

ISS - Initiation aux Systèmes d'exploitation et au Shell  
LU2IN020

Partiel du 20 novembre 2021

**Nom :**

**Prénom :**

**Numéro de groupe :**

**Aucun document autorisé pendant l'épreuve**

Les téléphones portables, les montres connectées et autres appareils doivent être rangés dans votre sac.  
Le barème n'est donné qu'à titre indicatif, pour vous permettre de juger de la difficulté des questions.

Attention : l'énoncé est imprimé recto-verso sur **10 pages**.

**Hypothèse pour l'ensemble de l'examen :** Pour simplifier, si les questions n'indiquent pas le contraire, on supposera que tous les exécutables sont bien présents dans le répertoire de l'exercice et que les droits nécessaires à leurs exécutions sont attribués à l'ensemble des utilisateurs.

**Exercice 1 : Questions de cours (6,5 points)**

**Question 1 – 1,5 point**

Dessinez, tel que présenté au début de chaque cours, l'ensemble de la mémoire d'un processus en indiquant ses différentes zones, les registres spéciaux, ainsi que l'ensemble des données de l'OS permettant de gérer le fonctionnement de ce processus.

**Question 2 – 1,5 point**

Dessinez l'ensemble des couches logicielles séparant l'application du matériel. Votre schéma devra distinguer clairement l'ensemble des couches qui constitue un système d'exploitation, ainsi que les trois grands services rendus par celui-ci. Il devra, en outre, faire apparaître le mode d'exécution du processeur lié à ces couches. Pour simplifier, il est ici inutile de multiplier les applications (une seule suffit).

**Question 3 – 1 point**

La fonction C `open()` retourne un entier, quel est son nom et à quoi correspond-il ? Où peut-on en voir une trace de ces entiers dans les mécanismes du *Bash* étudiés dans cette UE ?

**Question 4 – 0,5 point**

Quelle est la définition du cours du mot processus ?

**Question 5 – 0,5 point**

Qu'appelle-t-on *noyau* d'un système d'exploitation ?

**Question 6 – 0,5 point**

Lorsqu'un programme utilisateur fait un *appel système*, il obtient tous les pouvoirs. Pourquoi ceci ne présente-t-il pas une véritable faille de sécurité ?

**Question 7 – 0,5 point**

En déboguant une application sur mon téléphone, je vois apparaître l'adresse mémoire **0xC19F3100** (donc supérieur à 3 milliards) pourtant celui-ci ne possède que 2 Go de RAM. Comment expliquez-vous cela ?

**Question 8 – 0,5 point**

Qui définit la valeur du *shebang* ?

## Exercice 2 : Ping Pong, les débuts (4 points)

Dans cet exercice, on veut réaliser un ensemble de scripts permettant d'afficher une succession de ping / pong, où **chaque ligne est produite par une nouvelle instance**. Ainsi, l'exécution du script `match.sh` produira l'affichage suivant :

```
match.sh
#!/bin/bash
./mesScripts/ping.sh 5
```

```
moi@pc ~home/moi $ ./match.sh
ping
pong
ping
pong
ping
```

**Aide :** dans cet exercice, vous pouvez utiliser la commande `dirname` qui affiche le répertoire contenant le fichier dont le nom est passé en paramètre (*e.g.*, `dirname /tmp/dir/f1` affiche `/tmp/dir/`)

### Question 1 – 2 points

Sans modifier `match.sh`, donnez une implémentation de deux scripts `ping.sh` et `pong.sh` qui seront responsables des affichages des chaînes correspondantes à leurs noms. On supposera que ces deux scripts sont dans le même répertoire. Mais attention, votre solution devra rester fonctionnelle, quel que soit l'emplacement de ces scripts (*i.e.*, même si ce répertoire commun n'est pas `~/mesScripts/`), sachant que leur répertoire commun n'est pas forcément celui de `match.sh` et ne se trouve pas dans le PATH.

### Question 2 – 1 point

Dessinez un chronogramme correspondant au lancement du script `./match.sh` dans un *Bash*. Votre schéma devra faire apparaître l'ensemble des processus créés, leurs liens de parenté, ainsi que leurs éventuelles périodes de blocage.



**Question 3 – 1 point**

Donnez maintenant, une deuxième version de ces deux scripts qui se passe complètement de paramètre. Vous adapterez aussi le script `match.sh` pour qu'il soit compatible avec ces nouvelles versions.

**Exercice 3 : Tee time (4,5 points)**

Dans cet exercice, on veut créer un script générant les convocations aux examens pour l'ensemble des étudiants de la licence d'informatique.

