Rapport Projet Système

Albert BECQUET, Lucas RIOUX

18 Décembre 2022

Table des matières

1 Choix de conception

1.1 Organisation du code

L'ensemble de nos fichiers contenant le code source est situé dans le dossier `src`. L'importation des bibliothèques est réalisé dans le fichier `main.h`. Pour chaque structure de données utilisée ainsi que chaque fonction, une documentation a été réalisée dans le fichier `main.h`. Le fonctionnement global est situé dans le fichier `main.c`, tandis que chaque commande importante aura son propre fichier `fichier.c` associé. Le fichier `utils.c` regroupe toutes les fonctions utilitaires à notre projet.

1.2 Structures de données choisies

Voici les différentes structures de données que nous avons décidé d'utiliser pour notre projet. Voici la structure permettant la gestion des options, elle est composée de plusieurs champs booléens définis par défaut à `false` et dont les valeurs deviennent `true` si les options concernées sont utilisées :

```
struct option_t
{
    bool basique;
    bool nom;
    bool taille;
    bool date;
    bool mime;
    bool ctc;
    bool dir;
    bool perm;
    bool color;
    bool ou;
};
```

Voici la structure permettant de répertorier les paramètres de toutes les options de la commande exécutée. C'est ici que sont stockées le contenu des paramètres des fonctions.

```
struct val_t
{
    char nom[1000];
    size_params_t size_params;
    char date[100];
    mime_params_t mime_params;
    char ctc[1000];
    char perm[4];
};
```

Voici la structure spécifique aux paramètre de l'option -size :

```
struct size_params_t
{
    char signe;
    long taille;
    char symbole_taille;
};
```

On y stocke le signe du paramètre de -size, ainsi que la taille et le symbole de la taille. Par exemple, pour la commande `ftc . -size +9k`, '+' est le signe, '9' est la taille et 'k' est le symbole de la taille. La structure est remplie pendant le parsing du paramètre de -size, ce qui permet de gérer en même temps les erreurs de saisie du paramètre.

Voici la structure spécifique aux paramètre de l'option -mime :

```
struct mime_params_t
{
    char type[50];
    char subtype[50];
    bool has_subtype;
};
```

On y stocke le type du fichier, ainsi que son sous-type. Par exemple, si un fichier a pour type MIME `text/html`, le type sera `text` et le sous-type sera `html`. Étant donné que la recherche de fichiers peut également se faire seulement par type, on a rajouté un champ `has_subtype` qui stocke si le type MIME spécifié par l'utilisateur de la commande contient un sous-type.

Voici la structure de données spécifique à l'option -date :

```
struct date_params_t
{
    char color;
    bool etat;
};
```

Voici la structure de données spécifique à la fonction de vérification des fichiers :

```
struct verif_params_t
{
    char color;
    bool etat;
};
```

2 Extensions développées

2.1 Commandes principales

2.1.1 -test

Cette option permet d'observer le bon fonctionnement du système, en vérifiant que l'option suivie par -test et son paramètre sont bien acceptés.

2.1.2 -name

Cette option permet de trouver un fichier dans une arborescence en fonction de son nom. La fonction de vérification du nom prend en paramètre le nom qu'on souhaite rechercher ainsi que le nom du fichier à tester. Elle effectue un test permettant d'accepter les expressions de type regex, en utilisant la bibliothèque `regex.h` [3], et renvoie un booléen.

2.1.3 -size

Cette option permet de trouver un fichier dans une arborescence en fonction de sa taille. Après avoir parsé le paramètre de l'option -size avec la fonction `parseSizeParameter`, les champs de la structure `size_params_t` contiennent les informations relatives au paramètre : son signe, sa taille et le symbole associé à la taille. Le signe est facultatif : s'il n'y en a pas, on recherche des fichiers ayant exactement la taille spécifiée. La taille est, quant à elle, obligatoire. Le symbole de taille est également facultatif : par défaut, `c` est utilisé.

