Pratique des machines
TP4 : Manipulation de texte

am@up8.edu

Octobre 2022

Dans ce TP :
— Liste des processus et **kill**
— Construction d'une hiérarchie de fichiers propre ;
— Manipulation de fichiers grâce aux commandes cut, less, nl, wc, join, paste, sort, comm, shuff, tr ;
— Expansion de commandes.

Avant de commencer :
— créez un répertoire TP4 dans le répertoire dédié à ce cours. À la fin du TP vous ferez une archive de votre répertoire TP4 grâce à la commande :

```
$ zip -r TP4_VOTRENOM.zip TP4
```

Avant de partir :
— déposez l'archive TP4.zip sur le moodle.

Exercice 0 : lister les processus en cours sur votre machine

La commande **ps** (pour *Process Status* permet de lister la liste des processus actifs.

1. lancez la commande **ps** dans le terminal. 4 colonnes s'affichent :
   — PID : Process ID, l'identifiant du processus
   — TTY : TeleTYpewriter, l'identifiant du terminal
   — TIME : temps d'utilisation cumulé du CPU pour un processus donné
   — CMD : le nom de l'exécutable
2. lancez de nouveaux processus à partir du terminal déjà ouvert (par exemple firefox, mousepad, etc.) et d'un autre terminal et constatez l'évolution des différents champs (TTY notamment).
3. pour constater l'évolution de TIME, exécutez les commandes suivantes **une par une** dans le terminal :

```
$ ps
$ while [ 1 ]; do x=1; done # ceci est une boucle infinie, laissez
    la tourner quelques secondes
```

Ctrl-C # stoppe le processus en cours (donc la boucle)

```
$ ps
```

4. D'autres processus qui n'ont pas été lancés par ce terminal tournent sur votre machine, vous pouvez les afficher grâce à l'option **-x**. Ils sont très nombreux. Pour vous y retrouver, vous pouvez utiliser la commande **grep** afin de filtrer les résultats en n'affichant que les lignes qui *matchent* une certaine chaîne de caractères. Ouvrez plusieurs terminaux et essayez par exemple :

```
$ ps -x | grep sh
```

5. **kill** un processus : on peut, grâce à la commande **kill**, envoyer un signal d'arrêt à un processus en cours, grâce à la commande :

```
$ kill PID
```

   Avec PID le numéro de processus accessible grâce à la commande **ps** vue ci-dessus. Cette commande est notamment pratique si vous lancez un programme que vous ne savez pas comment arrêter (par exemple si le programme tourne en boucle).

   Tentez de « kill » un processus dont vous n'avez pas besoin (par exemple une instance de mousepad). Par défaut, le signal envoyé est SIGTERM (SIGnal de TERMinaison), il engendre la fin du processus mais peut être intercepté et ignoré dans certains cas. Trouvez l'option qui permet de tuer un processus en évitant toute interception.

1 Exercice 1 : manipulation d'une base de données textuelle

1. Placez vous dans le répertoire TP4 ;
2. Créez un fichier TP4_exercice1.sh dans lequel vous consignez toutes les commandes (fonctionnelles) que vous utilisez. À la fin de la séance, le script TP4_exercice1.sh doit vous permettre de reproduire les commandes de réponses aux différentes questions. Si vous oubliez certaines commandes, vous pouvez utiliser l'historique qui se trouve dans le fichier ∼**.bash_history**.
3. Téléchargez le fichier que vous trouverez à l'adresse `https://www.data.gouv.fr/fr/datasets/r/b363e051-9649-4879-ae78-71ef227d0cc5` grâce à la commande **wget** ;
4. Renommez le fichier téléchargé et donnez lui le nom : **base_de_donnees.csv** ;
5. Exécutez la commande suivante pour vous faire une idée de ce que contient le fichier. Vous pouvez naviguer dans le fichier « page par page » en utilisant les touches « f » (*forward*, vers l'avant) et « b » (*backward*, vers l'arrière) et tapez « q » lorsque vous avez fini :

```
$ less base_de_donnees.csv
```

6. Utilisez la commande **head** (la « tête ») et ses options pour récupérer uniquement la première ligne de ce fichier, et stockez-la dans un fichier nommé **en_tete.csv** (vous devez utilisez la redirection avec l'opérateur **>**) ;

Recommandations

(a) de manière générale, n'utilisez pas de caractères accentués, ni d'espaces, ni de caractères spéciaux (par exemple « [») dans vos noms de fichiers ou de répertoires ;

(b) vérifiez toujours l'extension des fichiers et répertoires que vous créez ou manipulez.