**Question 1 – 2 points**

Pour ce faire, nous voulons utiliser la commande `tee` qui n'est malheureusement pas installée sur notre machine. Cette commande effectue le traitement suivant :

1. lecture d'une ligne sur son entrée standard
2. ajout de la ligne lue à la fin de tous les fichiers dont les chemins sont passés en paramètre
3. réécriture de la ligne lue sur sa sortie standard
4. arrêt si le flux d'entrée est terminé, sinon la commande boucle sur la première étape

Donnez votre implémentation personnelle `my_tee.sh` de cette commande. Votre script devra retourner une erreur s'il n'y a aucun paramètre.

**Question 2 – 0,5 point**

Donnez, en justifiant votre réponse, le résultat de la commande suivante

```
moi@pc /home/moi $ echo "LU2IN020 - Partiel - Amphi 1" | ./my_tee.sh etudiant_A etudiant_B | tr 1 2 | ./my_tee.sh etudiant_X etudiant_Y > /dev/null
```

### Question 3 – 2 points

En reprenant cette ligne de commande, implémentez un script `convocations.sh` qui génère, pour chaque étudiant, un fichier de convocation indiquant la liste des examens. Ces convocations ne doivent comporter qu'une seule épreuve (examen, partiel, TME\_solo, ...) par ligne respectant le format suivant : `Nom_UE Type_examens Amphi`

Votre script prendra trois fichiers en paramètre (dont des exemples vous sont donnés plus bas).

1. un fichier contenant la maquette de la licence avec une ligne par UE comprenant son code, suivi de la liste de ses épreuves
2. un fichier contenant la liste des étudiants passant dans l'amphi 1 (un étudiant par ligne sans espaces)
3. un fichier contenant la liste des étudiants passant dans l'amphi 2 (un étudiant par ligne sans espaces)

`maquette.txt`

```
LU2IN010 Examen  
LU2IN020 Partiel Examen  
LU2IN030 TME_Solo Partiel Examen  
LU2IN040 Partiel Examen
```

`grp1.lst`

```
etudiant_A  
etudiant_B  
etudiant_C
```

`grp2.lst`

```
etudiant_X  
etudiant_Y  
etudiant_Z
```

Ainsi, sur cet exemple, la commande `convocations.sh maquette.txt grp1.lst grp2.lst` doit générer 6 fichiers comprenant chacun 8 lignes.

Pour simplifier, on supposera que `my_tee.sh` est dans le même répertoire et que votre script sera correctement utilisé.

**Exercice 4 : Le compte est bon(5 points)**

Pour l'ensemble de l'exercice, on suppose que l'on a accès à un fichier `comptes.txt` contenant les comptes d'une entreprise mois par mois sur plusieurs années. On trouve ainsi sur chaque ligne : l'année et le mois (noté en décimales), suivis des crédits et des débits sur ledit mois. Attention, ce fichier n'est pas un csv. Vous trouverez son format (supposé connu et fixe) sur la première ligne du fichier, dont voici un extrait :

**Début du fichier `comptes.txt`**

```
annee/mois,credit:debit
2021/10,234567:208743
2022/01,012333:010424
2022/03,010923:009214
2020/04,019345:019345
2018/10,029448:029448
2018/12,181823:919345
2018/09,119345:119345
```

**Question 1 – 1 point**

Proposez une commande qui affiche le nombre d'années mentionnées dans ce fichier.

**Question 2 – 1 point**

Proposez une commande qui affiche la valeur du plus grand débit mensuel enregistré dans le fichier.

**Question 3 – 1 point**

Proposez une commande qui affiche une liste des dates (année/mois) des entrées qui sont à l'équilibre, *i.e.*, où crédits et débits sont égaux.

**Question 4 – 2 points**

Proposez un script qui affiche sur une ligne, la liste des années mentionnées dans le fichier qui n'ont pas une entrée (*i.e.*, par mois).

**Exercice 5 : Ping-Pong, le retour royal (Bonus, juste pour le fun)**

Dans cet exercice bonus, on revient sur l'exercice 2 en ajoutant des contraintes.

**Question 1**

Répondre aux questions de l'exercice 2 en ajoutant les 2 contraintes suivantes :

- les deux fichiers des scripts `ping.sh` et `pong.sh` doivent être complètement identiques. Nous verrons plus tard dans le semestre qu'une telle solution permettra de faire de `pong.sh` un lien vers `-kwFping.sh` et ainsi simplifier leur maintenance.
- sans aucune modification, vos codes devront aussi être les solutions d'un exercice où l'on aurait substitué `ping.sh` par `king.sh` et `pong.sh` par `kong.sh`.

En absence de la commande `sed`, vous pourrez être amené à utiliser la commande `basename` qui prend un chemin en paramètre et affiche uniquement le nom du fichier.

Point de grand script ici, la solution peut être composée que de deux lignes de commandes.

