

## Introduction aux systèmes d'exploitation (IS1)

### TP n° 11 : Retour sur les filtres, « cut » et « sed »

#### 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.

La commande « cut » en particulier permet d'afficher des zones spécifiques d'un fichier et donc des manipuler des parties du fichier comme si elles étaient des champs.

Vous trouverez sur moodle un fichier `liste.csv`. Il contient des informations sur des étudiants (fictifs) : leur prénom, leur groupe, un jour de semaine où ils sont disponibles, la ville où ils habitent. Ces informations sont séparées par des caractères « : ».

#### Exercice 1 – « cut », « sort », « uniq »

1. Afficher le contenu du fichier `liste.csv`.
2. ➤ Afficher la liste des prénoms (et *uniquement* les prénoms).
3. Afficher la liste des villes (*uniquement* les villes) :
  - a. telle qu'elle apparaît dans le fichier ;
  - b. ➤ triée dans l'ordre alphabétique (avec les doublons éventuels) ;
  - c. ➤ triée dans l'ordre alphabétique, en n'affichant qu'une seule fois chaque ville (y compris Paris) ;
  - d. en faisant précéder chaque ville de son nombre d'occurrences dans le fichier ;
  - e. ➤ triée par nombre d'occurrences décroissant (avant de trier le résultat de la commande précédente, vous pouvez avoir besoin de « `tr -s ' '` »).

#### Exercice 2 – les filtres sur des commandes du système

Les commandes du système comme « `ls -l` » ou « `ps -l` » ajoutent des espaces pour rendre l'affichage plus lisible, ce qui peut être gênant si on veut considérer l'espace comme un séparateur.

Vous pourriez donc être amenés à utiliser le filtre « `tr -s ' '` » (qui transforme une suite d'espaces en un seul) pour utiliser « cut » sur la sortie d'une commande qui ajoute beaucoup d'espaces.

1. Afficher la liste des identifiants de processus en cours d'exécution sur votre ordinateur (et seulement les identifiants).
2. La liste précédente est longue. Pour s'assurer que l'on ne s'est pas trompé de colonne, on souhaite n'afficher que la première ligne. Ajouter un dernier filtre pour cela.
3. ➤ Afficher la liste des identifiants de processus apparaissant dans la colonne PPID de la commande « `ps -lax` », précédés chacun de son nombre de fils (éventuellement triée par nombre de fils décroissant).

### Exercice 3 – un peu plus de « *grep* »

Vous avez déjà utilisé la commande « *grep* » comme un filtre au TP précédent. « *grep* » peut également prendre en paramètre une ou plusieurs références de fichiers (ordinaires ou répertoires) sur lesquels effectuer la recherche.

1. ➤ Déterminer la liste des fichiers de votre arborescence personnelle contenant le mot `static`, et le numéro de ligne où ce mot apparaît.
2. ➤ Créer un fichier contenant les mêmes informations que `liste.csv` mais dans lequel toutes les lignes contenant le motif "GR1" ont été supprimées.
3. ➤ Afficher les informations concernant les étudiants dont le prénom commence par un 'L'.
4. ➤ Afficher les informations concernant les étudiants dont le prénom commence par un 'L' disponibles le mardi.
5. ➤ Afficher les informations concernant les étudiants dont le prénom commence par un 'N' qui ne sont pas disponibles le mardi.
6. ➤ Écrire une commande qui compte le nombre de lignes commençant par 'A' et se terminant par 't' dans `liste.csv`.
7. ➤ Écrire une commande qui compte le nombre de lignes commençant par 'B' et se terminant par '.' dans `liste.csv`.
8. ➤ Afficher le prénom et la ville concernant les étudiants dont le prénom se termine par un 's'.
9. ➤ Afficher le prénom, le groupe et la ville des étudiants dont la ville se termine par un 's'.
10. ➤ Afficher le prénom et le groupe des étudiants disponibles le mardi. Le prénom sera affiché en majuscule.
11. ➤ Afficher, sur une seule ligne, les numéros des lignes contenant le mot "mardi".

## La commande « sed »

La commande « sed » (*stream editor*) sert à effectuer des modifications sur chaque ligne lue (dans un fichier ou sur l'entrée standard).

« sed -e *cmd1* ... -e *cmdn* » applique les commandes *cmd1* ... *cmdn*.  
Si on ne souhaite effectuer qu'une commande, l'option -e est facultative.

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

« sed -e '/*expr*/d' » équivaut à « grep -v *expr* » (motif décrit avec la même syntaxe que pour « grep »).

« 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.

« sed -e '/*expr*/a\*unpeudetexte*' » ajoute *unpeudetexte* après chaque ligne contenant le motif.

« sed -e '/*expr*/i\*unpeudetexte*' » insère *unpeudetexte* avant chaque ligne contenant le motif.

Noter que l'utilisation du caractère / comme séparateur dans « sed » n'est pas imposée ; on peut utiliser le caractère que l'on souhaite.

### Exercice 4 – commenter du code java

1. Écrire un script `comment.sh` qui prend en argument un fichier Java et qui commente toutes les lignes du fichier, c'est-à-dire qui rajoute la chaîne `//` au début de chaque ligne du fichier. Testez votre script.
2. Écrire un script `uncomment.sh` qui prend en argument un fichier java et enlève les commentaires faits avec `//` en début de ligne. Testez votre script.

### Exercice 5 – 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`.

**Exercice 6** – manipulation d’affichages en java

1. Écrire un script `unprint.sh` qui prend en argument un fichier `.java` et qui l’affiche à l’écran en enlevant les lignes dans lesquelles est fait un appel à la fonction `System.out.println`. Testez votre script.
2. Écrire un script `signalPrint.sh` qui prend en argument un fichier `.java` et qui écrit la phrase `// appel a println` avant chaque appel à la fonction `System.out.println`. Testez votre script.
3. À l’aide de la commande « `p` » et l’option « `-n` » de « `sed` », afficher toutes les lignes d’un fichier `.java` qui contiennent un appel à `System.out.println` (simulation de « `grep` » par « `sed` »).

**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 7** – plan de site internet

La commande « `curl` », avec en paramètre une adresse internet, affiche le code source de la page à l’adresse indiquée.

1. Écrire un script `titre.sh` qui prend en argument une adresse internet, et qui affiche le titre de la page (il est entouré des balises `<title>` et `</title>`).
2. Écrire un script `plan.sh` qui prend en argument une adresse internet, affiche le titre de la page à l’adresse donnée ainsi que le texte contenu entre les balises `<h1>` et `</h1>`, et `<h2>` et `</h2>`.

**Exercice 8** – mise en forme de texte

On rappelle que le fichier `liste.csv` (sur Moodle) contient une liste de nom, groupe, jour et ville, séparés par des doubles-points.

1. Afficher les noms des étudiants du groupe 1 triés par ville, sous la forme :

```
Bagnolet : Livio
Bagnolet : Thibault
Montreuil : Alice
```

2. Afficher le contenu du fichier de manière lisible, sous la forme :

```
Etudiant : Yago
Groupe : GR1
date : lundi
ville : Paris
```

**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 `+`.

« `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 ensembles peuvent également être utilisées, comme `[A-Z]`, `[a-z]`, etc. (ici, les crochets doivent être écrits).

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 ' '`).

« `wc` »

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

« `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).

« `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.