ISS - Initiation aux Systèmes d'exploitation et au Shell
LU2IN020
Partiel du 18 novembre 2021

Nom : **Prénom :** **Numéro de groupe :**

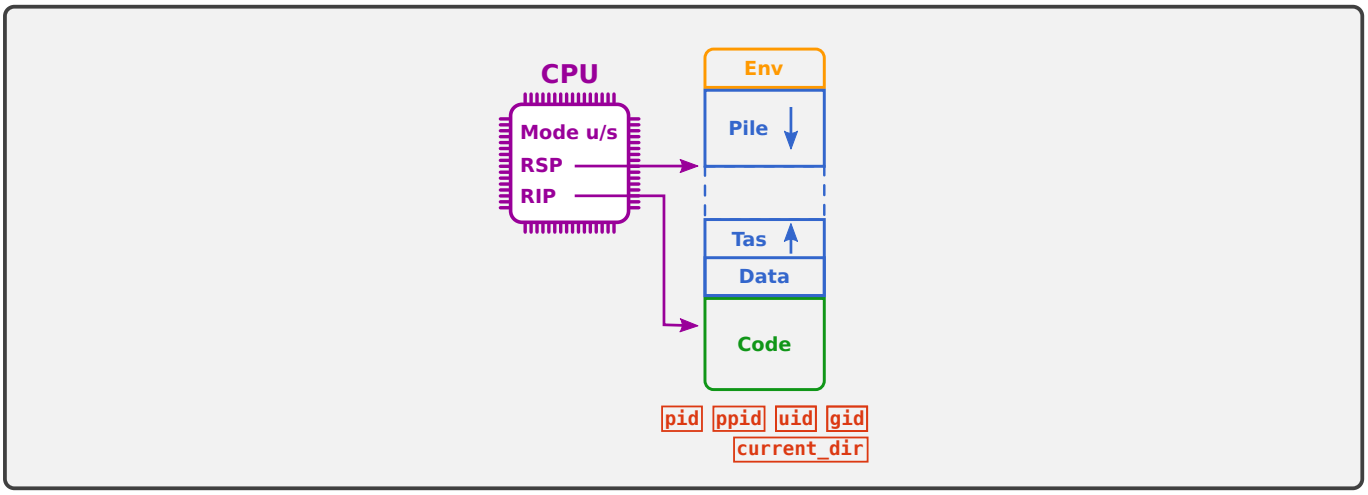
Aucun document autorisé pendant l'épreuve
Les téléphones portables, les montres connectées et autres appareils doivent être rangés dans votre sac.
Le barème n'est donné qu'à titre indicatif, pour vous permettre de juger de la difficulté des questions.
Attention : l'énoncé est imprimé recto-verso sur **7 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

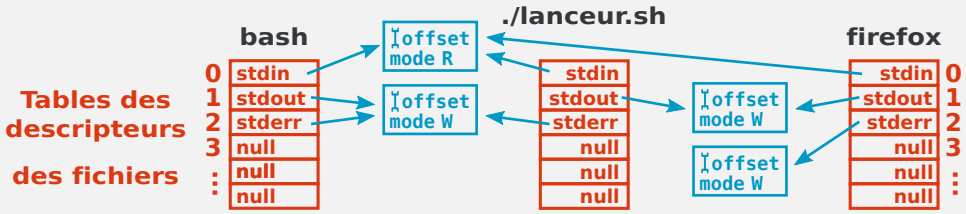
Dessinez, tel que présenté au début de chaque cours, l'ensemble de la mémoire d'un processus en indiquant ses différentes zones, les registres spéciaux, ainsi que l'ensemble des données de l'OS permettant de gérer le fonctionnement de ce processus, à l'exception de la table des descripteurs qui est l'objet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



Question 2 – 1,5 point

Dessinez maintenant sur un même schéma les 3 tables des descripteurs et les structures files (contenant notamment un champ offset) correspondant à 1 démarrage de Firefox après l'exécution suivante :

```
moi@pc /home/moi $ cat lanceur.sh
#! /bin/bash
Firefox 2> log2
moi@pc /home/moi $ ./lanceur.sh > Log1
```



Question 3 – 0,5 point

Quelle est la différence entre un kernel (noyau) d'un système d'exploitation et un système d'exploitation ?

Un système d'exploitation est formé d'un noyau et d'un ensemble de services ou de bibliothèques offrant une API haut niveau pour utiliser le matériel.

