TELECOM Nancy

Rapport de projet

Projet Systèmes "find the cat"

Auteurs
Thibaut CAGNON
Antoine YEBOUET

Responsable de module
Maiwenn RACOUCHOT
Christophe BOUTHIER

Table des matières

Chapitre 1

Introduction

1.1 Contexte

Dans le cadre de ce projet, l'objectif était de réaliser un programme ftc (find the cat) dont le but est de permettre à un utilisateur de trouver le chemin d'un fichier dans une arborescence de répertoires. Ainsi, la commande à utiliser dans un terminal était :
`./ftc [répertoire de départ]`
L'objectif second était d'implémenter plusieurs options afin d'affiner les recherches de fichiers.

Chapitre 2

Choix de conception

2.1 Boucle principale (Parser) et Structures

Afin de trouver tous les fichiers pour une commande donné. Nous avons choisi de parser la commande et de récupérer toutes les options et paramètres dans une structure. Lors du parcours du tableau `argv`, s'il s'agit d'une option, on exécute la fonction associée pour remplir la structure. On dispose de plusieurs fonctions, qui nous permettent d'assigner pour une option, la valeur du flag correspondant dans la structure. Nous réalisons ensuite le parcours de l'arborescence de fichiers, et nous vérifions pour chaque fichier si celui-ci répond aux critères de la structure. Nous ne réalisons donc qu'une seule boucle sur l'arborescence.

```
typedef struct struct_command
{
    char *name;
    char *size;
    char *date;
    char *mime;
    char *ctc;
    char *dir;
    int yesdir;
    char *color;
    char *perm;
    char *link;
    char *threads;
    int ou;
    int nb_of_flags;
} struct_command;
```

FIGURE 2.1 – Strucure utilisée pour stocker les options de la commande

En cas de mauvais argument (flag incorrect, flag sans valeur, flag qui apparaît plus d'une fois etc... Un message d'erreur est renvoyé sur la sortie standart et le programme s'arrête.

Chapitre 3

Fonctionnalités du programme

3.1 Extensions développées

3.1.1 Fonctionnalités obligatoires

Toutes les options obligatoires ont été implémenté dans leur totalité. Les options `-name` et `-ctc` fonctionne avec des regex. L'option `-size` accepte les paramètres +/-. L'option -date accepte l'usage d'un + dans ses paramètres.

Concernant la fonctionnalité ET inhérente au programme qui permet d'afficher uniquement les fichiers qui matchent tous les critères de la commande, nous gérons cette fonctionnalité lors du parcours de l'arborescence. En effet, pour chaque fichier/dossier parcouru, celui-ci est analysé à l'aide de fonctions de comparaison entre le fichier étudié et les paramètres d'options enregistrés dans notre structure. Un ET-logique est réalisé entre toutes ces fonctions.

3.1.2 Extensions optionnelles

Nous avons également réalisé d'autres fonctionnalités :

— OU-logique : en utilisant l'option -ou dans la commande. Cela indique au programme de réaliser un OU-logique lors de la lecture des options. Ainsi, le programme affichera tous les fichiers et dossiers correspondant à l'une des options passés en paramètre.

— Colorisation de l'affichage : en utilisant l'option -color, l'affichage sera le même, mais en gras et en jaune.

— Trouver un fichier selon ses permissions avec l'option `-perm` : en récupérant les permissions d'un fichier, on obtient une valeur, que l'on convertit ensuite en octal : on a alors une valeur de la forme : `100XXX` où `XXX` représente les permissions du fichier, allant de `000` à `777`. On compare donc cette valeur à celle entrée par l'utilisateur, et on retourne `1` si les valeurs sont identiques, `0` sinon.

— Une option -cat.

3.2 Bibliothèques utilisées

Pour la réalisation du projet nous avons utilisé plusieurs librairies C :

— `stdlib.h`
— `stdio.h`
— `string.h`
— `dirent.h`
— `sys/stat.h`
— `ctype.h`
— `time.h`
— `regex.h`

Nous avons aussi utilisé la librairie MegaMimes réalisée par Kobby Owen, pour l'option `-mime` (https ://github.com/kobbyowen/MegaMimes)

3.3 Difficultés rencontrées

La charge de travail importante à la fin du semestre (incluant les partiels et les autres projets en cours) ne nous a pas toujours permis d'avancer régulièrement sur l'implémentation des fonctionnalités. Cependant, nous avons tout de même su nous organiser afin de le terminer.

De plus, côté technique, nous avions des problèmes de mémoire, qui n'étaient pas forcément visibles dans le terminal, ce qui faisait échouer les tests en ligne. Il s'agissait de mémoire dans la pile. Selon nous, lorsque la pile était totalement remplie, le défaut de stockage impliquait que les variables crées ne pouvaient plus être stockées. Nous avons résolu ces problèmes en utilisant des allocations de mémoire (malloc).

Chapitre 4

Répartition du travail

Auteur	Conception	Implémentation	Tests	Rédaction du rapport	Total
Thibaut Cagnon	3	31	2	3	39
Antoine Yebouet	3	30	2	3	38