Projet Programmation systèmes

Le but du projet est d'écrire un programme qui implémente une partie (très simplifiée) des fonctionnalités des commandes `find` et `grep`. Votre programme recherche les fichiers selon les critères spécifiés par les options et, éventuellement, effectue certaines opérations sur les fichiers trouvés.

L'appel

`recherche` *options*

lance votre programme, la recherche s'effectue à partir des répertoires donnés avec l'option `-c` (voir la description détaillée ci-dessous) . **La recherche s'étend sur toute l'arborescence des fichiers à partir du chemin indiqué.**

Il y a deux types d'options. Les options de *recherche* donnent les critères de recherche, les options *d'actions* indiquent l'opération à effectuer sur le fichier sélectionné (le fichier qui satisfait les critères de recherche).

Dans la suite le terme *fichier* signifie n'importe quel type de fichier : un fichier régulier, un répertoire ou un lien symbolique (on ignore l'existence de fichiers spéciaux). Le terme *nom de fichier* signifie toujours le nom de base, par exemple dans `/usr/dupont/Enseignement/examen.txt` le nom de fichier est `examen.txt`.

Critères de recherche

`-c` *chemin*	On effectue la recherche à partir de *chemin*. Au moins une option `-c` doit être spécifiée. Plusieurs options `-c` sont possibles si nous voulons faire la recherche dans plusieurs répertoires.
`-nom` *motif*	vrai si le nom de fichier contient le motif. Le motif est une chaîne de caractères, mais le caractère * au début ou à la fin du motif signifie "n'importe quelle suite de caractères".**Exemples :** • `-nom toto` recherche les fichiers dont le nom est `toto` • `-nom '*toto'` recherche les fichiers dont le nom se termine par `toto` • `-nom '*toto*'` recherche les fichiers dont le nom contient `toto` • `-nom 'toto*'` recherche les fichiers dont le nom commence par `toto`

`-type t`	vrai uniquement si le fichier est de type `t` où `t` est un des types suivants : • `d` pour un répertoire • `l` pour un lien symbolique • `f` pour un fichier régulier
`-u` *propriétaire*	vrai si *propriétaire* est le propriétaire du fichier. La structure `struct stat` utilisée avec les fonctions `lstat()` et `stat()` contient un champ pour l'identifiant du propriétaire (type entier `uid_t`) et le propriétaire peut être spécifié par ce numéro. **Exemple :** `recherche -u 1000` cherche les fichiers dont l'identifiant du propriétaire est 1000. **Optionnel:** On pourra faire aussi `-u nom` si on veut chercher les fichiers par le nom du propriétaire au lieu du numéro. Pour faire le lien entre l'identifiant du propriétaire et son nom regardez les pages man des fonctions `getpwuid()` et `getpwnam()`. L'implémentation de `-u nom` n'est pas obligatoire (mais `-u uid` est obligatoire).
`-texte` *motif*	vrai si le fichier régulier contient la chaîne de caractères *motif.* **Exemple :** `recherche -texte ' toto' -c /home/dupont` cherche les fichiers réguliers qui contiennent le caractère espace suivie de "toto".
`-vide`	vrai si le fichier régulier est vide (de longueur 0) ou le répertoire est vide
`-noprint`	par défaut on écrira sur la sortie standard les chemins complets vers les fichiers sélectionnés. `-noprint` indique de supprimer cette sortie.
`-d` *droits*	vrai si nous avons les droits spécifiés pour le fichier. Par exemple `recherche -d rx -c .` trouve les fichiers pour lesquels nous avons les droits de lecture et exécution. Pour vérifier si nous avons les droits demandés le programme utilisera la fonction `access()`, faire `man 2 access` pour la description de cette fonction.

S'il y a plusieurs options de recherche alors le fichier sera sélectionné s'il satisfait tous les critères de recherche. **Exemple :** `recherche -droits x -type f -c /home/toto` cherchera les fichiers réguliers pour lesquels nous avons le droit d'exécution, la recherche s'effectue à partir du répertoire `/home/toto`

Options pour les actions à effectuer

`-s`	supprimer les fichiers sélectionnés. Cela concerne aussi bien les fichiers réguliers (ou plus exactement les liens et les liens symboliques) à supprimer avec `unlink()`, que les répertoires, à supprimer avec `rmdir()`
`-a`	La même chose que l'option précédente mais si le fichier à supprimer est un répertoire qui n'est pas vide alors on essaie aussi de supprimer récursivement tous les sous-répertoires et fichiers qu'il contient.

`-e` *paramètres*	Cette option, si elle est utilisée, est toujours la dernière. Les *paramètres* sont tous les paramètres de `recherche` qui suivent `-e`. Si cette option est spécifiée et si le fichier trouvé est un fichier pour lequel on a le droit d'exécution (encore `access()` à utiliser) alors on exécutera le programme trouvé avec les *paramètres.* **Exemple :**
	`recherche -nom papa -noprint -c /home/toto -e 123 aaa bbb`
	recherche dans le répertoire `/home/toto` les fichiers nommés `papa` et pour chaque `papa` trouvé on lance l'exécution de :
	`papa 123 aaa bbb`

La recherche s'effectue toujours non seulement sur tous les répertoires donnés par `-c` mais aussi sur tous les sous-répertoires, sous-sous-répertoires etc.

Au lieu de faire des fonctions récursives on demande de créer des nouveaux processus à chaque fois qu'il y a un nouveau sous-répertoire à parcourir.

Par exemple si on fait

```
recherche -c . autres_options
```

et si le répertoire courant . contient un fichier régulier `aaa` et deux sous-répertoires `bbb` et `ccc` alors on vérifie si `aaa bbb ccc` satisfont les critères de recherche, et si c'est la cas on applique les actions et on lance deux nouveaux processus qui doivent faire la même chose pour les répertoires `bbb ccc` (qui à leur tour peuvent lancer d'autres processus).

Modalités

Le projet sera réalisé en binôme. Les dates de remise et de soutenances seront fixées ultérieurement.

Pour délivrer le projet on crée un répertoire avec le code source (fichiers `*.c` et `*.h`) et un `Makefile`.
`Makefile` doit permettre la compilation avec `make`, `make clean` devrait supprimer tous les fichiers `*.o` et l'exécutable.
Le fichier `alire.txt` doit donner les noms des auteurs et vous pouvez y mettre toute remarque à l'attention du jury de soutenances.
Dans les fichiers source il faut au moins commenter

- les fonctions en indiquant ce qu'elles sont censées faire et
- les variables globales (au niveau 0).

Le répertoire avec le projet doit être compressé sous forme `xxx_yyy.tar.gz`
où xxx et yyy sont vos noms et après la décompression cela doit donner le répertoire nommé `xxx_yyy` contenant votre projet. Le mail avec le projet est à envoyer aux chargés de TD (pour votre groupe) et au chargé de cours avec comme sujet : `projet systemes`.