Symbole	**Taille recherchée**
+	Strictement supérieure à la taille saisie
-	Strictement inférieure à la taille saisie
(aucun)	Égale à la taille saisie

TABLE 1 – Signe du paramètre de -size

Symbole	**Symbole de taille**
c	1 byte. Utilisé par défaut
k	1.024 bytes
M	1.048.576 bytes
G	1.073.741.824 bytes

TABLE 2 – Signe du paramètre de -size

2.1.4 -date

Cette option permet de trouver un fichier dans une arborescence en fonction de sa date de dernière modification. La bibliothèque `time.h` [7] a été utilisé pour cette fonction, ainsi que la bibliothèque `sys/stat.h` [1].

Symbole	Durée
m	minutes
h	heures
j	jours

Table 3 – Symboles de base

Les symboles suivant ont été réalisés afin d'effectuer des recherches plus intuitives en accord avec les mots clés existants.

Symbole	Durée
now	-1m
today	-1j
yesterday	-2j
this month	-30j

Table 4 – Symboles optionnels

2.1.5 -mime

Cette option permet de trouver un fichier dans une arborescence en fonction de son type MIME : application, audio, image, text, video... La recherche de fichiers peut s'effectuer selon le type et le sous-type, ou bien uniquement selon le type. Le paramètre de l'option -mime est parsé grâce à la fonction `parse_mime_parameter`, ce qui remplira les champs d'un structure `mime_params_t`. Ensuite, lors du parcours de l'arborescence, le type MIME du fichier sera récupéré, puis comparé avec le type MIME recherché grâce à la fonction `check_mime_type`. En cas de correspondance, le chemin du fichier sera affiché sur le terminal. La libraire MegaMimes [4] a été utilisée : elle permet notamment de récupérer le type MIME d'un fichier avec la fonction `getMegaMimeType`.

2.1.6 -ctc

Cette option permet de trouver un fichier dans une arborescence en fonction de son contenu. Tous les fichiers vont être ouverts et parcourus chacun leur tour, et leur contenu sera stocké dans une variable. Ensuite, un appel à la même fonction utilisée par -name sera réalisé, afin de comparer le contenu du fichier avec la chaîne de caractère regex recherchée.

2.1.7 -dir

Cette option permet de trouver un dossier dans une arborescence en fonction de son nom. Elle utilise la même fonction que -name en vérifiant toutes fois la bonne nature du résultat. Dans le cas où aucun paramètre n'est donné à cette option, elle se contentera d'afficher tous les dossiers depuis la racine.

2.2 Commandes optionnelles

2.2.1 -color

L'option -color permet d'afficher les erreurs et les noms de fichier en couleur. Cette option a été développée suivant la documentation suivante [6]. Lorsque -color est indiqué dans la commande :

— Les messages d'erreur seront écrits en rouge,
— Les fichiers seront colorés selon leur date de dernier accès si la recherche de fichiers se fait selon la date (-date est indiqué dans la commande). Sinon, les fichiers seront colorés selon leur type MIME.

Dates

Si la recherche des fichiers est effectuée sur les dates, voici la coloration effectuée :

Date de dernier accès	Couleur
1m	Cyan
10m	Bleu
1h	Violet
1j	Vert
7j	Jaune
Au-delà	Rouge

TABLE 5 – Couleurs en fonction de la date de dernier accès

Type MIME

Sinon, la coloration sera effectuée comme suit :

Type MIME	Couleur
application	Vert
audio	Violet
image	Jaune
text	Bleu
video	Violet
autre	Blanc

TABLE 6 – Couleurs en fonction du type MIME

2.2.2 -perm

Cette option permet de trouver un fichier dans une arborescence en fonction des permissions qui lui sont accordées (lecture, écriture, exécution pour l'utilisateur, le groupe, les autres). La libraire stat.h [1] a été utilisée, afin de connaître les permissions d'un fichier grâce au champ `st_mode`.

2.2.3 -ou

Cette option permet d'utiliser plusieurs options simultanément, à la manière d'un *ou* plutôt qu'un *et* comme c'est le cas pas défaut.