7. Grâce à la commande **tail** (la « queue »), récupérez toutes les lignes sauf la première et stockez-les dans un fichier nommé **corps.csv** (en utilisant la redirection). Vous aurez sans doute besoin de connaître le nombre de lignes contenues dans le fichier au préalable.

8. Commande **comm**
   — En utilisant la fonction **sort**, classez les entrées de **corps.txt** par ordre de numéro de département, et trouvez un moyen de ne récupérer que les lignes concernant les académies dont le numéro de département est inférieur à 30, stockez les dans un fichier **dep_inf_30.csv**.
   — Extrayez les entrées des académies en zone A et stockez-les dans un fichier **zone_A.csv**
   — exécutez la commande

```
$ comm dep_inf_30.csv zone_A.csv
```

   comm produit une comparaison des deux fichiers passés en paramètres et renvoie sur trois colonnes : les lignes présentes dans le premier fichier seulement, les lignes présentes dans le second fichier seulement, et les lignes présentes dans les dexu fichiers. Exécutez :

```
$ comm -1 -3 dep_inf_30.csv zone_A.csv
```

   Les options **-1** et **-3** permettent de n'afficher que les lignes présentes *seulement dans le second fichier*. Cela fonctionne-t-il ?
   Consultez la documentation de **comm** et faites les modifications nécessaires afin de stocker dans un fichier *les lignes des académies dont le département est inférieur à 30 ne se trouvant pas en zone A*.

9. En utilisant la commande **cut** et à la commande **tr**, extrayez la liste des toponymes (villes, îles, agglomérations) présents dans la première colonne et classez-les par ordre alphabétiques. Supprimez les doublons et stockez la liste des toponymes dans un fichier **toponymes.txt**.

10. Extrayez de **corps.csv** la liste des lignes qui contiennent au moins un champ vide et stockez les dans un fichier dont le nom vous semble approprié.

11. commande **paste** :
    — En utilisant **echo** et la redirection (**>**), créez en une instruction un fichier **zones.txt** contenant le texte suivant :

```
Zone A
Zone B
Zone C
```

    — en utilisant 3 fois **grep** et **wc**, ainsi que les opérateurs **>** et **»** stockez dans un fichier **compte.txt** le nombre d'entrées correspondant à chaque zone.

— utilisez une commande et la redirection (**>**) pour créer un nouveau fichier **zone_compte.txt** dans lequel les zones et le nombre d'académies concernées apparaissent *côte à côte*. Regarder les documentations des commandes **join** et **paste** afin de choisir la plus adaptée.

2 Exercice 2 : Permissions

Commencez par créer un fichier exercice2.txt dans lequel vous consignerez les réponses aux questions de cet exercice.

Nous avons vu la semaine dernière que tous les fichiers et dossiers n'ont pas les mêmes permissions (lecture, écriture, exécution).

2.1 Protéger son espace personnel

1. Quelles sont les permissions sur votre dossier personnel (∼) ?
2. Quelles sont les permissions sur le dossier personnel de votre voisin ? Pouvez-vous créer un fichier **test_VOTRENOM.txt** dans le dossier personnel de votre voisin ?
3. Essayez depuis l'explorateur de fichiers de changer les paramètres de votre dossier personnel pour interdire l'écriture aux autres *users*. Regardez si cela a eu un effet.
4. Proposez une commande `bash` permettant de faire ce changement de permissions.

2.2 Conversion code octal - ligne de permissions en lettres

1. Pour chaque ligne de permission en lettres, donnez la permission en code équivalente.
   (a) rwxr-x–x
   (b) rw-r—–
   (c) -wx–x-w-
2. Pour chaque permission en code octal, donnez la permission en lettres
   (a) 775
   (b) 123
   (c) 871