Filtrage et premiers scripts

Exercice 1, Rechercher ou modifier des informations dans un fichier texte :

Téléchargez le fichier `se.txt.tar` sur la page web du cours, puis désarchivez-le. Dans toutes les questions suivantes, on vous demande de chercher de l'information dans ce fichier. Les questions suivantes se font bien avec `grep`, en ajoutant éventuellement un petit `wc`.

1) Affichez toutes les lignes qui contiennent le mot « `ordinateur` ».

SOLUTION. `grep -e 'ordinateur' se.txt`

2) Affichez toutes les lignes qui terminent par le mot « `ordinateur` ».

SOLUTION. `grep -e 'ordinateur$' se.txt`

3) Affichez les lignes qui contiennent l'un des mots « `ordinateur` », « `programme` » ou « `logiciel` ».

SOLUTION. Avec `grep -e`, on peut faire :
`grep -e '\(logiciel\|programme\|ordinateur\)' se.txt`
Attention, avec `egrep` ou `grep -E`, il ne faut pas échapper les parenthèses et le `|` :
`grep -E '(logiciel|programme|ordinateur)' se.txt`

4) Comptez le nombre de lignes vides que contient le fichier.

SOLUTION. On peut faire `grep -c -e '^$' se.txt` ou bien `grep -e '^$' se.txt | wc -l`

5) Afficher toutes les lignes sauf les lignes vides.

SOLUTION. Le plus simple est sûrement d'inverser la sélection avec `-v` : `grep -v -E '^$' se.txt`

6) Afficher toutes les lignes qui commencent par une majuscule.

SOLUTION. `grep -e '^[A-Z]' se.txt`

7) Afficher toutes les lignes non-vides qui ne commencent pas par une majuscule (donner deux méthodes).

SOLUTION. Toutes les lignes qui commencent par autre chose qu'une majuscules :
`grep -e '^[^A-Z]' se.txt`
On peut aussi le faire avec `-v`, mais il faut faire attention aux lignes vides :
`grep -v -e '\(^[A-Z]\|^$\)' se.txt`
Attention au fait que, si on utilise `-E` au lieu de `-e`, il ne faut pas échapper les parenthèses et le `|`.
Même chose avec `egrep` :
`egrep -v -e '(^[A-Z]|^$)' se.txt`

8) Afficher toutes les lignes qui contiennent un nombre, puis celles qui contiennent un nombre entre crochets, puis celles qui commencent par un nombre entre crochets.

SOLUTION. Cela donne :
`grep -e '[0-9][0-9]*' se.txt`
`grep -e '[[0-9][0-9]*]' se.txt`
`grep -e '[[0-9][0-9]*]' se.txt`

9) Consultez la page de manuel pour `grep`, et interessez-vous à l'option `-o`. Donnez ensuite une commande pour compter le nombre d'occurrences du mot « `ordinateur` ».

SOLUTION. `grep -o -e 'ordinateur' se.txt | wc -l`

Les questions suivantes impliquent la commande `sed`. Elles font toujours référence au même fichier.

1) Affichez le fichier sur la sortie standard, en utilisant `sed` de façon à remplacer toutes les occurrences de « `logiciels1` » par « `logiciels [1]` ». Une fois que vous-vous êtes assuré que le remplacement est bien fait, modifiez votre commande pour que `sed` modifie le fichier en place.

SOLUTION. `sed -e 's/logiciels1/logiciels [1]/g' se.txt`
`sed -i -e 's/logiciels1/logiciels [1]/g' se.txt`

2) Affichez le fichier en supprimant toutes les occurrences du mot « `applicatifs` » après le mot « `logiciels` ».

SOLUTION. La mini astuce, c'est qu'il suffit de substituer `logiciels` à `logiciels applicatifs` :
`sed -e 's/logiciels applicatifs/logiciels/g' se.txt`
(avec ou sans le `-i`...)

3) Affichez le fichier en remplaçant toutes les occurrences de « `Windows` et « `Mac OS` » par `XXX` ».

SOLUTION. `sed -e 's/\(Windows\|Mac OS\)/XXX/g' se.txt`
Notons que l'on peut aussi faire `sed -E`, mais avec la même remarque que pour `grep` :
`sed -E 's/(Windows|Mac OS)/XXX/g' se.txt`
Notons aussi que, si on ne sait pas comment faire la substitution des deux mots en une seule commande, on peut toujours les enchaîner :
`cat se.txt | sed -e 's/Windows/XXX/g' | sed -e 's/Mac OS/XXX/g'`

4) Affichez le fichier en remplaçant toutes les références de la forme « `[i]` (avec `i` un entier) par « `*** i ***` ».