2.3 Autres

2.3.1 Parcours récursif de l'arborescence

C'est une des fonctionnalités principales, permettant de parcourir tous les éléments (fichiers et dossiers) d'un dossier. La bibliothèque `dirent.h` a été utilisée [2], et nous nous sommes également aidé du cours [5].

2.3.2 Gestion des erreurs

Les messages d'erreur ont été redirigés vers la sortie d'erreur standard `stderr`. La fonction `print_error` permet d'écrire une chaîne de caractères sur `stderr`, tout en la colorant en rouge.

3 Difficultées rencontrées

3.1 Option -mime

Je n'ai pas eu de problème lorsque le paramètre de -mime est de la forme `type/sous-type` : en effet, la fonction `getMegaMimeType` [4] récupère une chaîne de caractères contentant le type MIME au format `type/sous-type`, donc la comparaison était aisée.

En revanche, lorsque le paramètre de -mime est de la forme `type`, il faut tronquer le type MIME du fichier. Je souhaitais utiliser `strtok`, mais j'obtenais une erreur de segmentation.

Dans un premier temps, j'ai pensé à générer tous les types MIME possibles, étant donné le type "principal" (text par exemple) grâce à la fonction `getMegaMimeExtensions` [4]. Après avoir généré tous les types MIME "complets" (type et sous-type), on comparait alors le type MIME du fichier et le type MIME recherché. Or, cette méthode, bien que fonctionnelle en théorie, peut échouer s'il manque des types MIME dans la liste récupérée via `getMegaMimeExtensions` [4]. J'ai donc simplifié la recherche, en décidant de comparer seulement les types "principaux", afin de gagner du temps de calcul et de réduire la complexité du code.

Pour un fichier donné, j'ai parcouru son type MIME jusqu'à rencontrer un slash "/" : je pouvais alors obtenir uniquement le type. Par exemple, si la commande `ftc . -mime text` est saisie, je dois tronquer le type MIME de chaque fichier lors de la recherche récursive, puis comparer le type obtenu avec la chaîne de caractères `text`. Ainsi, la comparaison fonctionne.

3.2 Option -date

De manière générale, j'ai rencontré des petites difficultées lors de l'utilisation de certaines bibliothèques, notamment celle de `time.h`. Il m'aura fallu un peu de réflexion pour convertir correctement les valeurs données par la fonction `time()`, et également pour utiliser correctement les fonctions `strchr()` et `strcat()`. Autrement, je n'ai pas eu de difficulté particulière.

4 Volume horaire

Catégorie	**Tâche**	**Albert BECQUET**	**Lucas RIOUX**
Conception	Conception	2	2
Implémentation	*Commandes*		
	-test	1	1
	-name	2	-
	-size	-	4
	-date	4	-
	-mime	-	6
	-ctc	2	-
	-dir	3	-
	-color	3	5
	-perm	-	1
	-ou	1	-
	Autre		
	Parcours récursif	2	2
	Regex	2	-
	Gestion des erreurs	1	2
Rapport	Rédaction du rapport	3	3
Total	Total	26	26

5 Bibliographie

Références

[1] Christophe Blaess. Manuel du programmeur Linux. http://manpagesfr.free.fr/man/man2/stat.2.html.

[2] The Open Group. The Single UNIX, dirent.h. https://pubs.opengroup.org/onlinepubs/7908799/xsh/dirent.h.html.

[3] The Open Group. The Single UNIX, regex.h. https://pubs.opengroup.org/onlinepubs/7908799/xsh/regex.h.html.

[4] kobbyowen. Librairie MegaMimes. https://github.com/kobbyowen/MegaMimes.

[5] Lucas Nussbaum. Cours, Chapitre 3, Fichiers et entrées/sorties. https://members.loria.fr/LNussbaum/RS/CM-ch3.pdf.

[6] The Urban Penguin. Adding Color to Your Output From C. https://www.theurbanpenguin.com/4184-2/.

[7] Infini Software. La librairie time.h. https://koor.fr/C/ctime/ctime.wp.