

# Introduction aux systèmes d'exploitation (IS1)

## TP n° 11 : pot-pourri

### Retour sur les filtres

Nous allons revenir sur de la manipulation de données à l'aide de filtres. Toutes les nouvelles commandes ainsi que les options nécessaires pour répondre aux exercices sont décrites à la fin de ce TP. Pour tout complément d'information, « man » est votre ami.







Vous trouverez sur moodle un fichier `ratp.csv`. Il provient des données publiques de la RATP (disponibles sur <https://data.ratp.fr/>) et contient des informations sur la fréquentation des diverses stations de son réseau ferré. Il s'agit d'un fichier texte au format CSV (pour *comma-separated values*), représentant un tableau à deux dimensions, les différentes colonnes étant séparées par un séparateur (usuellement une virgule, un point-virgule, une tabulation...)

### Exercice 1 – encore un peu de « *grep* »






1. Afficher la 1<sup>re</sup> ligne du fichier pour déterminer les entêtes de colonnes ainsi que le caractère séparateur utilisé.
2. Afficher uniquement les lignes concernant :
  - a. les stations parisiennes ;
  - b. les stations de banlieue ;
  - c. 🚩 les stations du 13<sup>e</sup> arrondissement ;
  - d. 🚩 les stations de la ligne 13 ;
  - e. 🚩 les stations de Sceaux.

### Exercice 2 – « *cut* », « *sort* », « *uniq* »

1. Afficher la liste des villes desservies
  - a. telle qu'elle apparaît dans le fichier ;
  - b. 🚩 triée dans l'ordre alphabétique (avec les doublons éventuels) ;
  - c. 🚩 triée dans l'ordre alphabétique, sans doublon, en majuscules ;

- d. en faisant précéder chaque ville du nombre de stations qui s'y trouvent ;
  - e.  triée par nombre de stations décroissant ;
  - f.  triée par nombre de stations décroissant, en séparant les arrondissements de Paris.
2.  Afficher la liste des stations
- a.  triée selon la fréquentation ;
  - b.  triée par ordre alphabétique, en affichant *seulement* les 10 stations les plus fréquentées ;
  - c.  triée dans l'ordre alphabétique, en n'affichant qu'une seule fois les stations de connexion Métro-RER.

### Exercice 3 – et maintenant tout ensemble

- 1.  Afficher *seulement* les stations de connexion Métro-RER.
- 2.  Afficher *seulement* les stations de connexion entre 3 lignes de métro (au moins).
- 3.  Afficher *seulement* les stations de connexion entre *exactement* 2 lignes de métro.
- 4.  Déterminer quel arrondissement possède le plus de stations de la ligne 7.
- 5.  Afficher, sur une seule ligne, la liste ordonnée des lignes qui desservent le 13<sup>e</sup> arrondissement.

### Travailler sur une arborescence avec « find »

La commande « find » permet de rechercher des fichiers et éventuellement d'exécuter une action dessus.

« find *[options]* chemins *[expression]* » parcourt les arborescences données par *chemins* en évaluant l'*expression* pour chaque fichier rencontré.

L'expression est constituée de tests et d'actions, tous ces éléments étant séparés par des opérateurs logiques. Quand un opérateur est manquant, l'opération par défaut -and est appliquée.

Par exemple, pour trouver les fichiers dont le nom contient toto dans l'arborescence de racine le répertoire personnel de l'utilisateur, on utilisera (à l'aide des mêmes jokers que le shell) :

```
find ~/ -name '*toto*'
```

## Les principales actions possibles

« -print » (action par défaut) liste les fichiers correspondant à la demande.

« -delete » supprime chaque fichier correspondant à la demande.

« -exec **cmd** {} ; » exécute *cmd* sur *chaque* fichier trouvé.

« -exec **cmd** {} + » exécute *cmd* sur *l'ensemble* des fichiers trouvés.

Les fichiers concernés sont désignés par « {} ». Par exemple, pour connaître le numéro d'i-nœud de tous les fichiers contenant le motif toto de son arborescence, on peut écrire :

```
find ~/ -name '*toto*' -exec ls -i {\} \;
```

Vous noterez qu'il est nécessaire d'échapper les caractères spéciaux « { », « } » et « ; ». Pour cela, on aurait aussi pu écrire :

```
find ~/ -name '*toto*' -exec ls -i '{}' ';' ;
```






## Exercice 4 – recherche par nom

- Lister tous les fichiers de votre arborescence
  - d'extension . java ;
  - dont le nom est de longueur 6 ;
  - dont le nom possède au moins 2 « . » ;
  - dont le chemin absolu possède au moins 2 « . ».
- Un usage excessif d'« emacs » provoque un trop-plein de fichiers inutiles se terminant par ~. Écrire une ligne de commande permettant de supprimer tous ces fichiers. *(Il est fortement recommandé de tester votre ligne de commande sur une sous-arborescence construite pour l'occasion).*
- Ôter le droit en exécution pour les utilisateurs autres que le propriétaire sur tous les fichiers d'extension . sh de votre arborescence personnelle.
- Créer une archive contenant tous les fichiers d'extension . java. de votre arborescence personnelle.

