ISS - Initiation aux Systèmes d'exploitation et au Shell
LU2IN020
Examen du 11 janvier 2022

Numéro d'anonymat :

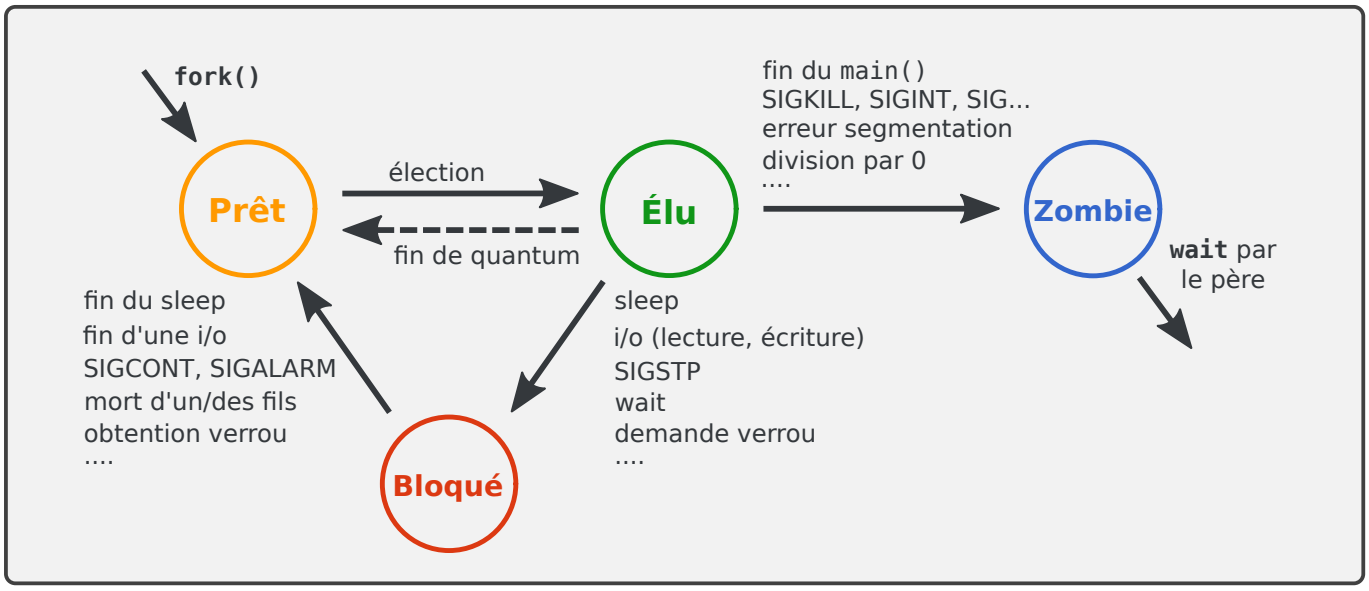
Aucun document autorisé pendant l'épreuve
Les téléphones portables, les montres connectées et autres appareils doivent être rangés dans votre sac.
Le barème n'est donné qu'à titre indicatif, pour vous permettre de juger de la difficulté des questions.
Attention : l'énoncé est imprimé recto-verso sur **8 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)

