Test n°1

Commandes UNIX et programmation shell

CORRECTION

Exercice 1 – Questions de cours

Solution :

1) 4 grands types de tâches qu'un SE réalise :
   - la gestion des processus,
   - la gestion de la mémoire,
   - la gestion du système de fichiers,
   - la gestion des périphériques d'E/S.

2) Un processus est un programme en cours d'exécution. Programme = instructions machines (statique). Processus = réalisation d'actions (dynamique).

3) Avantages à utiliser un script (l'un ou l'autre)
   - travail en ligne de commande souvent + efficace qu'à travers une interface graphique,
   - automatisation de tâches répétitives

Exercice 2 – Commandes UNIX

Solution :

1) `cat` ou `echo`

2) `pwd`

3) `who`

4) La commande `cat > fichier` recouvre le contenu du fichier. La commande `cat >> fichier` rajoute les données à la fin du fichier.

5) `*` : n'importe quelle chaîne de caractères (y compris vide)
   `?` : n'importe quel caractère
   `[ab]` : " a " ou " b "
   `[a-d]` : " a ", " b ", " c " ou " d "

6) `cd ..`

7) `find` ou `whereis`

8) `chmod` permet de changer les droits d'accès. Ex: `chmod 750 fich` ou `chmod g+x fich`

Exercice 3 – Expressions régulières

Solution :

- `grep '^R..$'` : lignes de 3 caractères exactement dont le 1er est un 'R'
- `grep '[agct]'` : lignes contenant les lettres a, g, c ou t
- `grep '[AGCT].$'` : lignes dont l'avant-dernière lettre est un A, G, C ou T
- `grep '[AGCT]{2}'` : lignes contenant un A, G, C ou T deux fois de suite
- `grep '[A-Z]'` : lignes contenant une majuscule
- `grep '[0123]'` : lignes contenant les chiffres 0, 1, 2 ou 3
- `grep '[0-9]'` : lignes contenant un chiffre compris entre 0 et 9
- `grep '[agct]+'` : lignes contenant les lettres a, g, c ou t au moins une fois

Exercice 4 – Bash : écriture de scripts

Solution :

```
#!/bin/bash
# On verifie si le fichier existe
if [ $# -lt 1 ]
then
    echo "Il faut donner un nom de fichier en argument"
    exit
fi
if !(test -f $1)
then
    echo "Fichier $1 inconnu !"
    exit
fi
# on efface l'ecran
tput clear
# on commence par faire une premiere passe pour
# isoler le max et le min. On suppose que l'entree
# est constituee d'entiers strictement positifs.
min=0
max=0
i=0
echo "------Fichier en entree: $1-------------"
cat < $1 | while true
   do
      read ligne
      # traiter la ligne
      if [ "$ligne" = "" ]; then break; fi
      one=`echo $ligne | cut -d " " -f 1`
      two=`echo $ligne | cut -d " " -f 2`
      if [ $i -eq 0 ]
      then
         max=$two
         min=$one
         i=$((i+12))
      fi
      if [ $one -gt $max ]
      then
           max=$one
      fi
```

```
      if [ $two -gt $max ]
      then
           max=$two
      fi
      if [ $one -lt $min ]
      then
           min=$one
      fi
     if [ $two -lt $min ]
     then
          min=$two
     fi
     # on ecrit le max et le min dans un fichier
     echo "$max $min" > /tmp/bidon
  done
# On effectue la deuxieme passe
# on commence par recuperer le max et le min
max=`cat /tmp/bidon |cut -d " " -f 1`
min=`cat /tmp/bidon |cut -d " " -f 2`
# on reprend
cat < $1 | while true
  do
     read ligne
     # traiter la ligne
     if [ "$ligne" = "" ]; then break; fi
     one=`echo $ligne | cut -d " " -f 1`
     two=`echo $ligne | cut -d " " -f 2`
     echo $one $two $((two+one)) $min $max
  done
# Exemple de resultat d'execution :
# ------Fichier en entree: e.txt-------------
# 1 2 3 1 9
# 1 3 4 1 9
# 4 2 6 1 9
# 5 6 11 1 9
# 7 3 10 1 9
# 3 9 12 1 9
```