SOLUTION. Il faut utiliser les échappements, de façon à ce que `sed` comprenne que `[` et `]` ne sont pas des caractères spéciaux :
`sed -e 's/\[\([0-9]\)\]/*** \1 ***/g' se.txt`
Avec `-E`, ça marche aussi, mais il faut enlever les échappements pour les parenthèses :
`sed -E 's/\[([0-9])\]/*** \1 ***/g' se.txt`

Exercice 2, Premier script shell :

1) Sur la ligne de commande, on peut entrer plusieurs commandes qui s'exécuteront les unes à la suite des autres : il suffit de la séparer par un `;`. Entrez par exemple la commande :
`a="Toto"; echo "$a fait du vélo !"`
Quel est l'affichage produit ?

2) Entrez maintenant sur la ligne de commande :
`read a; echo "$a fait du vélo !"`
Logiquement, vous devez rester bloqué sur une ligne vide : entrez un nom. Quel est l'affichage produit ?

3) Avec votre éditeur de texte préféré, créez un fichier texte `tst.sh` contenant les lignes suivantes :

```
#!/bin/sh

a="toto"
echo "$a fait du vélo !"
```

Il s'agit d'un script Shell, dans lequel chaque ligne non-vide (sauf la première, qui indique quel shell doit être utilisé à l'exécution) est une commande : à l'exécution du script, les commandes seront exécutées les unes à la suite des autres. Une fois votre fichier enregistré, rendez-le exécutable avec la commande `chmod +x tst.sh` ; ensuite, lancez le avec `./tst.sh` et observez le résultat.

4) Modifiez le script précédent, de façon à ce qu'il affiche « `Quel est votre nom ?` ». Si vous entrez « `Toto` », il devra répondre en affichant « `Bonjour Toto !` ».

Exercice 3, Un deuxième script shell :

Le fichier `/proc/cpuinfo` est mis à jour par le noyau Linux, et contient des informations sur votre processeur. Affichez le contenu de ce fichier avec `cat`. Vous pouvez observer que :

— vous avez des informations détaillées pour chacun des cœurs du processeur.
— les lignes `processor` donnent le numéro de chaque cœur en partant de 0 (si k est le numéro du dernier cœur à s'afficher, alors $k+1$ est le nombre de cœurs).
— les lignes `model name` donnent à chaque fois le modèle du processeur.
— les lignes `cpu MHz` donnent la fréquence de chaque cœur : on va dire que la fréquence du dernier cœur est celle du processeur.

Ecrivez un script `mycpu.sh` pour afficher le nombre de cœurs du processeur, le modèle du processeur, et sa fréquence. Par exemple, cela pourra donner :

```
nlouvet@LIP-E6220:~/TP$ ./mycpu.sh
Votre processeur comporte 4 coeurs
Son modèle est : "Intel(R) Core(TM) i3-2310M CPU @ 2.10GHz"
Actuellement, la fréquence est de 1400 MHz
```

Pour aboutir à une solution, commencez pas faire des tests en ligne de commande. Vous pouvez utiliser `grep`, `cut`, `sed`, `echo`, `expr` et les variables du shell.

SOLUTION. Voici une solution faite ne vitesse...Il y a sûrement plus élégant, mais ce n'est pas ce qui est demandé !

```
#!/bin/sh

n=$(grep 'processor' /proc/cpuinfo | tail -1 | cut -d':' -f2 | sed 's/ //g')
n=$(expr $n + 1)

m=$(grep 'model' /proc/cpuinfo | tail -1 | cut -d':' -f2 | sed 's/ //')

f=$(grep 'cpu MHz' /proc/cpuinfo | tail -1 | cut -d':' -f2 | sed -e 's/ *\([0-9][0-9]*\)\.[0-9]*/\1/g')

echo "Votre processeur comporte $n coeurs"
echo "Son modèle est : \"$m\""
echo "Actuellement, la fréquence est de $f MHz"
```

Le même, sans `cut`, et en prenant en compte qu'il pourrait y avoir des espaces/tabulations après le « : » :

```
#!/bin/sh

file="/proc/cpuinfo"

n=$(grep 'processor' $file | tail -1 | sed -e 's/^.*:[[:space:]]*\([0-9][0-9]*\)/\1/g')
n=$(expr $n + 1)

m=$(grep 'model name' $file | tail -1 | sed -e 's/^.*:[[:space:]]*\(.*\)/\1/g')

f=$(grep 'cpu MHz' $file | tail -1 | sed -e 's/^.*:[[:space:]]*\([0-9][0-9]*\)\.[0-9]*/\1/g')

echo "Votre processeur comporte $n coeurs"
echo "Son modèle est : \"$m\""
echo "Actuellement, la fréquence est de $f MHz"
```