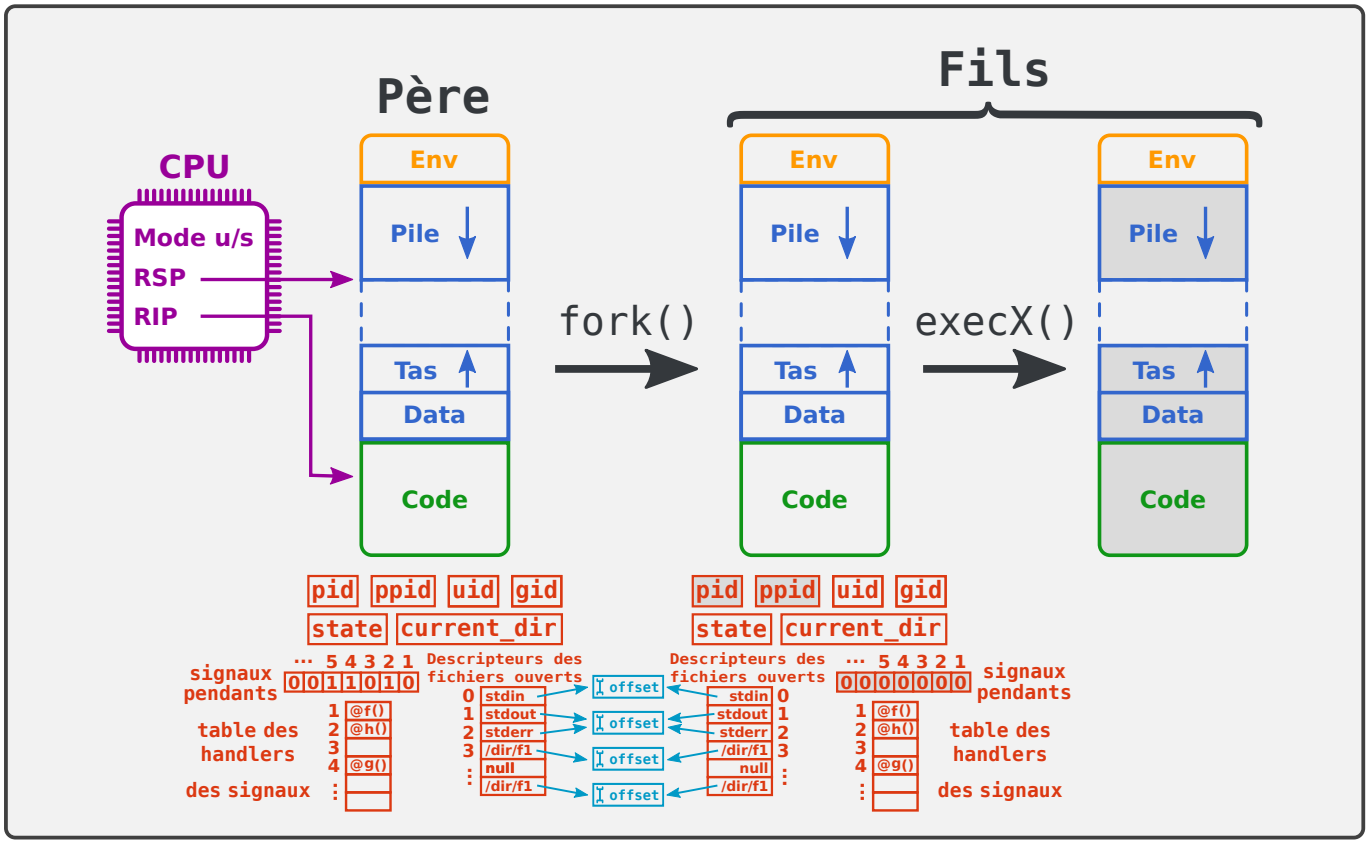
Question 1 – 1,5 point

Dessinez sous forme d'un automate le cycle de vie d'un processus depuis sa création jusqu'à sa complète disparition. Votre schéma devra indiquer l'ensemble des états possibles, ainsi qu'une raison pour chaque changement d'état. Votre schéma devra aussi différencier les systèmes fonctionnant en mode *temps partagé*, de ceux fonctionnant en mode *batch*.



Question 2 – 2 points

Dessinez 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. Puis représentez sur votre schéma le résultat d'un `fork` suivi d'un `exec` dans le fils.



Question 3 – 0,5 point

Qui est le deuxième processus à s'exécuter sur une machine ? Quels sont ses rôles ? Qui est son père ?

Il s'agit du processus *init* qui a pour père le *swapper*. Son rôle est double :
— tout d'abord il lance l'ensemble des services (démon) de la machine (par exemple : sshd, ntpd, cupsd, crond, ...)
— puis boucle sur un `wait()` pour arrêter les processus qu'il a adoptés lorsqu'ils sont passés à l'état *zombie*.

Question 4 – 0,5 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 5 – 0,5 point

En utilisant, la commande `strace` sur un simple `helloWorld.c`, on voit que ce dernier fait énormément d'appels systèmes, dont un `sys_write()` pour l'affichage. Pourtant, il n'y a pas d'appel explicite à cette fonction dans `helloWorld.c`. Où se trouve cet appel ? Comment s'appelle cette librairie ?

