Correction TP1 : LINUX ET SCRIPTS

Mini-exos

- **Afficher *"Hello !"*** :

```
#!/bin/bash
echo "Hello !"
```

- **Script qui execute *ls -aCF***

```
#!/bin/bash
ls -aCF $1
```

- **Invite l'utilisateur à donner son nom sur le standard input et initialise une variable shell : NAME. Ensuite affiche le contenu de cette variable**

```
#!/bin/bash
echo "Please give your name"
read NAME
echo "Your name is $NAME"
```

- **Affiche les variables d'environnements de l'utilisateur et du shell utilisé (*printenv* pour avoir la liste des variables d'environnement)**

```
#!/bin/bash
echo "$USER is using $SHELL"
```

- **Affiche les paramètres (arguments) du programme et le nom du programme. !! Bien mettre (( )) pour l'incrémentation de i (opération arithmétique) !!**

```
#!/bin/bash
echo "The script is : $0"
i=0
for val in $@; do
    ((i++))
    echo "The parameter $i is $val"
done
```

- **Affiche le texte entré malgrès les retour à la ligne (arrête dès qu'une ligne vide)**

```
#!/bin/bash
SAVE=""
echo "Give your text"
while read LINE; do
    if [ "$LINE" == "" ];
        then break
    else
        SAVE+=$LINE
    fi
done
echo "Your text is :"
echo $SAVE
```

Exo sauvegarde image et traitement erreurs

Ajouts :

- Demande quelle extension d'image le programme doit sauvegarder
- Vérifie qu'il y a bien un argument (et pas plus) d'indique, sinon affiche un message d'erreur dans le standard error output
- Vérifie qu'il y a bien un fichier avec l'extension à traiter, sinon message d'erreur standard error output **Attention**
- bien sortir dans le cas d'une erreur : `exit 1`
- indiquer que le message d'erreur va dans le standard error output : `echo "erreur ..." >&2`

```
#!/bin/bash
if [ $# -ne 1 ]
then
    echo "This program should have one argument" >&2
    exit 1
fi
DIR=original_images;
mkdir $DIR -p;
i=1;
nbok=0
EXTENSION=$1
nb=$(ls -l | grep "^-.*\.$EXTENSION$" | wc -l)
if [ $nb -eq 0 ]
then
    echo "$0 has no file to process" >&2
    exit 1
fi
echo "This program will backup and rename $nb $EXTENSION files"
for f in *.$EXTENSION
do
      cp $f $DIR
      mv $f $(printf "image_%0.3d.$EXTENSION" $i)
      mvreturn=$?
      if [ $mvreturn == 0 ]
      then
        ((nbok++))
      fi
      ((i++))
done
echo "$nbok files processed correctly"
```

Exercice final

- **Télécharger un/des fichier : *wget*, ensuite *-O* indique que tous les fichiers sont concatenés en un seul et *-O-* indique que ce fichier doit être écrit dans la standard output**

```
wget -O- https://www.cjoint.com/doc/23_03/MCBn305uIy7_sysmic-org-access.log.gz
```

- **Décompresser un fichier**

```
zcat MCBn305uIy7_sysmic-org-access.log.gz
```

- **Trier les adresses IP du ficher à partir en fonction du nombre d'accès (ordre décroissant) et les enregistre dans un nouveau fichier**

```
grep -E "^([0-9]+\.){3}[0-9]+" MCBn305uIy7_sysmic-org-access.log -o | sort | uniq -c | sort -n > resultat.txt
```

!! Utiliser *uniq -c* pour avoir le nombre de doublons.

!! Faire un *sort* avant *uniq* pour qu'il détecte tous les doublons (et pas que ceux qui se suivent).

!! 2ème sort :

- *-n* : indique qu'on trie des nombres, évite les problèmes avec des espaces (11 < ·8 par exemple).
- *-r* : indique l'ordre décroissant.

!! Enregistrer les résultats dans le fichier : `> resultat.txt`

- **Script final :**

```
#!/bin/bash
URL=https://www.cjoint.com/doc/23_03/MCBn305uIy7_sysmic-org-access.log.gz
nomFich=MCBn305uIy7_sysmic-org-access.log
wget $URL
zcat $nomFich.gz
grep -Eo  "^([0-9]+\.){3}[0-9]+" $nomFich | sort | uniq -c | sort -nr > resultat.txt
```

- **Même chose mais en *pipelining* et pas un script :** Juste une seule ligne avec plusieurs commandes séparées par `|`