

Introduction aux systèmes d'exploitation (IS1)

TP n° 9 : révisions, filtres « grep » et « cut »

Ce TP prolonge le TP n° 8 sur l'utilisation de tubes et de filtres pour manipuler des données. Pour cela nous aurons besoin de quelques nouvelles commandes, qui sont décrites à la fin de ce TP, avec certaines options utiles. Pour tout complément d'information, « man » est votre ami.

Tubes et filtres

La commande « grep »

« grep » (*global regular expression print*) sans option, permet de sélectionner les lignes d'un fichier qui contiennent un motif passé en premier paramètre. Par défaut (si aucun fichier n'est indiqué en deuxième paramètre), « grep » lit ses données sur l'entrée standard.

Le motif passé en paramètre à « grep » est une chaîne de caractères pouvant contenir des *caractères spéciaux* ; si certains de ces caractères sont les mêmes que les *jokers* utilisés par le shell pour manipuler des ensembles de noms de fichiers, leur signification n'est pas la même, et leur pouvoir expressif est bien plus grand.

Quelques caractères spéciaux de « grep »

- « ^ » et « \$ » représentent respectivement le début et la fin de ligne ;
- « . » représente un caractère quelconque ;
- une chaîne délimitée par les caractères « [» et «] » représente un caractère quelconque dans l'ensemble représenté (par liste exhaustive, intervalle(s) ou complément) ;
- les caractères « ? », « * » et « + » sont des *opérateurs unaires postfixes* permettant de représenter des chaînes formées à partir du caractère qui les précède. Par exemple,
 - « a? » représente *au plus un* caractère a,
 - « [a-z]* » représente une chaîne formée uniquement de minuscules,
 - « [^a-z]* » représente une chaîne qui ne contient aucune lettre minuscule,
 - « .+ » représente une chaîne d'*au moins un* caractère.

Attention, certains caractères spéciaux ne sont pas considérés comme tels par défaut (« ? » et « + » notamment). Pour qu'ils acquièrent leur signification particulière, il faut soit utiliser l'option « -E » de « grep » (permettant d'exprimer des *extended regular expressions*), soit les faire précéder d'un « \ ».

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 – « *grep* » et la recherche de motifs

1. Afficher la 1^{re} ligne du fichier pour déterminer les entêtes de colonnes ainsi que le caractère séparateur utilisé. Vous pouvez ensuite afficher par exemple les 10 premières lignes pour voir leur format.
2. Donner les commandes qui permettent d'afficher uniquement les lignes concernant :
 - a. 🚩 les stations qui prennent leur nom d'une mairie ; 🚩 chercher ensuite quelle option de « *grep* » permet de compter les lignes sélectionnées, puis compter les stations affichées par la commande précédente ; 🚩 obtenir le même résultat *sans* utiliser l'option précédente.
 - b. 🚩 les stations parisiennes ;
 - c. 🚩 la (seule) station qui se trouve à Paris sans arrondissement spécifié ;
 - d. 🚩 les stations de banlieue ; pour cela, chercher quelle option de « *grep* » permet d'inverser la sélection des lignes ;
 - e. 🚩 les stations qui desservent l'une des portes de Paris ; attention aux noms de gares qui contiennent le mot PORTE mais ne correspondent pas à des portes de la ville de Paris, c'est-à-dire les stations où PORTE est le premier mot ;
 - f. 🚩 les stations du 12^e arrondissement ;
 - g. 🚩 les stations de la ligne 12 ;
 - h. 🚩 les stations de la ville de Sceaux ; attention, il n'y en a que deux, même si trois lignes du fichier contiennent le mot Sceaux ; par ailleurs le fichier contient des erreurs de casse, et malheureusement, sans option « *grep* » est sensible à la casse...
 - i. 🚩 les stations dont le nom commence par 'M' ; attention, assurez-vous de ne pas sélectionner aussi les stations dans une ville dont le nom commence par 'M'. Idéalement, utilisez un seul « *grep* » dans votre réponse.
 - j. 🚩 la ou les stations qui contiennent un nombre dans leur nom.

La commande « cut »

La commande « cut » est en quelque sorte complémentaire de « grep » : si « grep » permet de sélectionner/filtrer des lignes, « cut » permet de sélectionner/filtrer des colonnes dans des fichiers de type CSV – ou, par extension, pour tout autre fichier texte.

En indiquant à « cut » quel caractère joue un rôle de séparateur (*delimiter* en anglais), « cut » considère tout ce qui se trouve entre deux de ces séparateurs comme un champ (*field* en anglais). On peut accéder à ces champs par leur numéro (le premier porte le numéro 1), ce qui permet d'en sélectionner un ou plusieurs, comme s'il s'agissait des colonnes d'un tableau.

Exercice 2 – « cut », mais aussi « sort » et « uniq »

On reprend le fichier `ratp.csv`.






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 (une option de « sort » qui ne soit pas sensible à la casse peut être utile) ;
- c. 🚩 triée dans l'ordre alphabétique, sans doublon (pas même les doublons "Sceaux" et "sceaux") ;
- 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 (en ordre croissant) ;
- 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. (Indice : ce sont les stations dont le nom contient "(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 de métro qui desservent le 13^e arrondissement.

