

# 3R-IN3 – Système d’exploitation – Contrôle d’unité

Durée : 20mn + 1h40mn.

Conditions de l’épreuve : voir encadré

Xavier Hilaire  
x.hilaire@esiee.fr

16 juin 2021

## TRÈS IMPORTANT

Vous devez rendre dans tous les cas **deux copies** à ce contrôle (même blanches) :

- Une première traitant l’exercice 1 ci-dessous (questions de cours) à l’issue des 20 premières minutes de l’épreuve.
- La deuxième traitant les autres exercices.

**Les documents ne sont autorisés qu’après ramassage des premières copies.**

**Le matériel électronique, sous toutes ses formes, est interdit durant toute l’épreuve.**

Conseils :

- **Soyez précis et concis** : aucune des réponses aux questions de ce sujet ne nécessite plus de 10 lignes d’explication ou de justification.
- La clarté de vos réponses est un élément d’appréciation essentiel.

## 1 Questions de cours (sans documents, 4 pt)

1. (1.5 pts) Soit la sortie suivante, produite par `ls -l`

```
$ ls -l
total 120
-rwxrw-r-x  1 tim      wheel    1932 Apr 22 18:59 a.out
-rwxr-xr--  1 michael admin     567 Apr 22 19:01 config
-r--r----- 1 root     wheel     587 Apr 22 19:01 efax.rc
-rw-rw----  1 joe      wheel     150 Apr 22 19:00 fstab.hd
-rwx---r-x  1 tim      users     106 Apr 22 19:00 init.inf
-rwsr-sr-x  1 boss     users   40960 Apr 22 19:01 accounting
$
```

L’utilisateur courant est joe, et n’appartient qu’au groupe users. Quels fichiers peut-il lire ? Modifier ? Exécuter ?

2. (0.5 pts) Quel est l’effet de la commande `shift` ?
3. (1 pt) Que fait la commande `wait` du shell ? Quelle variable spéciale est-elle affectée juste après son exécution ?
4. (1 pt) Soit le code suivant :

```

if untest; then
    echo oui
else
    echo non
fi

```

A supposer que `untest` soit un script shell, par quoi doit-il obligatoirement se terminer pour que l'on voit s'afficher oui à l'écran ?

## 2 Mélange (4 pt)

La variable spéciale `RANDOM` du shell renvoie un entier aléatoire entre 0 et 32767 à chaque fois qu'elle est évaluée. Exemple :

```

$ echo $RANDOM
17506
$ echo $RANDOM
2239
$

```

Vous disposez d'un **fichier** source, dont vous pourrez supposer qu'il contient moins de 32767 lignes, et vous voudriez écrire un script ou une fonction **desordre** tel que

```
cat fichier | desordre
```

produise une version « mélangée » du fichier **source**, c'est-à-dire que chaque ligne texte de **source** est bien sortie exactement une fois, mais à une nouvelle position imprévisible. Vous n'avez le droit de créer aucun fichier.

Ecrire ce script ou cette fonction. Indications : insérer un nombre aléatoire devant chaque ligne, puis trier, et enfin retirer les numéros.

## 3 Historique d'impression (6 pts)

Un serveur d'impression tient à jour un historique dans un fichier texte `/var/log/printing`, en y ajoutant une ligne à chaque fois qu'un utilisateur demande un travail d'impression. Voici un exemple de ce fichier :

```

joe:5:20110903
joe:2:20120721
mike:32:20120718
helen:3:20120718
joe:1:20120802
mike:1:20120803

```

Chaque ligne se présente sous la forme `login:nbpages:yyyymmdd` où `login` est le login de l'utilisateur qui a demandé une impression, `nbpages` le nombre de pages qu'il a consommées, et `yyyymmdd` la date à laquelle elle s'est achevée.

Ecrivez un script `compte.sh` prenant 1, 2 ou 3 arguments et se comportant ainsi :

1. S'il n'est appelé qu'avec un seul argument, alors il s'agit d'un login. Le script doit alors écrire ce login, suivi de la somme de toutes les pages qu'il a consommées. Exemple :

```

$ ./compte.sh joe
joe 8
$

```

2. S'il est appelé avec 2 ou 3 arguments, alors le premier est toujours le login, le deuxième une année et le troisième un mois entre 01 et 12. Dans ce cas de figure, le script ne doit retourner la somme des pages consommées soit que pour l'année spécifiée si le script n'est appelé qu'avec 2 arguments, soit que pour l'année *et* le mois spécifiés s'il est appelé avec 3 arguments :

```
$ ./compte.sh joe 2011
joe 5
$ ./compte.sh joe 2012
joe 3
$ ./compte.sh joe 2012 08
joe 1
$
```

## 4 Histogramme (6 pts)

Un développeur doit produire un script qui calcule le nombre d'occurrences des mots que comporte un fichier texte, passé en unique argument du script. Le fichier est codé en ASCII standard (1 octet par caractère). Il propose le code suivant :

```
#!/bin/sh
#

mkdir /tmp/toto
cat $1 | (while read line; do
    for mot in $line; do
        x=$(cat /tmp/toto/$mot 2>/dev/null)
        echo $((x+1)) > /tmp/toto/$mot
    done
done
)

for f in /tmp/toto/*; do
    echo -n "$mot  "
    cat /tmp/toto/$mot
done
```

Questions :

1. Quel est (somairement) le principe de fonctionnement utilisé dans ce script ? Qu'affiche t'il ?
2. Est-on sûr que l'affichage est toujours celui attendu d'une exécution à l'autre ? Dans l'affirmative, dire pourquoi ; dans la négative, donner les corrections à apporter.
3. En examinant le résultat produit sur quelques exemples, il se rend rapidement compte que les caractères de ponctuation posent un problème : par exemple, la phrase « si oui, qu'affiche t'il ? » donnera un décompte pour « si », « oui », « qu'affiche », « t'il ? » au lieu de « si », « oui », « qu », « affiche », « t », « il ». En admettant que seuls les caractères `, ; . : ? ! ' "` faussent le décompte, quelle modification (simple) proposez-vous pour résoudre ce problème ?
4. Décrivez (sans la coder) une solution qui produirait les mêmes résultats mais sans créer aucun fichier.