Le noyau c'est la partie du système d'exploitation qui s'exécute en mode privilégié (mode S). On peut le voir comme une bibliothèque exécutée en mode S par les processus lorsqu'ils font un appel système.

Question 4 – 0,5 point

Quelle est la définition du cours du mot processus ?

Un processus c'est un espace d'adressage dans lequel s'exécute un programme. Il contient un ou plusieurs fils d'exécution (thread).

Question 5 – 1 point

Quelles sont les limites d'un code qui s'exécute en mode U ? Qui est responsable de ce contrôle ?

Lorsqu'un code s'exécute en mode U, le processeur limite :
— les adresses accédées
— les instructions assembleurs

Question 6 – 0,5 point

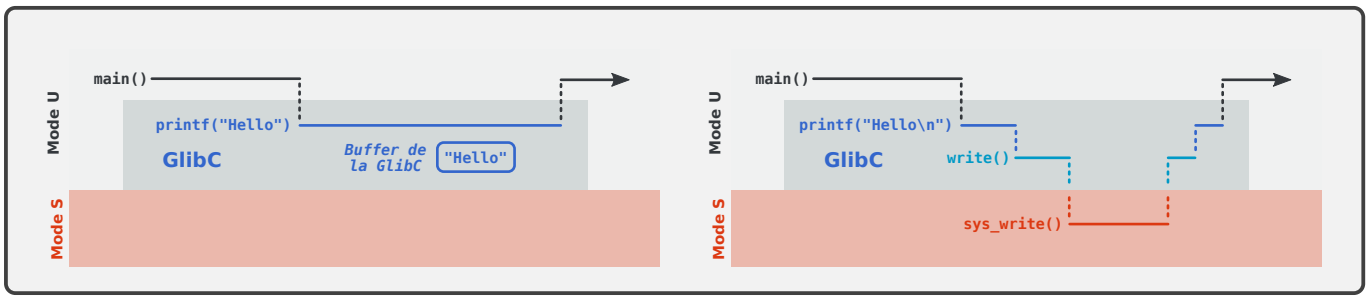
Pourquoi la commande `cd` ne peut être qu'une commande *builtin* ?

Cette commande modifie la valeur la variable `current_directory` si elle n'était pas *builtin* cette modification se ferait dans le fils. Le répertoire courant du père resterait donc inchangé.

Question 7 – 1 point

Dessinez les chronogrammes des appels de fonctions correspondant aux exécutions des deux lignes de *C* suivantes. Vos schémas devront faire apparaitre le mode d'exécution.

1. `printf("Hello");`
2. `printf("Hello\n ");`



Exercice 2 : Par ici la sortie (3 points)

Dans cet exercice on considère les deux scripts suivants :

lanceur.sh

```
#! /bin/bash

x="0"

./monScript.sh 10 0

source ./monScript.sh 20 1

## Question 2 :
# echo "\$\? = $?"

echo "x=$x"
```

