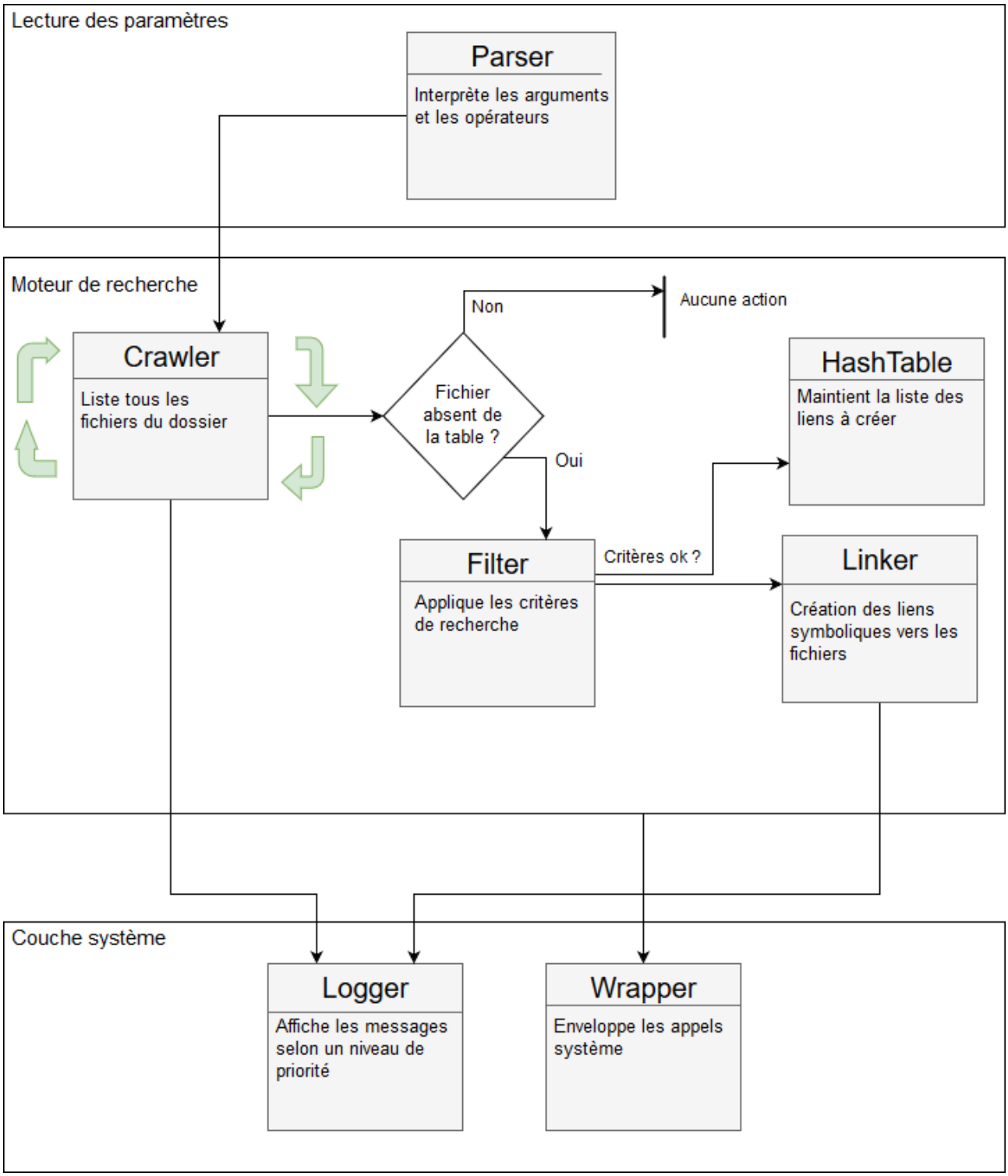
Smart Folder

Architecture

Nous avons décidé de diviser notre programme selon les modules suivants :



Lecture des paramètres

Parser

Le parser se chargera de lire les arguments donnés à notre programme. Pour ce faire il utilisera la fonction : `getopt_long_only`.

Il devra permettre d'analyser les options suivantes :

a) Nom du fichier correspondant exactement à une chaîne de caractères ou bien contenant une sous-chaîne de caractères ascii. `-name exactname` ou `-name_contain partOfName`

b) Taille de fichier plus grande, plus petite, ou égale à une valeur donnée. `-size [+/-/RIEN]taille[unité]` ou `unité = b(bytes), k(kilobytes), m(megabytes), g(gigabytes)`

c) Date de création/modification/utilisation du fichier plus ancienne, plus récente, ou égale à une valeur donnée. `-date [+/-/RIEN]YYYY-MM-JJ[c/m/u]`

d) Propriétaire ou groupe propriétaire égal ou différent d'une valeur donnée. `-owner[RIEN/NOT]]number[u/g]`

e) Droits d'accès du fichier correspondant à une valeur exacte donnée, ou contenant ou moins l'un des droits posés donnée ou tous les droits posés demandés. `-perm “rwxrwxrwx”` [USER:READ/WRITE/EXEC][GROUP:...][OTHER…]

f) Une expression booléenne combinant a) b) c) d) e) en implémentant par exemple un mécanisme similaire aux « opérateurs » de find(1). `-and -or -not`

