NSY103 =➔ Linux : principes et programmation

Introduction aux scripts shell

Un script shell permet d'automatiser une série d'opérations. Il se présente sous la forme d'un fichier contenant une ou plusieurs commandes qui seront exécutées de manière séquentielle.

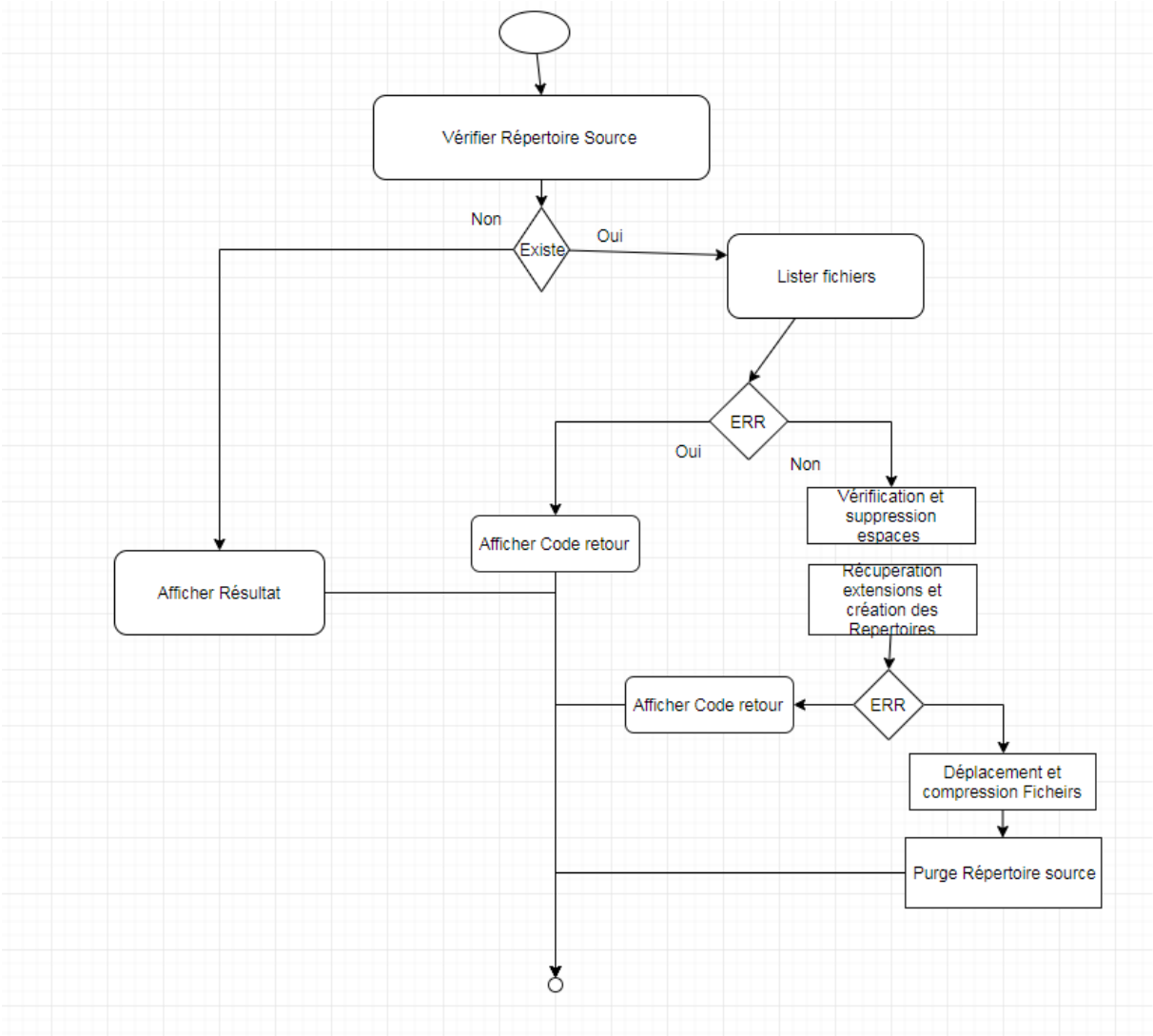
Voici les noms de quelques-uns des principaux shells qui existent.

- **sh** : *Bourne Shell*. L'ancêtre de tous les shells.
- **bash** : *Bourne Again Shell*. Une amélioration du *Bourne Shell*, disponible par défaut sous Linux et Mac OS X.
- **ksh** : *Korn Shell*. Un shell puissant assez présent sur les Unix propriétaires, mais aussi disponible en version libre, compatible avec bash.
- **csh** : *C Shell*. Un shell utilisant une syntaxe proche du langage C.
- **tcsh** : *Tenex C Shell*. Amélioration du *C Shell*.
- **zsh** : *Z Shell*. Shell assez récent reprenant les meilleures idées de bash, ksh et tcsh.

Exemple de programme :

```
#!/bin/bash
# Indique au système que l'argument qui suit est le programme utilisé pour exécuter ce fichier
# En règle générale, les "#" servent à mettre en commentaire le texte qui suit comme ici
echo Mon premier script
echo Liste des fichiers :
ls -la
```

Projet : Archivage de fichiers par type d'extension.



Détail du programme :

Partie1 :

```
#!/bin/bash
#mkdir dir
#cd ../
reset
source=input_files
destination=../backup
date=`date +%d%m%Y%H%M%S`
echo -e `date +%d-%m-%Y" "%H:%M:%S` "recuperation fichiers"
#On verifie si la commande est bien exécuté
ERR=$?
echo "Code retour : $ERR"
if [ $ERR -eq 0 ];
    then
        echo "pas d'erreur"
        #On recupere fichiers sources
        sleep 2
fi
```

Partie 2 :

```
cd $source
#ls
    #Suppression espaces nom de fichiers sinon traitement extension impossible
    for i in * ;
    do
        a=`echo $i | tr "[:blank:]" "_"`
        mv "$i" "$a"
    done

param='ls -l *'
data=${param%%.*}
for words in ${data} #avec nos parametres on va recuperer l'extension des fichiers'
    do
       echo "trie par extension" | mkdir -p ${words#*.}  #on creer le repertoire s'il existe pas d'erreur retournee
       for file in ${words%%*.} #On itere sur tous les fichiers
         do rep="${words#*.}" #variable contenant tous les repertoires
         mv "$file" "$rep" #pour chaque type d'extension, on va deplacer les fichiers dans le repertoire correspondant
       done
    done
```

Partie 3 :

```
#On compte le nombre de fichiers par extension
#On recupere les extensions pour creer autant de parametres
#On creer un parametre
#ls
echo -e `date +%d-%m-%Y" "%H:%M:%S` "trie et déplacement fichiers\n\a"
folder=backup_$date
mkdir $destination/$folder
mv * $destination/$folder
cd $destination

ERR=$?
if [ $ERR -eq 0 ];
    then
      echo "commande 1 pas d'erreur"
      echo `date +%d-%m-%Y" "%H:%M:%S` "Compression pour archivage..."

zip -r $folder.zip $destination
sleep 5
echo -e `date +%d-%m-%Y" "%H:%M:%S` "purge repertoire en cours\n\a"
rm -r $folder
sleep 5
echo -e "Archivage terminé avec succès\n\a"
    else
        echo "erreur, le programme ne peut pas continuer"
fi
```