Les bases du système Linux - TP séance 2 - SOLUTIONS

Gabriel Moreau
Gabriel.Moreau@univ-grenoble-alpes.fr

Frédéric Audra
frederic.audra@univ-grenoble-alpes.fr

Novembre 2018

1 Manipulation de fichiers texte, tubes et redirections

1. Créer un fichier RecetteCrepes.txt avec la commande **cat** et dont le contenu est le suivant :

```
cat <<FIN >RecetteCrepes.txt
> 1:250 g de farine.
> 2:4 oeufs.
> 3:0,5 l de lait.
> 5:50 g de beurre.
> 6:1 sachet de sucre vanillé
> 4:1 pincée de sel.
> FIN
more RecetteCrepes.txt
less RecetteCrepes.txt
cat RecetteCrepes.txt
```

2. Compter les lignes du fichier RecetteCrepes.txt. `man wc`

```
$ wc -l
```

3. Trier dans l'ordre des numéros (`man sort`) et envoyer le résultat vers le nouveau fichier RecetteCrepes_sorted.txt (une seule commande).

```
$ sort -n RecetteCrepes.txt > RecetteCrepes_sorted.txt
```

4. Supprimer les numéros du fichier RecetteCrepe_sorted.txt en début de ligne (supprimer aussi les « : »). Afficher le résultat directement sur le terminal. `man cut`

```
$ cut -f2 -d":" RecetteCrepes_sorted.txt
```

5. Ajouter une 7ème étape au fichier RecetteCrepes.txt (une ligne en fin de fichier) : « 7 :1 cuillère à soupe de rhum (5 cl) ». Utiliser les redirections.

```
$ echo "7:1 cuillère à soupe de rhum (5 cl)" >> RecetteCrepes.txt
```

6. Afficher sur le terminal toutes les lignes sauf celles contenant le mot « cuillère ». `man grep`

```
$ cat RecetteCrepes.txt | grep -v cuillère
$ grep -v cuillère RecetteCrepes.txt
```

7. Afficher les 3 dernières lignes du fichier RecetteCrepes.txt. `man tail`

```
$ tail -3 RecetteCrepes.txt
```

8. Exécuter la commande **cat** `RecetteTartes.txt`. Exécuter de nouveau cette commande en redirigeant les erreurs pour ne pas les afficher sur le terminal.

```
$ cat RecetteTartes.txt 2>/dev/null
```

9. Regrouper les dernières commandes (`tail` et **cat**) des deux dernières questions sur une seule ligne. Le résultat affiché sur le terminal ne doit pas contenir les erreurs.

```
$ (tail -3 RecetteCrepes.txt; cat RecetteTartes.txt) 2>/dev/null
$ tail -3 RecetteCrepes.txt; cat RecetteTartes.txt 2>/dev/null
```

2 Editeurs

1. Remplacer le mot « lait » par « bière » avec la commande **sed** dans le fichier fichier Recette-Crepes.txt. Créer en même temps le fichier RecetteCrepesBiere.txt

```
$ sed 's/lait/bière/g' RecetteCrepes.txt > RecetteCrepesBiere.txt
```

2. Remplacer le mot « bière » par le mot « lait » avec l'éditeur interactif `vi` dans le fichier `RecetteCrepesBiere.txt`.

```
$ vi RecetteCrepesBiere.txt
:%s/bière/lait/g
:wq
```

3. Remplacer avec **sed** les « `l` » (litres) par des « `dl` » dans la recette.

```
$ sed -e 's/l/dc/;' RecetteCrepes.txt   # Remplace que le premier l
$ sed -e 's/l/dc/g;' RecetteCrepes.txt  # Remplace tous les l
$ sed -e 's/ l / dc /g;' RecetteCrepes.txt
```

3 Premier script

1. Dans un terminal, créer votre premier script :

```
$ cat > mon_script.sh << FIN
echo "Bonjour, ceci est mon premier script"
echo "Arguments: $*"
FIN
```

2. Vous devriez obtenir un fichier `mon_script.sh`. Visualisez-le. Que remarquez-vous ? Le dollar a disparu !
3. Recommencez, avec cette fois-ci un « \ » pour protéger le « $ ».
4. Ajouter le sheband en début de script (avec un éditeur interactif). Rendre le script executable (`chmod`). Puis executer le script en donnant des arguments.

4 Second script

1. Créez un fichier « ingredients.txt » contenant le texte suivant, permettant de réaliser une recette bien de chez nous :

```
cat > ingredients.txt << EOF
patates:1:kg
lardons:200:g
oignons:200:g
reblochon:1
huile:2:cuillieres
ail:1:gousse
sel:1:pincee
poivre:1:pincee
EOF
```

2. Faites un script qui parcours ce fichier et affiche les ingrédients sous une forme humainement lisible : Le script utilisera le shell `/bin/bash` et devra s'interrompre en cas d'erreur. Prenez garde à bien traiter le cas particulier du reblochon qui n'a pas d'unité !