## Exercice 5 – autres critères de recherche


Pour résoudre les questions qui suivent, feuilleter le manuel de « find ». Il conviendra de créer le cas échéant des fichiers adhoc pour que la recherche trouve quelque chose...

- Lister tous vos liens symboliques.
- Lister tous vos répertoires.


3.  Lister tous vos répertoires vides.
4.  Lister tous vos fichiers de plus de 10 mégaoctets.
5. Lister tous vos répertoires ayant strictement plus que 5 liens.
6. Lister tous vos fichiers ordinaires ayant 2 ou 3 liens, en précisant leur numéro d'inœud.
7.  Lister tous vos liens vers un inœud donné (pour un des numéros obtenus à la question précédente).
8.  Lister tous vos fichiers sur lesquels vous avez tous les droits, tandis que votre groupe et les autres n'en ont aucun.
9.  Lister tous les fichiers qui sont chez vous mais qui ne vous appartiennent pas. L'affichage se fera avec « `ls -l` ».

### Exercice 6 – *pot pourri* : vérifier le nom des fichiers de réponse

Lors des précédents TP, certains parmi vous n'ont pas respecté la consigne concernant le nom du fichier à rendre... Normalement, vous devriez avoir dans chacun de vos répertoires TP<sub>i</sub> un fichier `reponses_TPi.txt`, ce que nous allons vérifier.

1. () Écrire un script « `check_name.sh` » prenant la référence d'un répertoire en paramètre, et affichant son paramètre sur la sortie standard, suivi du message...
  - "répertoire non concerné" si le nom de base de ce répertoire n'est pas de la forme TP<sub>i</sub> (avec *i* entier),
  - "OK" sinon, si le répertoire contient un fichier de nom `reponses_TPi.txt`,
  - "fichier de réponse introuvable, nom à vérifier" sinon ; dans ce cas, le script listera ensuite tous les fichiers ordinaires du répertoire d'extension `.txt`, ou sans extension.

La commande « `basename` » pourra être utile.

2. () À l'aide de « `find` », écrire une ligne de commande pour exécuter le script « `check_name.sh` » sur tous les répertoires de nom TP<sub>i</sub> de votre arborescence IS1 (et seulement eux).

**Les commandes** En principe, au cours de ce TP on a été amené à utiliser les commandes :

« `cut [-f champs] [-d caractere]` »

Les lignes de l'entrée standard sont vues comme des champs (*field* en anglais), délimités par le caractère TAB (`'\t'`) par défaut, ou le caractère indiqué par `-d`. La commande affiche sur la sortie les champs indiqués par `-f`.

« **grep** [-E] [-n] [-r] [-v] *expression* [références] »

permet de sélectionner toutes les lignes contenant le motif décrit par *expression*. L'option -v inverse la sélection, -n permet d'afficher les numéros des lignes sélectionnées, et -r permet de chercher récursivement dans tous les fichiers d'un répertoire. Sur certains systèmes l'option -E est nécessaire pour utiliser les opérateurs ? et +.

« **sort** [-n] [-r] »

trie les lignes, par ordre alphabétique par défaut, par ordre numérique avec -n, et inverse l'ordre avec -r (il est également possible de trier sur une autre colonne que la première avec les options -k et éventuellement -t, mais reportez-vous au man pour plus de détails).

« **tr** [-d] [-s] *chaîne1* [*chaîne2*] »

sans options, l'entrée standard est copiée sur la sortie standard après substitution des caractères spécifiés : un caractère ayant une occurrence dans *chaîne1* est remplacé par le caractère de même position dans *chaîne2*. Des chaînes décrivant des intervalles peuvent également être utilisées, comme A-Z, a-z, etc.

Avec une seule chaîne, l'option -d supprime (*delete*), dans son entrée, les caractères contenus dans la chaîne ; l'option -s (*squeeze*) supprime les caractères consécutifs (très utile pour les espaces par exemple : `tr -s ' '`).

« **uniq** [-c] »

recopie l'entrée standard en supprimant les lignes identiques consécutives (elle doit donc généralement être utilisée combinée avec `sort`). Avec l'option -c, elle précise le nombre de doublons qu'elle a comptés.

« **wc** »

compte le nombre de lignes, mots et caractères des fichiers ou de l'entrée standard.