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 09 novembre 2016 / L3 - Cours 7
Boucles et récursion

Un moyen en bash de répéter une série d'actions est par le biais de l'itérateur *for*

```
tim> for f in foo bar zoo
>do
> echo "itération  avec $f"
>done
iteration avec foo
iteration avec bar
iteration avec zoo
tim>
```

La syntaxe est la suivante :
for *nom de la variable d'itération* in *liste des valeurs d'itération*
do
liste des commandes itérées
done

Attention aux nombreuses manières de produire la liste des valeurs d'itération !

```
tim> for f in foo bar zoo
>do
> echo "itération  avec $f"
>done
iteration avec foo
iteration avec bar
iteration avec zoo
tim>
```

```
tim> for f in "foo bar zoo"
>do
> echo "itération  avec $f"
>done
iteration avec foo bar zoo
tim>
```

```
tim> for f in echo "foo bar zoo"
>do
> echo "itération  avec $f"
>done
iteration avec echo
iteration avec foo bar zoo
```

Utilisation de backquote " ‘ " (2 niveaux d'évaluation)

```
tim> for f in ‘echo foo bar zoo‘
>do
> echo "itération  avec $f"
>done
iteration avec foo
iteration avec bar
iteration avec zoo
tim>
```

Utilisation de backquote " ‘ " avec quote " ’ "

```
tim> for f in ‘echo ’foo bar zoo’‘
>do
> echo "itération  avec $f"
>done
iteration avec foo
iteration avec bar
iteration avec zoo
tim>
```

Utilisation de backquote " ‘ " avec doublequote " " "

```
tim> for f in ‘echo ’foo bar zoo’‘
>do
> echo "itération  avec $f"
>done
iteration avec foo
iteration avec bar
iteration avec zoo
tim>
```

avec une liste de fichiers contenue dans un fichier (avec *cat* et *backquote*)

```
tim> ls -1 /bin > /tmp/ liste_fichiers.txt
tim> for f in `cat /tmp/liste_fichiers.txt`
> do
>  echo "iteration avec $f"
> done
iteration avec bash
iteration avec bunzip2
...etc...
tim>
```

avec une liste de nombres consécutifs avec la commande *seq*

```
tim> for f in `seq 1 4`
> do
>  echo "iteration avec $f"
> done
iteration avec 1
iteration avec 2
iteration avec 3
iteration avec 4
tim>
```

avec une liste de nombres non consécutifs avec la commande *seq*

```
tim> for f in `seq 1 3 9`
> do
>  echo "iteration avec $f"
> done
iteration avec 1
iteration avec 4
iteration avec 7
tim>
```

Programmer en shell (suite)
Principe de la programmation recursive

Principe de la **programmation recursive**, le programme contient dans son code un appel à lui même (son propre nom), phénomène comparable au fait que dans un dictionnaire, le texte de la définition d'un mot peut contenir le mot même qu'il définit.
La programmation récursive est utilisée en complément du principe : *diviser pour régner*, qui consiste à scinder un problème en deux sous-problèmes plus petits et ainsi de suite, pour chaque sous-problèmes, jusqu'à arriver à un problème trivial à resoudre, puis rassembler les résultats de tous les problèmes triviaux pour former la solution ; sans avoir un recour à des instructions mentionnant des boucles explicites (*for*, *while*) .

Considérons le programme */tmp/compteA.sh* suivant écrit avec une boucle :

```
#!/bin/bash

echo "mon premier argument est $1"

for i in `seq 1 $1`
do
 echo "itération avec $i"
done
```

Lorsqu'on l'exécute

```
tim> /tmp/compteA.sh  3
mon premier argument est 3
itération avec 1
itération avec 2
itération avec 3
```

Voici le même programme */tmp/compteB.sh* écrit récursivement (sans boucle) :

```
#!/bin/bash

echo "mon premier argument est $1"
echo "mon second argument est $2"

#!/bin/bash

I="$1"
MAX="$2"

if [ "$I" == "$MAX" ];
then
  echo "$I";
else
  echo "$I";
  I=`echo "$I + 1" | bc`
  /tmp/compteB.sh  $I $MAX
fi
```

Remarquez que *compteB.sh* a perdu une boucle mais gagné un paramètre par rapport à compteA.sh

Lorsqu'on l'exécute :

```
tim> /tmp/compteB.sh 1 3
1
2
3
tim> /tmp/compteB.sh 3 1
3
4
5
6
7
8
9
10
11
...
467
...
```

Le programme */tmp/compteC.sh* utilise un test numérique *−lt* (inférieur à *less than*).

```
#!/bin/bash

I=$1
MAX=$2

if [ "$I" -lt "$MAX" ];
then
    echo "$I";
    I=`echo "$I + 1" | bc`
    /tmp/compteC.sh  $I $MAX
else
  if [ "$I" == "$MAX" ];
  then
    echo "$I";
  fi
fi
```

Et son utilisation

```
tim> /tmp/compteC.sh 1 3
1
2
3
tim> /tmp/compteC.sh 3 1
tim>
```