   Voici 3 solutions différentes et une solution dans le langage de script PERL.

Listing 1 – Solution 1

```
#!/bin/bash
set -e
```

```
for ingredient in $(cat ingredients.txt)
do
   nom=$(echo "${ingredient}"      | cut -f1 -d:)
   quantite=$(echo "${ingredient}" | cut -f2 -d:)
   unite=$(echo "${ingredient}"    | cut -f3 -d:)
   if [ "${unite}" = "" ]
   then
      echo "${quantite} ${nom}"
   else
      echo "${quantite} ${unite} de ${nom}"
   fi
done
```

Listing 2 – Solution 2

```
#!/bin/bash
set -e

cat ingredients.txt | while read line
do
   nom=$(echo "${line}"      | cut -f1 -d:)
   quantite=$(echo "${line}" | cut -f2 -d:)
   unite=$(echo "${line}"    | cut -f3 -d:)
   if [ "${unite}" = "" ]
   then
      echo "${quantite} ${nom}"
   else
      echo "${quantite} ${unite} de ${nom}"
   fi
done
```

Listing 3 – Solution 3

```
#!/bin/bash
set -e

IFS=: #IFS Internal Field Separator

while read nom quantite unite
do
   if [ "${unite}" = "" ]
   then
      echo "${quantite} ${nom}"
   else
      echo "${quantite} ${unite} de ${nom}"
   fi
done < ingredients.txt
```

Voici une solution avec (**awk**) qui est un langage de traitement de lignes, disponible sur la plupart des systèmes Unix. Il est principalement utilisé pour la manipulation de fichiers textuels pour des opérations de recherches, de remplacement et de transformations complexes (wikipedia). Nous pouvons écrire le même script en une seule ligne !

```
awk 'BEGIN{FS=":";} {if ($3 == "") print $2 " " $1; \
else print $2 " " $3 " de " $1;}' ingredients.txt
```

3. Modifiez votre script afin qu'il prenne un argument qui sera un multiplicateur pour la quantité d'ingrédients (par exemple, l'argument « 2 » multiplie par 2 toutes les quantités)

```
#!/bin/bash

set -e
```

```
multiplicateur=$1

for ingredient in $(cat ingredients.txt)
do
   nom=$(echo "${ingredient}"      | cut -f 1 -d ':')
   quantite=$(echo "${ingredient}" | cut -f 2 -d ':')
   unite=$(echo "${ingredient}"    | cut -f 3 -d ':')
   ((quantite=quantite*${multiplicateur}))
   #quantite=$(($(echo "${ingredient}"|cut -f 2 -d ':') * ${multiplicateur}))
   if [ "${unite}" = "" ]
   then
      echo "${quantite} ${nom}"
   else
      echo "${quantite} ${unite} de ${nom}"
   fi
done
```

4. Nous voulons tester si l'argument donné est bien numérique. Un moyen de faire est de réaliser une opération qui génère une erreur lorsque l'argument n'est pas numérique et nous allons traiter cette erreur. Proposez un tel test en début de script, afin d'afficher un message « L'argument doit être un nombre » en cas d'argument non numérique.

```
#!/bin/bash
multiplicateur=$1
let multiplicateur++ 2>/dev/null
if [ $? -ne 0 ] # astuce : si l'arguement n'est pas un nombre
then            # alors la commande d'incrementation retourne une erreur
   echo "Largument '$1' nest pas un nombre entier !"
   exit 1
fi

set -e

for ingredient in $(cat ingredients.txt)
do
   nom=$(echo "${ingredient}"      | cut -f 1 -d ':')
   quantite=$(echo "${ingredient}" | cut -f 2 -d ':')
   unite=$(echo "${ingredient}"    | cut -f 3 -d ':')
   ((quantite=quantite*$1))
   #quantite=$(($(echo "${ingredient}"|cut -f 2 -d ':')*$1))
   if [ "${unite}" = "" ]
   then
      echo "${quantite} ${nom}"
   else
      echo "${quantite} ${unite} de ${nom}"
   fi
done
```

Solution utilisant les tests avancés et une expression rationnelle.

```
#!/bin/bash

set -e

if [[ $1 =~ ^[1-9][0-9]*$ ]]
then
   multiplicateur=$1
else
   echo "Largument '$1' nest pas un nombre entier !"
   exit 1
fi

for ingredient in $(cat ingredients.txt)
do
```

```
    nom=$(echo "${ingredient}"      | cut -f 1 -d ':')
    quantite=$(echo "${ingredient}" | cut -f 2 -d ':')
    unite=$(echo "${ingredient}"    | cut -f 3 -d ':')
    ((quantite=quantite * multiplicateur))
    if [ "${unite}" = "" ]
    then
        echo "${quantite} ${nom}"
    else
        echo "${quantite} ${unite} de ${nom}"
    fi
done
```