# rsfind – recherche multi-critères de fichiers

RS : Réseaux et Systèmes
Deuxième année

L'objectif de ce projet est de réaliser un outil de recherche multi-critères de fichiers.

## 1 Logistique

**Groupes.** Apprendre à travailler en groupe fait partie des objectifs de ce projet. Il vous est donc demandé de travailler en binôme (dont la composition est libre – il est possible de mélanger deux groupes de TD, et étudiants de TELECOM Nancy et étudiants de l'école des Mines). Il est **interdit** de travailler seul (sauf pour au plus un étudiant, si vous êtes un nombre impair). Un ingénieur travaille rarement seul.

**Tests automatiques.** Une part importante de l'évaluation du projet sera faite à l'aide d'une batterie de tests exécutés sous debian 9 Stretch. Il est donc très important, au cours de votre travail, d'accorder une large part à vos propres tests et de vérifier leur bonne exécution dans l'environnement d'évaluation (une machine virtuelle est disponible via vagrant sous le nom debian/stretch64 et vagrant peut être installé via apt-get sous linux ou homebrew sous mac os). Plusieurs «tests blancs» seront également organisés au cours du projet : votre code sera récupéré, compilé, et testé sur des cas de tests qui feront partie de ceux utilisés pour l'évaluation finale. Il est donc indispensable de commencer à travailler tôt pour bénéficier de ces «tests blancs».

**Rapport.** Vous devez rendre un mini-rapport de projet (5 pages maximum hors annexes et page de garde, format pdf). Vous y détaillerez vos choix de conception, les difficultés auxquelles vous avez été confronté, et comment vous les avez résolues. Vous indiquerez également le nombre d'heures passées sur les différentes étapes de ce projet (conception, implémentation, tests, rédaction du rapport) par membre du groupe.

**Soutenances.** Des soutenances pourront être organisées (éventuellement seulement pour certains groupes, par exemple s'il y a des doutes sur l'originalité du travail rendu). Vous devrez nous faire une démonstration de votre projet et être prêts à répondre à toutes les questions techniques sur l'implémentation de l'application.

**Plagiat et aide extérieure au binôme.** Si, pour réaliser le projet, vous utilisez des ressources externes, votre rapport doit les lister (en expliquant brièvement les informations que vous y avez obtenu). Un détecteur de plagiat [1] sera utilisé pour tester l'originalité de votre travail (en le comparant notamment aux projets rendus par les autres groupes). Toute triche sera sévèrement punie (jusqu'à un 0 au projet).

Concrètement, il n'est pas interdit de discuter du projet avec d'autres groupes, y compris de détails techniques. Mais il est interdit de partager, ou copier du code, ou encore de lire le code de quelqu'un d'autre pour s'en inspirer.

**Informations complémentaires.** Des informations complémentaires pourront être fournies sur `https://gitlab.telecomnancy.univ-lorraine.fr/nicolas.schnepf/rs2018/blob/master/README.md`. Ces informations complémentaires doivent être considérées comme faisant partie du sujet. Il est donc conseillé de surveiller cette page régulièrement.

**Questions.** Vos questions éventuelles peuvent être adressées à `nicolas.schnepf@inria.fr`. Les réponses (et les questions correspondantes) pourront être publiées sur la page du projet.

## 2 Description du projet

Le projet de cette année vise à développer un outil de recherche multi-critères de fichiers. Il pourrait être vu comme une combinaison (intégrée) des commandes `find`, `grep`, `ls` et `file` du shell.

Le programme à réaliser est décrit par sa page de manuel ci-dessous.

1. `http://theory.stanford.edu/~aiken/moss/`

RSFIND(1) Commandes RSFIND(1)

NOM

rsfind – Recherche multi-critères de fichiers

SYNOPSIS

rsfind [OPTION]... [CHEMIN]

DESCRIPTION

Recherche des fichiers dans CHEMIN (dans le répertoire courant par défaut) et ses sous-répertoires et affiche éventuellement des informations. Les fichiers trouvés sont affichés dans le même ordre que la commande find.

OPTIONS

Par défaut, rsfind affiche tous les fichiers, répertoires, liens symboliques, etc. trouvés. Il est possible d'utiliser rsfind pour n'afficher que les fichiers contenant une chaîne, portant un nom donné et/ou étant détectés comme des images par la bibliothèque libmagic.

