiées.

4 Difficultés rencontrées

La principale difficulté rencontrée fut de tester notre programme. En effet, nous ne pouvions pas convenablement tester nos fonctions, entraînant donc des erreurs sur les tests de Gitlab. Pour résoudre ce problème, nous avons dû obtenir les tests présents dans l'image docker afin de pouvoir tout effectuer en local. Cela nous a permis de savoir ce qui était attendu et donc de corriger bien plus efficacement le programme. Cependant, beaucoup de tests passent en local mais pas sur le pipeline de Gitlab. Nos tests sur -date, -perm, -dir et -mime fonctionnent tous en local, alors qu'aucun d'entre eux ne passent les tests d'intégration continue sur Gitlab.

L'implémentation du "et" nous a demandé un peu de réflexion : nous avions au début pensé à introduire une structure de liste chaînée afin d'ajouter dans celle-ci l'ensemble des fichiers étant en accord avec le premier test, puis de supprimer au fur et à mesure les fichiers qui ne sont pas valides sur les tests suivants. Cependant, nous voulions utiliser quelque chose de plus simple à mettre en place, nous avons donc finalement pris la décision d'utiliser une série de variables globales nommées "FLAG_*OPTION*" (le flag passe à 1 si l'option est présente dans la ligne de commande, dans ce cas, la fonction correspondante est donc testée et le path du fichier actuel est retournée si tous les tests passent).

L'option -mime nous a également demandée du temps. Nous avions eu l'idée d'utiliser le fichier mime.types mais il aurait fallu créer des fonctions afin de récupérer les informations contenues dans

ce fichier, ce qui aurait demandé plus de temps. Nous avons donc décider d'utiliser la librairie MegaMimes créer par "kobbyowen" sur Github. Celle-ci nous permet de récupérer les extensions liées à un type mime spécifique, il ne nous reste plus qu'a les comparer à l'extension des fichiers parcourus. Pour cela nous avons du comprendre comment bien utiliser les fonctions présentes dans MegaMimes et quels formats d'arguments et de valeurs retournées utiliser. Il a donc fallu gérer les fichiers qui n'avaient pas d'extensions (tel qu'un Makefile) et faire attention au fait qu'un type mime comme "text" n'existe pas pour MegaMimes mais qu'il faut ajouter un "/*" pour qu'il reconnaisse tous les types text.

5 Répartition du travail

	Conception	Implémentation	Tests	Rédaction du rapport	Total
Noé	1h	18h	3h	2h	24h
Edgar	2h	17h	2h	1h	22h

TABLE 1 – Temps passé sur chaque étape

6 Conclusion

Pour finir ce projet nous a permis d'encore améliorer notre maîtrise du langage C. Celui-ci a aussi été une réussite au niveau de la communication dans notre binôme car après 3 projets effectués ensemble, nous avons pu rapidement nous mettre d'accord sur la conception du code à réaliser puis réussir à s'entraider lorsque cela était nécessaire durant l'implémentation. Aussi, nous avons pu enrichir notre culture générale informatique avec notamment la découverte des types mimes qui nous étaient complètement inconnus jusqu'alors. Enfin, nous avons conscience que notre code peut-être largement mieux optimisé car nous avons pu comparer la différence de vitesse d'exécution entre lui et la commande find.