L'appel se trouve dans l'implémentation de la fonction `printf` qui appartient à la `GLibC`.

Question 6 – 1,5 point

Dessinez un chronogramme correspondant à l'exécution complète sur un multicoeur du script `./lanceur.sh` dans un terminal. Parmi tous les scénarios possibles, vous choisirez de représenter un de ceux qui conduisent à avoir le maximu de processus vivants :

— au moment du `wait`

— apres le `echo "fin du lanceur"`

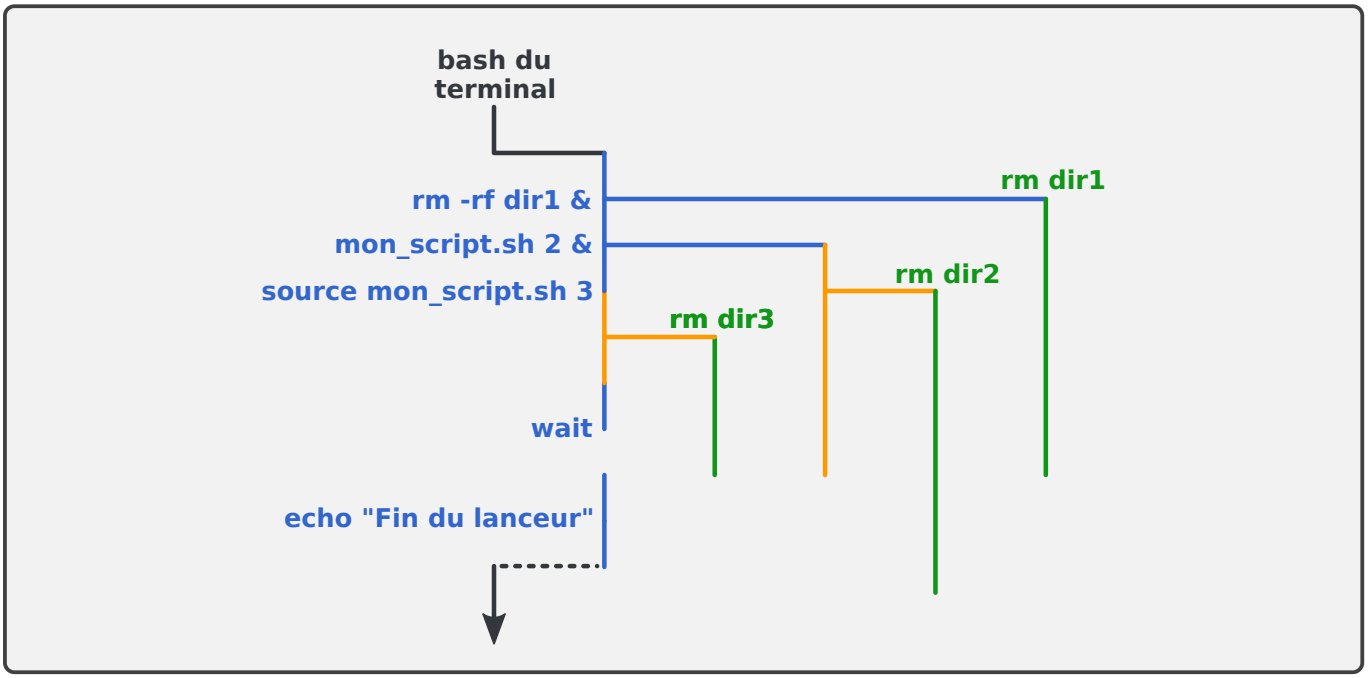
Pour simplifier, vous n'avez pas à représenter les fonctions C correspondant à l'implémentation du bash, ni les périodes où les processus passent dans l'état zombie. Mais votre schéma devra faire apparaître l'ensemble des processus créés, leurs liens de parenté, leurs éventuelles périodes de blocage.

