Introduction aux systèmes d'exploitation (IS1)

TP 6 – filtres et tubes, suite

Filtres

Vous trouverez sous Didel un fichier `liste`. 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 `':'`. Enregistrez le contenu de ce fichier dans un fichier `fic:txt`, les exercices qui suivent concerneront ce fichier.

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

1. Afficher le contenu du fichier `fic:txt`.
2. Afficher la liste des prénoms telle qu'elle apparaît dans le fichier (*uniquement* les prénoms).
3. Afficher la liste des villes telle qu'elle apparaît dans le fichier (*uniquement* les villes).
4. Afficher la liste des villes, triées dans l'ordre alphabétique (avec les doublons éventuels).
5. Afficher la liste des villes, triées dans l'ordre alphabétique, en n'affichant qu'une seule fois chaque ville (y compris Paris).
6. Afficher la liste des villes, chacune étant précédée de son nombre d'occurrences dans le fichier.
7. Même question que précédemment, mais la liste doit maintenant être triée par nombre d'occurrences décroissantes. 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. La commande « `tr -s ' '` » est donc souvent indispensable pour utiliser « `cut` » sur la sortie d'une telle commande.

1. Afficher la liste des identifiants de processus en cours d'exécution sur votre ordinateur (et seulement les identifiants).
2. Si 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 nombres de fils décroissants).

Exercice 3 – *« grep »*

1. Créer un fichier de nom `Z` contenant les mêmes informations que `fic:txt` mais dans lequel toutes les lignes contenant le motif "GR1" ont été supprimées.
2. Afficher le prénom et le groupe des étudiants disponibles le mardi. Le prénom sera affiché en majuscule.
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 `fic:txt`.
7. Écrire une commande qui compte le nombre de lignes commençant par `'B'` et se terminant par `'.'` dans `fic:txt`.
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'`.

Tubes nommés

Les tubes créés par l'opérateur | sont ce que l'on appelle des *tubes anonymes*. Ce sont des sortes de fichiers "tampons" créés directement par le shell pour une commande donnée. Il est possible de créer manuellement des tubes en leur donnant un nom. Cela permet ensuite de les utiliser dans plusieurs commandes. Ces tubes sont ce que l'on appelle des *tubes nommés*.

La commande « `mkfifo` » permet de créer un tube nommé. Plus précisément, `mkfifo nom` crée un tube de lien `nom`.

Exercice 4 – *Création de tubes*

1. Ouvrez deux terminaux, dans lesquels vous vous placerez dans le même répertoire. On voudrait utiliser le premier terminal pour écrire dans un fichier et le second pour lire le contenu au fur et mesure.

   Dans le premier terminal, vous commencerez à créer, au moyen de la commande « `cat` » et d'une redirection de sa sortie standard, un fichier `fic` dont le contenu sera ce que vous tapez au clavier, mais sans terminer encore la saisie par . Dans le deuxième terminal, afficher le contenu du fichier `fic`.
2. Rajoutez deux lignes de saisie dans le premier terminal. Bien entendu, ces deux lignes ne s'affichent pas dans le deuxième ; pour les voir, on doit afficher à nouveau tout le contenu (essayez). Terminez la saisie par dans le premier terminal.
3. Créez un fichier nommé `tube1` avec la commande « `mkfifo` ». Regardez ses caractéristiques (droits, taille, type de fichier, etc.), que constatez-vous ?
4. Essayez d'ouvrir ce fichier avec un éditeur, que constatez-vous ? Dans la suite, toutes les lectures et écritures dans les différents fichiers se feront à l'aide de la commande « `cat` ».
5. Depuis le premier terminal, écrivez dans ce fichier sans terminer la commande.
6. Depuis le deuxième terminal, lisez le fichier `tube1`. Que constatez-vous ? Essayez d'écrire autre chose dans le premier terminal. Que se passe-t-il ?
7. Que constatez-vous lorsque vous terminez la commande qui écrit dans le tube ?
8. On appelle généralement *écrivain* le programme chargé d'écrire dans le tube et *lecteur* celui qui lit son contenu. Peut-on avoir plusieurs écrivains pour un lecteur ? Essayez avec au moins deux écrivains.
9. Que se passe-t-il si on a plusieurs lecteurs ? Essayez avec un seul écrivain et au moins deux lecteurs. Qu'en déduisez-vous sur le fonctionnement des tubes nommés ?

Exercice 5 – *Messagerie instantanée*

Le but de cet exercice est de reproduire à l'aide de tubes nommés un système de messagerie instantanée comme celui fourni par la commande `talk`.

1. Ouvrez deux terminaux, et faites en sorte que ce que vous écrivez dans le premier soit recopié dans le deuxième.
2. Recommencez en lançant la lecture du tube en arrière-plan.
3. Faites en sorte que la communication fonctionne dans les deux sens. Attentin, si vous faites plusieurs essais, pensez à tuer les processus inutiles.
4. Utilisez la commande « `tee` » pour enregistrer la conversation dans un fichier `historique`.

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 par défaut, ou le caractère indiqué par `-d`. La commande affiche sur la sortie les champs indiqués par `-f`.

« `grep [-v]` ***expression*** **[***fichiers***]** **»**
permet de selectionner toutes les lignes contenant le motif décrit par *expression*. L'option `-v` inverse la sélection.

« `tr [-d] [-s]` ***chaine1*** **[***chaine2***]** **»**
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 cartactères 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.