

# Contrôle court de système d'exploitation

## Semestre décalé

Département d'informatique IUT Villetaneuse

Lundi 13 novembre 2006, 1h

**Remarques :** tous les documents de cours, td/tp sont autorisés. Le barème est indicatif.

### 1- Commandes unix, questions de cours – 4 pts

	<i>Commandes</i>	<i>Questions / Affirmations</i>	<i>Réponse / Justification</i>	<i>Points</i>
1.1	<code># gcc -O4 -S essai.c</code>	Le code assembleur généré n'est pas optimisé	Vrai ou Faux	1/2
1.2	<code># x=\$((n))</code>	En bash, l'écriture suivante est correcte ?	Vrai ou Faux	1/2
1.3		Bash implémente une notion de fonction.	Vrai ou Faux	1/2
1.4		La commande test est une instruction interne de Bash.	Vrai ou Faux	1/2
1.5		Quelle est la commande que l'on applique sur un exécutable pour lister les fonctions que cet exécutable contient ?		1/2
1.6		Dans <code>grep -E '[[[:digit:]] [[[:blank:]]]+' bio.txt</code> l'expression <code>[[[:digit:]] [[[:blank:]]]+' est une expression régulière.</code>	Vrai ou Faux.	1/2
1.7	<code>\$ find . -name "a*b*c" -print</code>	Si le répertoire courant contient le fichier 'abc', il sera affiché.	Vrai ou Faux	1

## 2- Langage de commande bash - 16 pts

### 2.1 Le problème du dictionnaire (7 pts)

Un dictionnaire est une structure de données qui permet d'accéder à une information par une clé. On représente un dictionnaire de la manière suivante :

```
{ a="Professeur" , b="1" , c="XXX" }
```

Ici les clés sont a, b, c. La valeur de la clé a est "Professeur" ; la valeur de la clé b est "1"; la valeur de la clé c est "XXX". Le séparateur , sert à délimiter les paires (clé, valeur) alors que { et } servent à délimiter un dictionnaire.

a) Ecrire un programme bash qui prend en paramètre un fichier contenant sur une seule ligne un objet dictionnaire, qui comporte une boucle et qui affiche toutes les paires (clé,valeur).

b) Ecrire un programme bash qui prend en paramètre un fichier contenant sur une seule ligne un objet dictionnaire et une clé et qui retournera la valeur de la clé correspondante.

Note : on pourra utiliser `cut` et la forme syntaxique suivante pour éliminer des caractères dans une chaîne :

```
${parameter/pattern/string}  
${parameter//pattern/string}
```

Le motif 'pattern' est expansé pour produire un nom. Si ce nom est trouvé dans la valeur de la variable parameter alors la 'sous-chaine' nom dans parameter est remplacée par string. Dans la première forme, c'est seulement la première occurrence de pattern qui est substituée. Dans la seconde forme, ce sont toutes les occurrences qui sont remplacées par string.

Exemples :

```
# j="{ a=Professeur , b=1 , c={ d=1 } }"  
# echo ${j/{/}  
a=Professeur , b=1 , c={ d=1 } }  
# echo ${j//=/}  
{ aProfesseur , b1 , c{ d1 } }  
# echo ${j/=*/}  
{ a
```

### 2.2 Vérification d'une propriété (9 pts)

Ecrire un code Bash qui liste les 5 premières lignes des fichiers en .txt du répertoire donné en paramètre 1 de la ligne de commande SI l'on ne trouve pas dans ces 5 lignes un motif passé en paramètre 2. Si on trouve le motif on affichera le nom du fichier. On fera toutes les vérifications d'usage. L'exécution donnera par exemple:

```
# /bin/bash MonCode.sh . "P*o"
```

```
==== Fichier ./ara.txt
```

Bonjour,

Le site des ARA est <http://arassia.loria.fr>

```
==== Fichier ./bio.txt contient le motif
```

```
==== Fichier ./licPRO.txt contient le motif
```

## Correction du 1

Faux  
vrai  
vrai  
vrai  
nm  
vrai  
vrai

## Proposition de correction du 2.1

Le dictionnaire est traité comme un tableau dont les colonnes sont séparées par des ,. L'algorithme est donc :

```
# $1 => nom du fichier
# $2 => la clé recherchée
si $1 est un fichier et qu'il est lisible alors
    pour toutes les colonnes C_i faire
        si C_i contient le motif $2= alors
            # on est en presence de l'objet (clé,valeur) recherché
            traiter C_i # repérer la valeur associée à $2 => utilisez ${C_i//}
        finsi
    fin pour
sinon
    afficher probleme avec $1
finsi
```

```
#!/bin/bash
# Version simplifiée de l'algorithme ci-dessus
j="{ a=Professeur , b=1 , c={d=1 , k=33} , e=bonjour }"
myclef="e"
```

```
# Attention : ce code ne fonctionne pas lorsque la valeur d'une clé est
# un dictionnaire. En effet les , dans c={d=1 , k=33} perturbent la fouille
# par colonne effectuée par cut et de plus le ${inter/*=/} élimine tous les
# caractères jusqu'au = le plus à droite ce qui nous fait retourner
# 33 en lieu et place de {d=1 , k=33}
```

```
i=1
inter=`echo $j | cut -d "," -f 1`
# on examine un par un les champs sur chaque colonne
while [ "$inter" != "" ]
do
    inter=${inter/{/}      # on elimine les {
    inter=${inter//\}/}    # on elimine les }
    #echo $inter
    # on verifie si on a affaire a la cle
    if [ ${inter/*=/} = "$myclef" ]
    then
        # on affiche le resultat
        echo "La valeur de la clef $myclef est ${inter/*=/}"
        # on sort du while puisqu'on a trouve'
        break
    else
        # on passe au champs suivant
        i=$((i+1))
        inter=`echo $j | cut -d "," -f $i`
    fi
done
```

```
# Dans la solution ci dessous c'est la separation des 'tokens' par des
# [:blank:]ou des , qui fait que le resultat est correct. Ce programme
# est donc dépendant de la syntaxe... et de plus, des dictionnaires
# qui sont des valeurs de clés ne sont toujours pas bien traités.
```

```
#!/bin/bash

j="{ a=Professeur , b=1 , c={d=1,k=33} , e=bonjour }"
myclef="c"

# on examine un par un les champs sur chaque colonne
for inter in `echo $j`
do
    #echo $inter
    # on verifie si on a affaire a la cle
    c=`echo "$inter" | grep "$myclef=" `
    echo $c
    if [ "$c" != "" ]
    then
        # on affiche le resultat
        echo "La valeur de la clef $myclef est ${c#*=}"
        # on sort du while puisqu'on a trouve'
        break
    fi
done
```

Correction du 2.2

```
#####
```

```
#!/bin/bash
# Lister les 5 premières lignes de tous les fichiers en .txt
# du répertoire en paramètre 1 et vérification que ces 5 lignes ne
# contiennent pas le motif $2
```

```
if [ $# -ne 2 -o ! -d "$1" ]; then
    echo "Usage : $0 <nom-de-repertoire> <motif>"
    exit 1
fi
for i in $1/*.txt; do
    if [ -f $i ]; then
        inter=`head -5 $i | grep $2`
        if [ "$inter" = "" ]; then
            echo -e "\n\n==== Fichier : $i\n"
            head -5 $i
        else
            echo -e "\n\n==== Fichier $i contient le motif\n"
        fi
    fi
done
```