lanceur.sh

```
#!/usr/bin/env bash

rm -rf dir1 &

./mon_script.sh 2 &

source mon_script.sh 3

wait

echo "fin du lanceur"
```

mon_script.sh

```
#!/usr/bin/env bash

rm -rf dir$1 &
```



Exercice 2 : Déploiement sur un parc de machines (4,5 points)

Dans cet exercice, on veut réaliser un script qui déploie un fichier sur un ensemble de machines de la PPTI. Pour cela, nous allons utiliser la commande `scp` qui permet de copier des fichiers entre machines. Son fonctionnement pour envoyer un fichier est le suivant :

```
scp <fichierLocal> <nomDeLaMachineDistante>:/tmp/
```

Pour automatiser cette tâche, on possède un fichier décrivant les salles machines. Ce fichier au format *csv* contient sur chaque ligne le nom d'une salle suivi du nombre de machines de cette salle (sur deux caractères). En voici un exemple :

sallesMachines.csv

```
301,13
302,12
303,16
401,12
402,08
501,04
502,12
503,23
```

Question 1 – 2 points

Avant de commencer à déployer le fichier, il nous faut obtenir le nom des machines. Hors, à la PPTI ceux-ci suivent le schéma suivant : *ppti-14-<numeroSalle>-<numeroMachine>* où les numéros de machine sont des entiers consécutifs allant de 01 jusqu'au nombre de machines de la salle.

Dans un premier temps, implémentez un script `listeMachines.sh` qui prend en paramètre le chemin vers le fichier `sallesMachines.csv` et qui affiche le nom de toutes les machines de toutes les salles de la PPTI avec une machine par ligne.

Pour simplifier, on supposera que le paramètre est correct et vous pourrez utiliser la commande `seq` pour générer une séquence de nombre, avec l'option `-w` qui assure que tous les chiffres comportent le même nombre de caractères. Par exemple `seq -w 04` affiche `01 02 03 04`.

listeMachines.sh

```
#! /bin/bash

while IFS=, read salle nbMachine ; do
  for id in $(seq -w $nbMachine) ; do
    echo ppti-14-$salle-$id
  done
done < $1
```

listeMachines.sh alternative sans utiliser IFS

```bash
#! /bin/bash

while read line ; do
  salle=$(echo $line | cut -d , -f 1)
  nbMachine=$(echo $line | cut -d , -f 2)
  for id in $(seq -w $nbMachine) ; do
    echo ppti-14-$salle-$id
  done
done < $1
```

Question 2 – 0,5 point

Sachant que le script `listeMachines.sh` sera enregistré dans le répertoire `~/scripts`, comment peut-on s'assurer de pouvoir le lancer sans avoir à 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 `listeMachines.sh` (que l'on supposera ici fonctionnel) et le fichier `~/sallesMachines.csv`, implémentez un nouveau script `deploiement.sh` qui prend en paramètre le nom d'un fichier local et utilise la commande `scp` pour le copier dans le répertoire `/tmp` de toutes les machines de la PPTI.

Pour plus de performances, votre implémentation lancera tous les `scp` en concurrence, puis attendra que les envois soient terminés avant d'afficher le message `"Fin du déploiement"`.

En cas d'absence de paramètre, ou si ce paramètre ne correspond pas à un fichier, votre script devra afficher un message d'erreur et retourner 1.

deploiement.sh

```bash
#! /bin/bash

if [ $# -ne 1 -o ! -f $1 ] ; then
  echo "Usage : $0 <unFichier>"
  exit -1
fi

for machine in $(./listeMachines.sh ./sallesMachine.csv) ; do
  scp $1 $machine:/tmp/ &
done

wait

echo "Fin du deploiement"
```

Exercice 3 : Alea jacta est (6 points)

Dans cet exercice, on veut implémenter une commande de jeté de dé. Nous allons donc devoir concevoir un mécanisme pseudo-aléatoire.

Question 1 – 2 points

Pour commencer, implémentez un script `boucleValeur.sh` qui initialise une variable à 0, puis l'incrémente de 1 infiniment, en la ramenant à zéro dès quelle dépasse une valeur passée en paramètre.

Ce script devra en outre afficher la valeur actuelle de cette variable à chaque réception d'un signal `SIGUSR1`.

