Rapport de Projet de Système

Rémi Bourdais, Louis Chatard

Octobre - décembre 2022

Table des matières

1 Introduction

Ce projet a été réalisé pour le module de système. L'objectif de ce projet était d'implémenter en C, une fonction qui ressemble à la commande bash "find" qui permet de rechercher des fichiers au sein d'un répertoire de travail en fonction de plusieurs paramètres du fichier comme son nom, sa taille, la date de sa dernière modification, son type, etc.
Plusieurs arguments de recherche optionnels ont été proposés dans les attendus du projet comme l'affichage en couleur, la recherche par permission, la prise en compte des liens symboliques, des paramètres supplémentaires pour les arguments, etc.

2 Implémentation du projet

2.1 Fonctionnement du programme

Pour pouvoir implémenter un tel programme, on récupère premièrement les options de recherche grâce au paramètre natif du langage C *char *argv[]* dans la fonction *main* qui permet de récupérer la liste des chaînes de caractères espacées qui ont été données en appelant le programme.

On vérifie ensuite qu'il y a au moins un deuxième argument qui correspond au répertoire de recherche demandé. Une fois cette première vérification franchie, on analyse les arguments correspondants aux options de recherche et leur paramètre éventuel, on vérifie également que le paramètre de l'option de recherche est du bon format avant de les stocker dans un tableau. Maintenant que l'on sait ce que l'on cherche, on va parcourir l'arborescence de fichiers à partir du répertoire donné à l'appel, et l'on stocke tous les chemins (relatifs au dossier de départ demandé) des fichiers dans une liste. Pour chaque paramètre, on actualise la liste de chemins en ne gardant que ceux qui satisfont les conditions demandées par l'utilisateur.

Une fois tous les arguments examinés, on affiche l'ensemble des fichiers satisfaisant les demandes de l'utilisateur.

2.2 Structures de données utilisées

Pour gérer les paramètres et les chemins d'accès aux fichiers, nous avons réfléchi à une façon de stocker l'ensemble de ces variables et nous avons décidé de créer trois structures : une qui retient les attributs d'un arguments, le tableau avec tous les arguments et la liste de chemins des fichiers.

```c
typedef struct flag {
    bool isflag;
    char* flagname;
    char* flagvalue;
} flag;

typedef struct tabflag {
    struct flag* tab;
    int size;
} tabflag;


typedef struct listfile {
    char* path;
    struct listfile* next;
} listfile;
```

FIGURE 1 – Définition des structures implémentées

Structure *flag*

La structure *flag* stocke les options données aux paramètres appelés et possède trois paramètres :

- Un booléen *isflag* qui retient si le paramètre a été appelé par l'utilisateur
- Une chaîne de caractères *flagname* stockée en char*, qui contient le nom du paramètre à donner pour l'appeler
- Une chaîne de caractères *flagvalue* stockée en char*, qui contient la valeur avec laquelle l'utilisateur souhaite utiliser ce paramètre.

Structure *tabflag*

La structure *tabflag* est une liste simplement chaînée des paramètres appelés pour la recherche, et possède deux paramètres :

- Un flag* *tab* qui pointe vers le premier paramètre appelé du tableau
- Un entier *size* qui contient le nombre de paramètres appelés.

Structure *listfile*

La structure *listfile* est une liste simplement chaînée de chaînes de caractères qui correspondent aux chemins des fichiers correspondant aux paramètres déjà testés, elle possède deux paramètres :

- Une chaîne de caractères *path* stockée en char*, qui contient le chemin relatif par rapport au dossier demandé
- Un pointeur vers un listfile *next*, qui pointe vers le chemin du fichier suivant.

Évidemment, lorsque le programme est appelé avec le paramètre *-dir*, ce sont les chemins des dossiers et non fichiers qui sont stockés dans cette structure.

2.3 Différents paramètres de recherche développés et options possibles

La liste des paramètres définie dans les attendus comprenant à la fois des paramètres obligatoires et facultatifs, la liste de ceux que nous avons intégrés au programme avec une succincte description et les options acceptées est visible sur le tableau 2.

2.4 Architecture de l'application

Pour réaliser ce programme nous avons dû créer de nombreuses fonctions, aussi bien pour gérer les structures de données que nous avons créées et le parcours dans l'arborescence de fichiers, que pour la gestion des critères de recherche de l'utilisateur. Le *main*, la fonction principale qui gère l'entièreté de l'application est définie dans le fichier ftc.c nommé comme le programme. Le parcours de l'arborescence de fichiers est géré dans le fichier listfiles.c tandis que les fonctions plus générales ou relatives au structures de données sont définies dans le fichier util.c. Ensuite, nous avons créé un fichier par critère de recherche possible avec à l'intérieur, les fonctions nécessaires pour utiliser chaque critère. Enfin pour gérer les mauvaises entrées utilisateurs qui pourraient empêcher la recherche, nous avons un fichier error.c avec les fonctions qui renvoie à l'utilisateur, l'erreur qu'il a faite en appelant le programme.
Bien entendu, pour définir toutes les fonctions et gérer toutes les importations nécesssaires, chaque fichier .c est accompagné d'un *header* .h qui se trouve dans le répertoire *includes*.