Il est également possible de demander certains affichages en console tel que l'affichage des fichiers vérifiant les critères de recherche, le listing de leurs propriétés ou encore l'exécution d'une commande shell pouvant comporter des pipes.

Aucune option n'est obligatoire.

Si un chemin n'est pas précisé, le répertoire courant est utilisé.

-l
listing long (détaillé). Affichage des permissions, du propriétaire et du groupe, de la taille, de la date de création et de la cible des liens symboliques.

-t CHAINE
Restreint la sortie aux fichiers contenant la chaîne CHAINE.

-i
Restreint la sortie aux fichiers détectés comme des images par la bibliothèque libmagic (ceux dont le type MIME commence par image/).

--name CHAINE
Restreint la sortie aux fichiers nommés CHAINE.

--exec "CMD"
Exécute la commande CMD sur les résultats de la recherche ({} indique l'emplacement ou doit figurer le chemin relatif).

Les options --name, -i, et -t peuvent être combinées. Dans ce cas, les fichiers affichés sont ceux correspondant aux différents critères. C'est par exemple utile pour analyser des images vectorielles SVG, dont le format est basé sur XML.

De même, les options --print, -l et --exec peuvent être combinées. Dans ce cas la sortie en ligne de commande est l'accumulation des

```
       différentes sorties en ligne de commande dans l'ordre affichage /
       listing / exécution pour chacun des résultats de la recherche.

VALEUR DE RETOUR
       rsfind s'arrête en renvoyant la valeur de retour 0 si tous les
       traitements se sont déroulés avec succès. En cas d'erreur (répertoire
       inaccessible, fichier ne pouvant être analysé car non-lisible par
       l'utilisateur, etc.), rsfind s'arrête en renvoyant la valeur de
       retour 1.

EXEMPLES
       rsfind
            Liste les fichiers (et répertoires) trouvés dans le répertoire
            courant et ses sous-répertoires.

       rsfind -l /usr/bin
            Liste de manière détaillée le contenu du répertoire /usr/bin
            (et ses sous-répertoires)

       rsfind -l -t human -i /usr/share/openclipart/svg/
            Cherche les images contenant la chaîne de caractères "human"
            dans /usr/share/openclipart/svg/, et les affiche sous la forme
            d'un listing détaillé.

VOIR AUSSI
       find(1), file(1), grep(1), libmagic(3), ls(1), magic(5)
```

## 2.1 Conseils de réalisation

La quantité de code à produire est relativement faible (quelques centaines de lignes de code tout au plus). Mais le niveau de difficulté du code à produire est très important. Il est crucial de programmer de manière prudente, réfléchie, claire, en testant bien les différents cas d'erreur.

N'hésitez pas à utiliser `strace`, `gdb`, `valgrind`, etc. pour déboguer vos programmes.

L'implémentation correcte de l'intégralité des points de cette section vous permettra d'espérer une note de 12 au projet. Pour obtenir une note supérieure il vous est vivement recommandé d'implémenter une ou plusieurs extensions, voir la section suivante pour plus de détails sur ce point.

Il est conseillé de réaliser votre projet dans l'ordre qui suit.

**Étape 1 : analyse des options de la ligne de commande**
Il est possible de programmer manuellement l'analyse des options passées sur la ligne de commande, mais il est bien plus commode d'utiliser la fonction `getopt` ou `getopt_long` pour cela.

**Étape 2 : listing du contenu d'un répertoire** Il peut être intéressant d'implanter les options –name et –print ici de sorte à faire quelques tests.

**Étape 3 : listing des sous-répertoires**
Il peut être pratique d'utiliser de la récursivité.

**Étape 4 : listing détaillé (`-l`)**
Les champs à afficher sont :

- les permissions (comme dans `ls -l`, en gérant les cas des fichiers, répertoires, liens symboliques) ;
- le nombre de liens symboliques pointant vers le fichier ;
- le nom du propriétaire et le nom du groupe ;
- la taille (en octets, décimal) ;
- la date de création ;
- le nom .

**Étape 5 : recherche de texte (`-t`)**
Il faut chercher le texte passé en paramètre dans les fichiers (pas les liens symboliques), tout en continuant d'explorer les sous-répertoires. Seuls les fichiers contenant la chaîne sont listés. De sorte

à mettre cette question en perspective avec le cours il vous sera demandé d'implanter la lecture du fichier via la commande open(2), vous n'avez donc pas droit aux interfaces de plus haut niveau telles que fopen(3).