monScript.sh

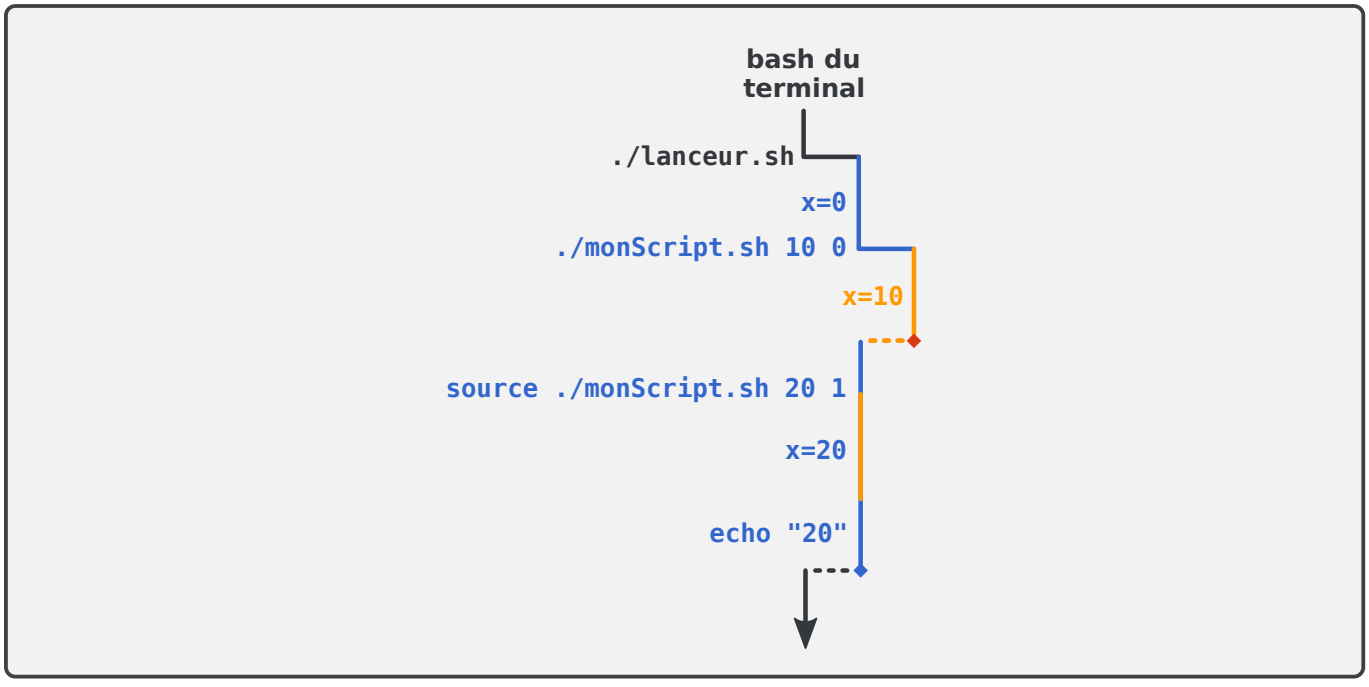
```
#! /bin/bash

x="$1"

## Question 2 :
# exit $2
```

Question 1 – 2 points

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



Question 2 – 1 point

Que s'affiche-t-il si l'on relance le script `lanceur.sh` après avoir décommenté la ligne de l `echo` dans ce dernier et la ligne de l `exit` dans `monScript.sh` ? Justifiez votre réponse.

Cette ligne ne s'affichera pas, car le dernier `exit` est exécuté par le processus qui exécute `lanceur.sh`. C'est donc ce dernier qui va s'arrêter. Il n'y aura donc pas d'affichage puisque le processus se terminera avant d'arriver sur la ligne décommentée.

Exercice 3 : Sacem (5 points)

Un agent de la Sacem traite un répertoire contenant des fichiers audio déposés par Jul. Un peu perdu, il décide de renommer ces fichiers avec un vrai langage de "djeunse". Il va donc télécharger le fichier `julForBoomer.txt`, contenant sur chaque ligne un mot et sa traduction séparés par un `;`. En voici un extrait :

julForBoomer.txt

```
...
deum;keum
hess;zermi
kichta;flouze
teum-teum;booster
thug;kaira
s;posse
...
```

Question 1 – 2 points

Dans un premier temps, il décide d'implémenter un script de traduction générique `translate.sh` qui a les spécifications suivantes :

- il prend en entrée un paramètre correspondant au mot à traduire
- il utilise un dictionnaire dont le chemin est contenu dans la variable d'environnement `DICO`
- en absence de paramètre, il retourne -1 et affiche un message d'usage (qui reste correct même si le script est renommé)
- si le dictionnaire n'est pas accessible, il retourne -2
- dans les autres cas il affiche la traduction trouvée dans le dictionnaire, ou rien si le mot recherché n'est pas présent dans le dictionnaire.

Donnez une implémentation d'un tel script. Vous ferez bien attention ici à avoir un script fonctionnel, quel que soit le dictionnaire et notamment à prendre en compte le cas où le mot cherché est inclus dans un autre (pare exemple `s` et `hess`).

translate.sh

```bash
#! /bin/bash

if [ $# -ne 1 ] ; then
  echo "Usage : $0 <dirName>"
  exit -1
fi

if [ ! -r $DICO ] ; then
  echo "Le dictionnaire '$DICO' "
  exit -2
fi

grep -E "^$1;" $DICO | cut -d ';' -f 2
```

Question 2 – 0,5 point

Sachant que le script sera enregistré dans le répertoire `~/scripts`. Comment peut-on s'assurer de pouvoir le lancer sans indiquer son chemin, quel que soit le répertoire courant ?