Il sera possible de combiner plusieurs opérations avec un `-and` ou un `-or`. Exemple : `arg1 -and arg2 -or arg3 -and arg4` Il est également possible d'utiliser le contraire d'un argument en utilisant `-not` Exemple : `-not arg1` Le parser vérifie que l'arrangement des conditions et des opérateurs est correcte. Il stockera ces informations dans la structure de donnée qui est la suivante.

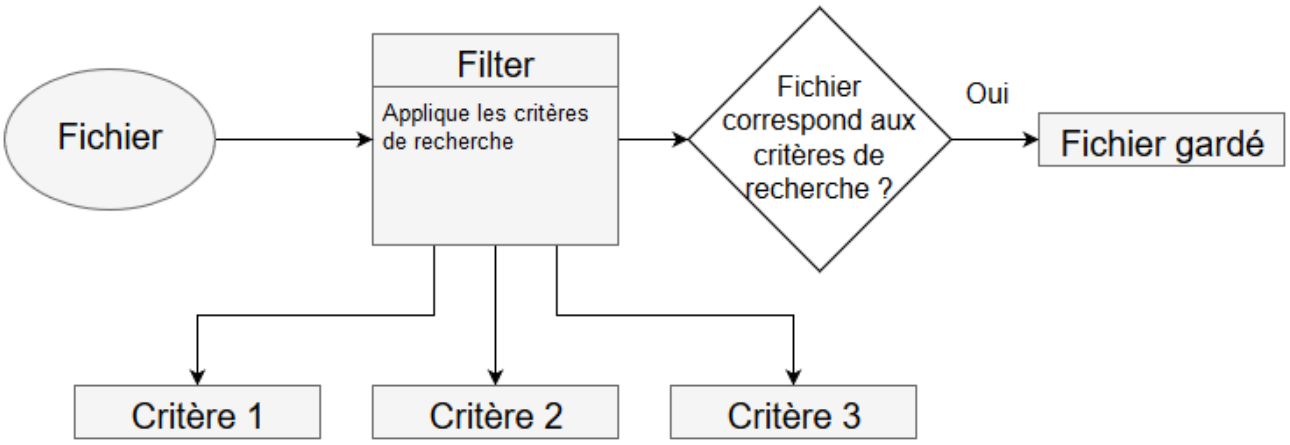
Moteur de recherche

Crawler

Finalement, nous avons fait un Crawler « maison » qui parcoure tous les sous-dossiers du dossier à scanner, en évitant les boucles infinies.

Filter

Le filtre teste chaque fichier et indique s'il correspond aux critères demandés. Si c'est le cas, il appelle le HashTable qui s'occupera de la suite des opérations. Il utilise la structure de donnée générée par le parser.



L'information sera représentée sous la forme d'un tableau avec les arguments et les opérateurs. Les arguments seront testés et le tableau sera parcouru afin de savoir si notre fichier passe le filtre en fonction des opérations.

HashTable

La HashTable est une table des fichiers qui correspondent aux critères. Elle est remplie par le filtre. Le linker est appelé uniquement si le fichier n'est pas présent dans cette table. Dans le cas où deux fichiers ont le même nom il n'y a pas de problèmes car ils sont discriminés par leur chemin complet.

Linker

Le rôle du Linker sera plus simple en comparaison des autres modules : il devra créer les liens symboliques (avec `int symlink (const char *oldname, const char *newname)`, `man 2 symlink`) des fichiers reçus du Filter et les stocker dans le search folder créé à cet effet.

Daemon

Toute cette partie Moteur de recherche devra tourner en tâche de fond grâce au fork. Une fois que le Parser aura terminé son travail, il n'aura plus de raison de s'exécuter.

Couche système

Logger

Le Logger sera utilisé pour tout ce qui est affichage sur les flux standards. Il pourra être déscativé au besoin. Il attendra comme arguments le niveau de priorité de messages qui doivent être affichés.

Wrapper

Ce module sert de "lien" entre les appels systèmes et les fonctions du programme. En effet, plutôt que d'appeler directement les fonctions système, nous ferons appel aux fonctions du module qui elles-mêmes appeleront les fonctions système. Ainsi, le programme pourrait être porté sur un autre système (il n'y aurait que le Wrapper à adapter pratiquement) et il gagne en modularité et lisibilité.