Examen court de système d'exploitation

Département d'informatique – IUT Villetaneuse

Mercredi 30 mars 2011 – 08 : 30 — 10 : 00

Remarques : tous les documents de td/tp sont autorisés MAIS pas les documents de cours. Le barème est indicatif. Veuillez expliquer ce que vous faites et ne pas vous contenter d'un programme.

1 Compréhension et développements de programmes Bash - 12 pts

1.1 Que fait le code suivant ? (2pt)

Donnez un exemple d'utilisation ainsi que le résultat pour le code :

```
#!/bin/bash

echo "Entrez une phrase:"
read PH
cmpt=0
for i in $PH
do
    cmpt=‘expr $cmpt + ${#i}‘
done
echo $cmpt
```

Le code se lance sans argument, il demande à saisir une phrase (suite de mots séparés par des blancs) et il calcule le nombre de caractères de la phrase.

1.2 Amélioration sur le code précédent (2pts)

Modifier le code précédent pour ne renvoyer que la somme totale des voyelles apparaissant dans la phrase.

```
#!/bin/bash

echo "Entrez une phrase:"
read PH
cmpt=0
for i in $PH
do
    for ((j=0;j<${#i};j++))
    do
       if [ ! -z `echo "aeiouyAEIOUY" | grep ${i:$j:1}` ]
       then
          cmpt=`expr $cmpt + 1`
       fi
    done
done
echo $cmpt
```

1.3 Compléter le code précédent (2pts)

Modifier le code précédent pour afficher la somme totale des voyelles apparaissant dans la phrase ainsi que la somme totale des autres caractères.

Le premier code retourne (disons dans la variable s) le nombre de caractère de la phase. Le deuxième code retourne (disons dans la variable t) la somme des voyelles présentes dans la phase. Une solution consiste à afficher s-t.

1.4 Développement d'un code (6pts)

La commande Unix `cal <month> <year>` permet d'afficher un calendrier sous la forme suivante :

```
$ cal 12 2009
   December 2009
 S  M Tu  W Th  F  S
       1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 31
```

On vous demande d'écrire un script Bash qui lira la sortie de la commande `cal` et la transformera, pour notre exemple, en :

```
 ;  ;  1  ;  2  ;  3  ;  4  ;  5
 6  ;  7  ;  8  ;  9  ;  10 ;  11 ;  12
13  ;  14 ;  15 ;  16 ;  17 ;  18 ;  19
20  ;  21 ;  22 ;  23 ;  24 ;  25 ;  26
27  ;  28 ;  29 ;  30 ;  31 ;  ;
```

Veuillez commencer :

* par expliquer les difficultés techniques qui se posent ; (1pt)
* puis par expliquer vos approches afin de présenter un algorithme ; (1pt)
* et enfin vous coderez cet algorithme en Bash ; (4pts)

Difficultés techniques :

- Isoler les lignes pertinentes (peut se régler avec une combinaison de head/tail)
- On supprime les 2 premières lignes du fichier (facile avec tail)
- Pour les lignes qui comportent 7 entiers, on peut simplement construire une chaine qui contient la concaténée d'un entier suivi d'un ;
- Pour les lignes qui comportent moins de 7 entiers, il est difficile de repérer si le "bloc d'entier" est calé sur la gauche de la ligne ou sur la droite. En fait on remarque assez vite que des blancs en tête de ligne ne peuvent apparaitre que sur la première ligne et que les blancs en fin de ligne ne peuvent appraitre que pour la dernière ligne.

Pour isoler les lignes pertinentes issues de la commande cal, on peut partir comme suit :

```
# calcul du nombre de lignes
NbLigne=‘cal $1 $2 | wc -l‘
echo "$NbLigne"
NbLigne=‘expr $NbLigne - 2‘
echo "$NbLigne"
n=‘expr $NbLigne - 1‘
echo "$n"
MesLignes=‘cal $1 $2 | tail -n $NbLigne | head -n $n‘
echo "$NbLigne -- $MesLignes"
```

Idée d'algorithme :

On met en place un traitement spécifique pour la première ligne, puis un traitement spécifique sur les lignes centrales, puis un traitement spécifique pour la dernière ligne.

```
1) Pour la première ligne :
     o on compte le nombre de "mots" (nb_mots)
     o on produit 7 - nb_mots ; suivi des nb_mots séparés par les ;
nb=‘echo "       1  2  3  4  5" | wc -w‘
if [ $nb -lt 7 ]
then
  j=1
  for ((i=1;i<$((8-$nb));i++))
  do
      echo -n " ; "
  done
  for i in ‘echo "       1  2  3  4  5"‘
  do
    if [ $j -lt $nb ]
    then
        echo -n "$i ; "
    else
        echo -n "$i"
    fi
    j=$((j+1))
  done
  echo
fi


2) Pour les lignes centrales :
     o on produit 7 mots séparés par des ;
nb=‘echo " 6  7  8  9 10 11 12" | wc -w‘
if [ $nb -eq 7 ]
then
  j=1
  for i in ‘echo " 6  7  8  9 10 11 12"‘
  do
    if [ $j -lt $nb ]
    then
        echo -n "$i ; "
    else
        echo -n "$i"
    fi
    j=$((j+1))
  done
```

```
  echo
fi


3) Pour la dernière ligne :
     o on compte le nombre de "mots" (nb_mots)
     o on produit nb_mots séparés par des ; suivi de 7-nb_mots-1 ;
nb=‘echo "27 28 29 30 31" | wc -w‘
if [ $nb -lt 7 ]
then
  j=1
  for i in ‘echo "27 28 29 30 31"‘
  do
    if [ $j -lt $nb ]
    then
        echo -n "$i ; "
    else
        echo -n "$i"
    fi
    j=$((j+1))
  done
  for ((i=$j;i<8;i++))
  do
      echo -n " ; "
  done
  echo
fi

Remarque : 3) peut servir aussi au traitement mentionné au point 2)
```

2 Expression régulière (4pts)

a) Donner une expression régulière qui permette de tester si une ligne d'un fichier texte se termine par un entier sur exactement 1 ou 2 digits.

```
echo "       1  2  3  4  555" | egrep '[^[:digit:]][[:digit:]]{1,2}$'
```

b) Expliquez pourquoi la solution suivante ne convient pas :

```
echo "       1  2  3  4  555" | egrep '[[:digit:]]{1,2}$'
```

Il faut tester que le caractère avant les un/deux derniers digits n'est pas un entier.

c) Donner une expression régulière qui permette de tester si une ligne débute par pas plus de 3 digits consécutifs.

```
echo "1111  2  3  4  555" | egrep '^[[:digit:]]{1,3}[^[:digit:]]'
```

3 Questions de cours (4pts)

a) Dans le dernier cours, Make a été comparé avec un outil du monde Java. Quel est-il ?

ANT

b) Dans le dernier cours, j'ai présenté la commande nm. A quoi sert-elle ?

nm permet de lister dans un executable le nom des fonctions
qui apparaissent dans l'exécutable et si ces fonctions
apparaissent dans l'exécutable ou si elles sont externes
(symbole U => undefined => dans une librairie)

c) Comment traduit-on Prefetching en français et en quoi consiste cette technique ?

Prefetching = Lecture anticipée en mémoire.

Il s'agit de précharger dans le cache des données
dont on va avoir besoin dans un futur proche.

d) Résumez ce qu'il y a dans le polycopié Bash concernant la définition de fonctions dans une fonction.

On peut écrire une fonction dans une fonction... mais elle n'est
pas locale à la fonction englobante, elle devient globale à tout le
programme.
(cf page 53)