**Étape 6 : recherche d'images (`-i`)**

Il faut détecter les fichiers contenant une image. Regarder leur extension ne suffit pas, puisqu'on peut très bien changer l'extension du fichier. Une bonne manière de procéder est donc d'utiliser la bibliothèque `libmagic`, qui est fournie avec l'utilitaire `file`[2], et qui, à l'aide d'une base de données de *valeurs magiques*, permet d'identifier le contenu d'un fichier. Consultez la page de manuel `libmagic(3)`[3] pour plus de détails sur son utilisation. Une bonne manière de traiter son résultat est de demander le type MIME, et de vérifier qu'il commence par `image/`.

**Étape 7 : exécution de sous-commande**

L'exécution de sous-commandes vous permet de réaliser des opérations sur les fichiers résultant de la recherche de rsfind. Il s'agit de parser la commande reçue en argument de l'option –exec de sorte à extraire les éléments nécessaires à un appel à exec, en notant que l'utilisation des accolades {} sert à indiquer l'emplacement où doit être renseigné le nom du fichier pour lequel on veut exécuter le programme reçu en argument. Notez que le programme rsfind doit attendre la terminaison de son fils avant de continuer. Pour rendre cette question un peu plus intéressante que ce qui a été vu en TP il vous sera demandé de prendre en charge les pipes dans votre implantation de –exec, la commande find classique n'implante pas cet aspect mais elle présente un intérêt non négligeable dans le cas où l'on voudrait effectuer des traitements en chaîne sur les résultats d'une recherche de fichiers.

## 2.2 Extensions possibles

Les extensions réalisées seront valorisées (mais il faut d'abord réaliser correctement tous les points décrits ci-dessus). Elles doivent être décrites dans le rapport. Comme indiqué plus haut la partie obligatoire du projet sera notée sur 12, pour obtenir une meilleure note il faudra donc sérieusement songer à implémenter plusieurs des extensions présentées ci-dessous.

— Utilisation de `libdl` pour ne charger `libmagic` que si la bibliothèque est nécessaire, au lieu de se lier dynamiquer avec.

— Utilisation de `libpcre` pour rajouter la recherche d'expressions régulières *Perl* (en rajoutant une option -T pour la différencier de la recherche de chaînes).

— Autre utilisation des expressions régulières pour prendre en compte les méta-caractères shell dans la recherche de noms de fichiers (en rajoutant une option –ename pour la différencier de la recherche classique sur le nom de fichier).

— Mise en place de la parallélisation sur N threads via une option -p N permettant d'accélérer notamment la recherche d'images ou de textes. Notez que cette paralélisation ne doit pas perturber l'ordre dans lequel l'affichage est réalisé, une idée pour implanter cette extension consiste à commencer par rechercher les fichier, à les placer dans un tableau de chaînes de caractères et à démarrer l'analyse avec le nombre de threads autorisés par le paramètre N (pour préserver l'ordre de l'affichage il est important de garder un thread afficheur qui se charge d'afficher les sorties dans l'ordre correspondant).

Si vous souhaitez vous lancer dans une autre extension, n'hésitez pas à demander conseil à Nicolas Schnepf.

## 2.3 Exemple de sorties

Le comportement de la commande rsfind est défini par analogie avec le comportement de plusieurs commandes shell indiquées plus bas. Il est bien évidemment strictement interdit de récupérer lesdites commandes et de faire appel à exec pour les exécuter facilement dans votre programme, toute tentative de tricherie sera prise en compte lors de l'exécution des tests unitaires et sévèrement sanctionnée le cas échéant. Attention, le respect du format des sorties est crucial pour permettre le test de votre programme. Une sortie différente sera considérée comme fausse.

En particulier :

— Les entrées sont affichées dans l'ordre dans lequel le gestionnaire de fichiers les indexe

2. `http://www.darwinsys.com/file/`
3. `http://manpages.debian.net/cgi-bin/man.cgi?query=libmagic`

— Elles sont préfixées par le répertoire à analyser

— Les entrées pour '.' et '..' ne sont pas affichées

— En mode parallèle, les entrées restent affichées dans l'ordre

Seule la sortie standard (*stdout*) est utilisée pour l'évaluation. Il est donc utile d'envoyer vos messages de debug éventuels vers *stderr*.