Il faut ajouter `~/scripts` à la variable `PATH`

Question 3 – 2 points

En utilisant votre script `translate.sh` (que l'on supposera ici fonctionnel) et le dictionnaire `~/julForBoomer.txt`, implémentez un nouveau script `renameForBoomer.sh` qui :

- renomme les fichiers et les répertoires du répertoire courant avec une traduction de son nom, si celui-ci est présent dans le dictionnaire ;
- affiche à la fin le nombre de renommages effectués.

renameForBoomer.sh

```bash
#! /bin/bash

export DICO=~/julForBoomer.txt

cpt=0
for file in * ; do
    vf=$(translate.sh $file)
    if [ -n "$vf" ] ; then
        mv $file $vf
        cpt=$((cpt+1))
    fi
done

echo $cpt
```

Question 4 – 0,5 point

Dans le dernier script, pourquoi serait-il incorrect de retourner le nombre fichiers renommés au lieu de l'afficher sur la sortie standard ?

La valeur retournée par un exit doit être inférieure à 256, ce qui dans notre cas n'est pas garanti. En pratique, elle doit être réservée au traitement des erreurs (0 si tout se passe bien).

Exercice 4 : Que de systèmes (7 points)

Dans cet exercice, on va étudier grâce à des commandes *Bash* les différents systèmes utilisés par les équipes de football en vue de la prochaine coupe du monde.

Pour l'ensemble de l'exercice, on suppose que l'on a accès à un fichier `systemes.csv` contenant des schémas tactiques utilisés par différents pays. Utilisant la virgule comme séparateur, il contient une première colonne avec les formations et une deuxième colonne avec le nom du pays l'ayant utilisée. Chaque schéma est lui composé d'une suite de chiffres, séparés par un tiret, dont la somme est égale à 10.

Voici un extrait d'un tel fichier :

Extrait de systemes.csv

```
3-5-2,Belgique
4-3-3,Espagne
6-3-1,Iles Féroé
3-4-3,Irlande
4-3-3,Portugal
5-4-1,Norvege
6-3-1,France
4-4-2,Turquie
4-3-3,Pays-Bas
4-3-3,Norvege
3-5-2,Belgique
4-1-4-1,Luxembourg
3-5-2,Serbie
4-4-2,France
```

Question 1 – 1 point

Proposez une commande qui affiche le nombre de schémas distincts dans le fichier.

```
moi@pc /home/moi $ cut -d , -f1 systemes.csv | sort | uniq | wc -l
```

Question 2 – 1 point

Proposez une commande qui affiche l'un des pays qui ont su nous faire rêver avec sa formation, *i.e.*, qui affiche l'un des pays qui a utilisé l'un des schémas ayant le plus grand premier chiffre (nombre de défenseurs)

```
moi@pc /home/moi $ sort systemes.csv | cut -d , -f 2 | tail -n 1
```

Question 3 – 1 point

Proposez une commande qui affiche la liste des systèmes ayant un schéma avec au moins deux chiffres consécutifs égaux (2 lignes proches avec le même nombre de joueurs). Votre commande n'affichera pas de doublons.

```
moi@pc /home/moi $ grep -E "([1-9])-\1.*," | sort | uniq
```

Question 4 – 1 point

Proposez une commande qui inverse les deux colonnes dans le fichier, *i.e.*, que chaque ligne soit composée d'un nom (s'il existe) suivit d'un schéma.

```
moi@pc /home/moi $ sed -E 's/(.+),(.*)/\2,\1/' systemes.csv
```

Question 5 – 1 point

Proposez une commande qui génère une nouvelle version du fichier (`systemes_v2.csv`) contenant toutes les lignes du fichier original triées par ordre alphabétique des noms de pays. Attention, les lignes doivent rester inchangées dans la nouvelle version.

```
moi@pc /home/moi $ sed -E 's/(.+),(.*)/\2,\1/' systemes.csv | sort | \
                              sed -E 's/(.+),(.*)/\2,\1/' > systemes_v2.csv
```

Question 6 – 2 points

Proposez un script qui affiche la liste de schémas qui apparaissent au moins quatre fois dans le fichier (4-3-3 dans notre extrait). Attention, vous n'afficherez que les schémas et sans doublons.

```
#! /bin/bash

for s in $(cut -d , -f 1 | sort | uniq); do
    [ $(grep $s systemes.csv | wc -l) -gt 3 ] && echo $s
done
```