Linguistique de corpus

Outils et méthodes de traitement de corpus

Patrick Paroubek

LIMSI-CNRS
Dépt. CHM - Groupe LIR
Bât. 508 Université Paris XI, 91403 Orsay Cedex
pap@limsi.fr

mercredi 13 novembre 2016 / L3 - Cours 6

Une commande que l'on crée peut avoir des arguments, ceux-ci sont identifiés dans le fichier texte du programme qui réalise la commande par des variables dont le nom est la position de l'argument considéré dans la ligne de commande tapée, le nom de la commande ayant le nom **0**, le premier argument **1**, etc. Par ex., si l'on veut pouvoir préciser un argument à la commande *affiche* lors de l'appel

```
>affiche /tmp/
```

et le fichier */home/tnml3/bin/affiche* contiendra :

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

Si le programme suivant est rangé dans le fichier exécutable */home/tim/prog.sh*

```
#!/bin/bash

echo "la commande est : $0"
echo "mon premier argument est : $1"
echo "mon second argument est : $2"
echo "mon troisième argument est : $3"
```

On aura les interactions shell suivantes :

```
tim> cd /home/tim
tim> ./prog.sh
la commande est : ./prog.sh
mon premier argument est
mon second argument est :
mon troisième argument est :
```

Si le programme suivant est rangé dans le fichier exécutable */home/tim/prog.sh*

```
tim> ./prog.sh arg1
la commande est : ./prog.sh
mon premier argument est arg1
mon second argument est :
mon troisième argument est :
tim>
tim>  ./prog.sh "arg1" 2 leTrois
la commande est : ./prog.sh
mon premier argument est arg1
mon second argument est : 2
mon troisième argument est : leTrois
tim>
tim>  ./prog.sh "arg1" 222 leTrois QUATRE
la commande est : ./prog.sh
mon premier argument est arg1
mon second argument est : 222
mon troisième argument est : leTrois
tim>
```

Remarquez que pour la dernière commande le quatrième argument est ignoré.

```
#!/bin/bash
#--------- exemple de branchement conditionnel -------------
echo "my first argument is $1"

if [[ $1 == "OUI" ]];
then
  echo "YES!!!!!!!!!!"
fi
```

```
#!/bin/bash
#--------- exemple de branchement conditionnel -------------
#          avec alternative
echo "my first argument is $1"

if [[ $1 == "OUI" ]];
then
  echo "YES!!!!!!!!!!"
else
  echo "nooooooooooo"
fi
```

```
#!/bin/bash

echo "my first argument is $1"

if [ "$1" == "OUI" -o "$1" == "oui" ];
then
   echo "YES!!!!!!!!!!"
 else
   echo "nooooooooooo"
 fi

echo "============ again ========"

if [ "$1" = "OUI" -o "$1" = "oui" ];
then
   echo "YES!!!!!!!!!!"
 else
   echo "nooooooooooo"
 fi

echo "=== une variante non standard (non POSIX) du shell  ========"
echo "===  qui utilise pour le test \"[[" au lieu de \"[" ========"

if [[ "$1" == "OUI" || "$1" == "oui" ]];
then
   echo "YES!!!!!!!!!!"
 else
   echo "nooooooooooo"
 fi
```

- Une commande rend toujours un résultat dans la variable d'environnement "$?".
- Ce résultat vaut 0 si la commande a pu produire un résultat (aucun problème rencontré).
- Ce résultat vaut 1,2,... si la commande n'a pas pu produire un résultat (un problème a été rencontré).

```
echo "hello" > /tmp/foo.txt;
ls -l /tmp/foo.txt
if [ "$?" == "0" ];
then
   echo "le fichier existe"
 else
   echo "le fichier n'existe PAS"
 fi
```

Tester si l'on a trouvé ou pas un motif dans un fichier

```
egrep "lo"  /tmp/foo.txt;
if [ "$?" == "0" ];
then
   echo "motif trouvé"
 else
   echo "motif NON trouvé"
 fi
```