

Programmation système

SmartFolder

David Gonzalez - Claudio Sousa

23 décembre 2016

1 Introduction

Ce TP de deuxième année en programmation système consiste à implémenter un programme similaire au SmartFolder sur MacOSX.

Le SmartFolder sur MacOSX recherche sur le disque des fichiers correspondant à un/des critères et, pour chacun des fichiers trouvés, le programme crée un lien symbolique dans un dossier spécifié.

1.1 Spécification fonctionnelle

Ce programme possède deux modes de fonctionnement.

1.1.1 Mode recherche

Le premier est le mode *recherche*. C'est le mode par défaut et simule le SmartFolder sur MacOSX.

Dans ce mode, le programme tourne en arrière-plan et maintient dans le dossier de destination une liste de liens vers les fichiers trouvés correspondant aux critères de recherche choisis. Cette liste est dynamique et mise à jour si des nouveaux fichiers répondent aux critères de recherche ou, si au contraire, des fichiers ne répondent plus à ces critères.

La recherche de fichiers est recursive et suit les liens symboliques.

Ce mode prend 3 paramètres :

- `<dir_name>` : chemin où stocker les liens ;
- `<search_path>` : chemin de recherche ;
- `<expression>` : critères de sélection.

`<dir_name>` et `<search_path>` sont de simples chemins vers des dossiers.

`<expression>` correspond à une liste de critères dont l'interface est un sous-ensemble à celle de *find*.

Expression

Comme dit précédemment, l'interface est semblable à celle de *find*¹. Voici la liste des options pris en charge :

- `... : ...;`

Output/comportement

Si des liens symboliques existent, ils doivent être suivis. Des fichiers à doublet ne doivent pas apparaître dupliqués dans le dossier de destination.

1.1.2 Mode stop

Le deuxième est le mode *stop*. Il permet d'arrêter une recherche en cours.

Ce mode prend 1 paramètre :

- `-d <dir_name>` : termine le SmartFolder pour le chemin spécifié.