Pour aller plus loin : manipuler des textes ligne à ligne avec « sed »

La commande « sed » (*stream editor*) sert à effectuer des modifications sur chaque ligne lue, ou seulement sur certaines ; elle lit par défaut sur l'entrée standard mais peut lire dans un fichier donné en argument. Il s'agit d'une commande certes un peu difficile à maîtriser, mais très puissante – en particulier, elle peut remplacer « tr », « cut », « grep »...

« sed -e *act1* ... -e *actn* » applique les actions *act1* .. *actn*.
Si on ne souhaite effectuer qu'une commande, l'option -e est facultative.

« sed -f *fic* » applique les actions listées dans le fichier *fic*.

Chaque action est formée, optionnellement, d'une description des lignes à traiter, puis d'une commande à exécuter sur ces lignes.

sélection des lignes à traiter

- « *n* » : un numéro de ligne
- « *m,n* » : un intervalle de numéros de lignes
- « */expr/* » : les lignes contenant le motif *expr* (décrit avec la même syntaxe que pour « grep »)

Les expressions peuvent utiliser les mêmes opérateurs que « grep », et il est donc conseillé de mettre les actions entre guillemets simples pour éviter les expansions à mauvais escient.

Quelques commandes

sélection de lignes grâce aux commandes « p » (print) et « d » (delete) ; noter l'option « -n » pour éviter de recopier chaque ligne sur la sortie.
« sed -n -e '*/expr/p*' » équivaut à « grep *expr* »
« sed -e '*/expr/d*' » équivaut à « grep -v *expr* »

translitérations et substitutions grâce aux commandes « y » et « s » (substitution), suivies de la description de la modification.

« sed -e 'y/chaîne1/chaîne2/' » équivaut à « tr chaîne1 chaîne2 »

« sed -e 's/expr/rplc/' » remplace la première occurrence du motif *expr* par l'expression *rplc*.

« sed -e 's/expr/rplc/g' » remplace toutes les occurrences du motif.

Noter que l'utilisation du caractère / comme séparateur n'est pas imposée; on peut utiliser le caractère que l'on souhaite (et en particulier, en choisir un qui n'apparaît pas dans les expressions manipulées).

Vous pouvez vous référer à ce site pour plus d'exemples, ou au manuel complet pour une description exhaustive des possibilités de « sed ».

Exercice 4 – remplacements de base

Télécharger le fichier `lettre.txt` sur Moodle.

1. Afficher le contenu de `lettre.txt` après remplacement des chaînes « \ 'e » par le caractère « é ».
2. À l'aide de l'option « -i » de « sed », corriger tous les accents de `lettre.txt`.
3. Dans le fichier `ratp.csv`, corriger toutes les erreurs de casse dans les noms de ville (il ne devrait plus y avoir de lettres minuscules au début d'un champ).
Indication : dans l'expression de remplacement, \U& permet de remplacer toutes les lettres minuscules du motif par la majuscule correspondante.

Les blocs La commande « s » permet non seulement de faire des substitutions, mais aussi de manipuler des blocs de texte : pour sélectionner un bloc de texte dans l'expression d'origine, il suffit de le placer entre \ (et \) ; les blocs ainsi définis peuvent être rappelés avec les expressions \1, \2, etc ..., où le numéro correspond à l'ordre d'apparition. Par exemple, la commande

```
sed -e 's/\([a-zA-Z]*\) \([a-zA-Z]*\)/\2 \1/'
```

inverse les deux premiers mots de chaque ligne.

Exercice 5 – mise en forme de texte

On reprend le fichier `ratp.csv`.

1. Afficher les noms des stations triées par ville, sous la forme :

Antony : ANTONY

Antony : PARC DE SCEAUX

2. Afficher la description des stations d'une ville de votre choix de manière plus verbale, sous la forme :

Station : LA MOTTE-PICQUET-GRENELLE

Ville : Paris

Correspondances : 6 8 10

Les commandes En principe, au cours de ce TP sont utilisées les commandes :

« cut [-d *caractere*] » [-f *champs*]

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] [-i] [-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, -i permet d'ignorer la casse 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 +.

« sed [-f *cmdfile*] [-n] [-i] [-e *act1* ...-e *actn*] *file* »

permet de modifier ligne à ligne l'entrée standard, ou un fichier *file*. La commande applique les actions *act1* ... *actn*, où chaque action est précédée de l'option -e ; et les actions contenues dans le fichier *cmdfile*. L'option -n permet d'éviter de recopier chaque ligne sur la sortie. L'option -i permet de modifier un fichier en place. L'action '/*expr*/p' équivaut à `grep expr`, l'action '/*expr*/d' équivaut à `grep -v expr`, l'action 'y/*chaîne1*/*chaîne2*/' équivaut à `tr chaîne1 chaîne2`, l'action 's/*expr*/*rplc*/' remplace la première occurrence du motif *expr* par l'expression *rplc*, l'action 's/*expr*/*rplc*/g' remplace toutes les occurrences du motif, l'action '/*expr*/a\←*texte*' ajoute *texte* après chaque ligne contenant le motif, tandis que l'action '/*expr*/i\←*texte*' insère le texte avant la ligne contenant le motif. (← représente un retour à la ligne.)

« sort [-n] [-r] [-k] [-t] »

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.