3 Organisation du projet

3.1 Organisation au sein du groupe

Pour la réalisation de ce projet, nous ne nous sommes pas beaucoup penchés dessus au début et nous avons vite été dépassés par les examens et autres projets que nous avons eus depuis novembre. C'est pourquoi le projet est passé en second plan, face aux autres rendus et attendus du semestre. La semaine précédant les vacances de fin d'année a permis la réflexion sur la façon d'implémenter le projet qui n'avait pas réellement eu lieu lors de l'écriture des premières fonctions. Nous avons donc pu passer à l'écriture de la majeure partie du code sachant vers où nous allions. Rémi Bourdais s'étant coupé un doigt, il n'avait qu'une main ce qui rend compliqué et allonge le temps de frappe au clavier ; c'est pourquoi la majeure partie de l'écriture s'est faite en *LiveShare* sur l'ordinateur, en étant en appel pour réfléchir ensemble au fur et à mesure que l'on avançait. Le rapport a donc été majoritairement rédigé par Louis Chatard, avec l'aide et l'écoute des recommandations de Rémi Bourdais.

Paramètres	Description
-test	Ce paramètre indique que l'on teste le programme, il n'est pas utile pour se servir de ce dernier. Cependant il renvoie le paramètre et sa valeur associée. Seuls les paramètres possédant une valeur sont affichés.
-name	Cherche les fichiers dont le nom contient l'option donnée. Cela supporte les chaînes de caractères et le regex.
-size	Cherche les fichiers qui correspondent a une taille demandée, celle doit être du format (+ou-) ?XcXkXMXG, où X représente des entiers. Quand la taille demandée commence par un +, la recherche portera sur les fichiers de taille supérieure à ce qui est demandé, avec un - ce sera ceux de taille inférieure et sans signe, on cherche ceux dont la taille est égale à ce qui est demandé. Si l'on ajoute aucune lettre, l'unité par défaut sera le c (qui correspond à un byte) ; le k correspond au kilobyte (1024b), le M correspond au megabyte (1024*1024b) et le G correspond au gigabyte (1024*1024*1024b)
-date	Cherche les fichiers plus récents qu'une durée au format XjXhXm où X représente des entiers, on peut accéder aux fichiers plus anciens en écrivant +XjXhXm. Plusieurs durées littérales sont acceptées comme *now*, *today*, *yesterday*, *this week*, *this month* et *this year*.
-mime	Cherche les fichiers en fonctions de leur type *mime*, cela fonctionne aussi bien avec un type général ou un sous-type spécifique.
-ctc	Cherche les fichiers contenant la chaîne de caractères passée en option à l'intérieur du fichier, cela accepte aussi du regex.
-dir	Cherche parmi les dossiers plutôt que les fichiers, appelé sans paramètres il affiche la liste de tous les dossiers, on peut rajouter directement une chaîne de caratères (ou regex) pour rechercher un nom de dossier.
-color	Affiche les résultats en couleur
-perm	Cherche les fichiers ayant la permision passée en paramètre, celle-ci doit être au format XXX, où $X \in 0, 1, ..., 7$.
-link	demande à ftc de suivre les liens symboliques.
-ou	Cherche les fichiers remplissant au moins une des conditions passées en paramètres

FIGURE 2 – Description des critères de recherche possibles et de leur utilsation

3.2 Répartition en volume horaire

Nous avons synthétisé dans le tableau 3 un volume horaire estimé de chacun des membres à chaque partie du projet.

	Louis Chatard	**Rémi Bourdais**
Conception	4.5h	4.5h
Implémentation	18h	21h
Tests	18h	21h
Rédaction du rapport	4h	2h

FIGURE 3 – Volume horaire estimé

4 Conclusion

Pour finir ce rapport, nous pouvons dire que l'organisation du groupe n'a pas été parfaite, de par le fait que la plupart du projet ait été faite à la fin et que tout le temps possible pour la réalisation n'a pas pu être pleinement exploité ce qui a limité le développement des fonctionnalités supplémentaires proposées dans le sujet. Cependant la période de réflexion qui a précédé l'écriture du code a permis de savoir ce qu'il y avait à faire et comment le faire, ce qui a permis une implémentation rapide du programme. La gestion de l'handicap partiel de Rémi Bourdais a été bien gérée bien que celui-ci aurait moins entravé l'implémentation du projet si celle-ci avait été plus étendue dans le temps, et plus anticipée. D'un point de vue pédagogique, ce projet a permis de revoir l'utilisation de structure de données travaillée en première année ainsi que l'utilisation de librairies C qui peuvent être intéressantes à savoir manipuler parce qu'elles peuvent éventuellement être nécessaires dans de futurs programmes.