1. <https://linux.die.net/man/1/find>

2 Development

2.1 Architecture

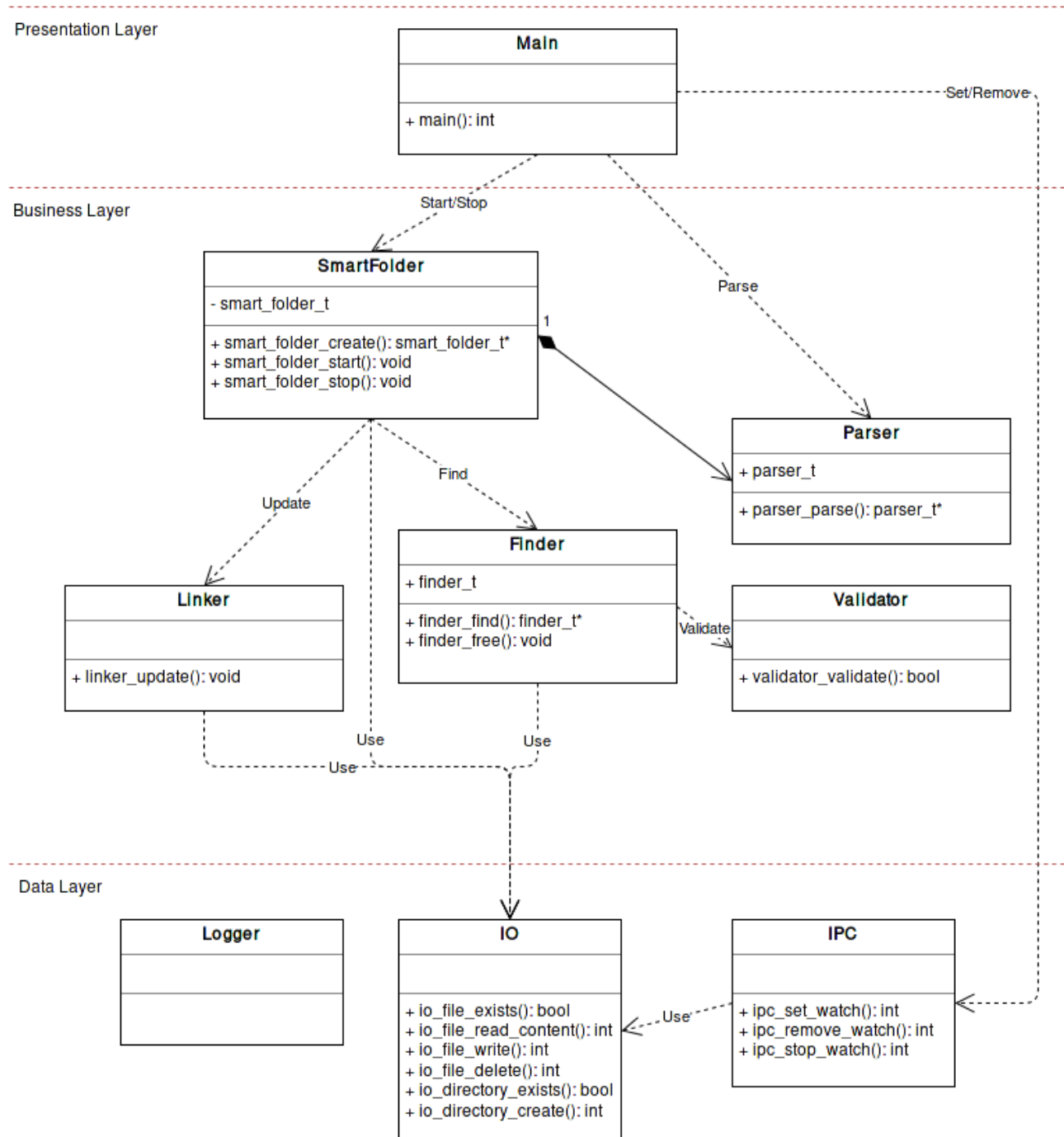


FIGURE 1 – Architecture du SmartFolder

2.1.1 Main

Le programme principal a pour rôle de vérifier les arguments et de sélectionner le bon mode de fonctionnement.

Dans le mode *recherche*, il a pour tâche de :

- met le processus en arrière-plan ;
- demande au module *Parser* de traiter l'expression ;
- initialise le module *IPC* ;

— initialise et lance le module *SmartFolder*.

Dans le mode *stop*, son seul rôle est de signaler l'arrêt à l'autre instance (voir IPC).

2.1.2 SmartFolder

SmartFolder est le module principal qui va orchestrer la recherche et la mise à jour du dossier de destination.

Lorsque lancé, il va continuellement utiliser le module *Finder* pour rechercher les fichiers correspondant au critère, puis donner la liste des fichiers retournée au module *Linker* pour qu'il mette à jour le répertoire de destination.

A noter qu'entre chaque recherche, il y a une pause de quelques secondes.

A l'arrêt, il est chargé de détruire le répertoire de destination et de libérer ses ressources.

2.1.3 Parser

Parser est le module qui transforme l'expression spécifiée dans la ligne de commande (critères de recherche) en une structure interne utilisable par le module *Validator*.

La complexité des différents opérateurs logiques possibles (priorité, parenthèses, opérateur unaire) doit être connue uniquement de ce module. La structure retournée doit pouvoir être traitée simplement par le module *Validator*.

Suggestion d'implémentation : utiliser l'algorithme de *shunting-yard*² pour le traitement de l'expression.

2.1.4 Validator

Ce module vérifie si un fichier est valide selon l'expression créée par le module *Parser*.

2.1.5 Finder

Ce module est responsable de produire la liste de tous les fichiers du dossier de recherche respectant les critères de recherche.

La vérification des fichiers contre les critères est déléguée au module *Validator*.

Lors du processus de recherche effectué au sein de ce module, un parcours d'arborescence est effectué récursivement et les liens symboliques sont suivis.

Des boucles deviennent alors possibles. Ce module doit donc gérer ce cas spécifique et prévenir les boucles infinies ainsi que le traitement des fichiers et dossiers dupliqués.

Possibilité d'implémentation : utilisation d'une hashtable contenant les noeuds d'index (inodes) des fichiers et répertoires déjà parcourus.

2.1.6 Linker

Le module *Linker* a pour rôle de mettre à jour le dossier de destination à l'aide d'une liste de fichiers passée en paramètre.

Pour chaque fichier, il crée un lien si celui-ci n'existe pas. Les liens qui ne sont plus valides sont effacés.

2. https://en.wikipedia.org/wiki/Shunting-yard_algorithm (par Edsger Dijkstra)

2.1.7 IPC

Ce module a pour but de répondre au problème du mode *stop*. En effet, lorsqu'une instance de *SmartFolder* souhaite arrêter une autre instance en cours d'exécution, il faut d'une manière ou d'une autre permettre une communication simple entre les deux processus afin qu'une instance puisse signaler un arrêt à une autre instance.

Le moyen de communication choisi a été les signaux POSIX, le signal *SIGTERM* pour être plus précis. Une instance qui veut donc signaler l'arrêt doit envoyer le signal cité au processus concerné.

Afin de pouvoir lancer un signal, il faut que le PID du processus cible soit connu. Pour cela, le PID d'une instance en mode *recherche* est stocké dans un fichier dans le répertoire utilisateur.

Comme une instance de *SmartFolder* est unique par dossier de destination, ce fichier se nommera d'après le chemin de ce répertoire.

2.1.8 IO

Le but de ce module est d'offrir une interface simple aux appels systèmes et de centraliser la gestion des erreurs.

2.1.9 Logger

Ce module centralise l'affichage des logs de débogage.