— rsfind DOSSIER = find DOSSIER

— rsfind DOSSIER –name CHAINE = find DOSSIER -name CHAINE

— rsfind DOSSIER –print = find DOSSIER -print

— rsfind DOSSIER -t CHAINE = grep -r -l DOSSIER -e CHAINE

— rsfind DOSSIER -i = find DOSSIER -type f -exec file \; — grep image — cut -d : -f 1

— rsfind DOSSIER -l = find DOSSIER -exec ls -l {} \;

— rsfind DOSSIER –exec "COMMANDE" = find DOSSIER -exec COMMANDE \;

### 2.4 Règles du jeu

Seules les fonctions fournies par les bibliothèques suivantes peuvent être utilisées :

— bibliothèque C standard (libc) : `*dir`, `getopt`, pthreads, ...

— bibliothèque `libmagic`

Vous n'avez donc pas le droit d'utiliser de bibliothèque externe, de plus haut niveau, qui réaliserait une partie du travail à votre place. Si vous avez un doute sur le statut d'une fonction, posez la question.

## 3 « Rendu » du projet.

Le « rendu » du projet se fera via l'instance Gitlab de TELECOM Nancy. En dehors de la configuration correcte de votre dépôt, il n'y a rien à faire le jour de la fin du projet. Une page web (dont l'adresse sera communiqué sur le site du cours), et les tests blancs, vous permettront de vérifier la bonne configuration de votre dépôt.

Pour créer votre projet, il faut aller sur
`https://gitlab.telecomnancy.univ-lorraine.fr/projects/new` et choisir "Import project from" ..."Repo by URL", puis rentrer l'URL
`https://gitlab.telecomnancy.univ-lorraine.fr/Nicolas.Schnepf/rs2018`
Votre *Project name* doit commencer par `rs2018-` et contenir les noms des deux étudiants du binôme (par exemple `rs2018-dupont-durand`). Son *Visibility Level* doit être *Private*.
Une fois le projet créé, allez dans Settings, Members, et ajoutez *Nicolas.Schnepf* avec *Master* comme *role permission* (pour que le projet puisse être récupéré). Ajoutez également votre binôme.

Votre dépôt Git doit contenir, à la racine du dépôt :

— Un ficher `AUTHORS` listant les noms et prénoms des membres du groupe (une personne par ligne) ;

— Un `Makefile` compilant votre projet en créant un fichier exécutable nommé `rsfind` ;

— Un fichier `rapport.pdf` contenant votre rapport au format PDF.

Le dépôt `rs2018-template` que vous avez utilisé comme base contient un fichier `.gitlab-ci.yml` qui permet de lancer des tests à chaque fois que vous *poussez* des modifications sur Gitlab. Il est conseillé (mais pas demandé) d'ajouter vos propres tests pour vous assurer que votre projet ne régresse pas.

## 4 Calendrier

— Des «tests blancs» seront effectués à au moins trois reprises, aux alentours du 09/11/18, du 23/11/18, et du 07/12/18. Votre code sera récupéré, compilé, et testé sur des cas de tests qui feront partie de ceux utilisés pour l'évaluation finale. Les résultats vous seront communiqués, mais ne seront pas pris en compte pour l'évaluation finale. L'expérience montre que l'environnement de développement et d'exécution (architecture, système, version du compilateur ...) peut influencer les résultats et mettre en évidence des bugs qui pourraient ne pas être visibles sur vos machines. Il est donc très utile de commencer à travailler tôt pour bénéficier de ces «tests blancs». Aucune réclamation ne pourra être acceptée si votre programme ne se comporte pas correctement dans

l'environnement d'évaluation, puisque vous aurez pu bénéficier de plusieurs ≪tests blancs≫ avant le rendu du projet.
Seule la sortie *standard* sera utilisée pour comparer la sortie de votre programme avec celle attendue. Il est donc conseillé d'utiliser la sortie *erreur* pour afficher vos messages de débogage.

— La version finale de votre projet est à rendre pour le **vendredi 21/12/2018 à 23h55**. Il sera récupéré directement sur vos dépôts git. Vous n'avez donc pas d'action particulière à effectuer pour *rendre* le projet, mais vous devez vous assurer que les fichiers requis sont bien présents. Une bonne manière de vérifier que tous les fichiers sont bien présents sur le dépôt est de réaliser un nouveau *checkout* et d'en vérifier le contenu. Les groupes dont le projet ne pourra pas être récupéré correctement seront évidemment sanctionnés.