Pour simplifier, on supposera que le script est toujours appelé avec un entier en paramètre.

boucleValeur.sh

```
#! /bin/bash

trap 'echo $value' SIGUSR1

value=0

while true ; do
    value=$(( ( value + 1 ) % $1 ))
done
```

Question 2 – 0,5 point

Pouvions-nous choisir `SIGKILL` à la place de `SIGUSR1` dans la question précédente ? Pourquoi.

Non, on ne pouvait pas, car on ne peut pas redéfinir le handler du signal `SIGKILL`.

Question 3 – 2 points

Grâce à ce premier script `boucleValeur.sh` (que l'on supposera correct), implémentez un script `sixFaces.sh` qui lit infiniment sur le clavier et qui à chaque saisie lue envoie un signal `SIGUSR1` à un processus exécutant `boucleValeur.sh` pour faire apparaître un entier compris entre 0 et 5.

sixFaces.sh

```
#! /bin/bash

./boucleValeur.sh 6 &
pidBoucle=$!

while true ; do
    read
    kill -SIGUSR1 $pidBoucle
done
```

Question 4 – 0,5 point

Pour arrêter un processus exécutant votre implémentation du script `sixFaces.sh`, il peut être tentant de faire un `Crtl+C` (qui envoie un SIGINT au processus d'avant plan). Si cela conduit bien à l'arrêt du processus visé, cela n'est pas très propre pourquoi ?

Si le processus `sixFaces.sh` va bien s'arrêter (comportement par défaut lorsque l'on reçoit un `SIGINT`), il n'en est pas de même pour son fils qui exécute le script `boucleValeur.sh`. Ce dernier va alors tourner infiniment en tâche de fond.

Question 5 – 1 point

Proposez une solution permettant une terminaison propre lorsque l'on fait un `Ctrl+C` après avoir lancé `sixFaces.sh`. Vous pouvez, au besoin, modifier aussi le script `boucleValeur.sh`.

sixFaces.sh avec terminaison propre

```
#! /bin/bash

trap 'kill -SIGINT $pidBoucle && exit 0' SIGINT

./boucleValeur.sh 6 &
pidBoucle=$!

while true ; do
    read
    kill -SIGUSR1 $pidBoucle
done
```

Exercice 4 : Que de stats (3 points et 2 points bonus)

Dans cet exercice, on va étudier les chiffres de la vaccination par classe d'âge et par vaccin. Pour ce faire, on va travailler sur le fichier `stats.csv` généré ce 2 janvier à partir d'un fichier `csv` plus complet chargé depuis le site web https ://datavaccin-covid.ameli.fr

Utilisant la virgule comme séparateur, il contient :

1. le nombre de personnes total appartenant à une tranche d'âge
2. la tranche d'âge concernée
3. un nom de vaccin ou "Tous vaccins" dans le cas des statistiques tous vaccins confondus
4. le pourcentage de personnes de la classe d'âge qui ont commencé un cycle de vaccination avec le vaccin concerné
5. le pourcentage de personnes de la classe d'âge qui ont terminé leur cycle de vaccination avec le vaccin concerné

Notons que pour simplifier l'exercice, les pourcentages ont été arrondis et que tous les entiers d'une colonne sont écrits avec le même nombre de caractères. Vous pouvez donc utiliser la commande `sort` sans vous préoccuper du fait qu'il s'agisse d'une valeur numérique.

Voici un extrait du fichier `stats.csv` :

Extrait de stats.csv

```
09500311,de 0 à 11 ans,COMIRNATY Pfizer-BioNTech,00%,00%
09500311,de 0 à 11 ans,COVID-19 Vaccine Moderna,00%,00%
09500311,de 0 à 11 ans,VAXZEVRIA AstraZeneca,00%,00%
09500311,de 0 à 11 ans,COVID-19 Vaccine Janssen,00%,00%
09500311,de 0 à 11 ans,COMIRNATY Pfizer-BioNTech pédiatrique,01%,00%
09500311,de 0 à 11 ans,Tous vaccins,01%,00%
05032773,de 12 à 17 ans,COMIRNATY Pfizer-BioNTech,77%,74%
05032773,de 12 à 17 ans,COVID-19 Vaccine Moderna,04%,03%
05032773,de 12 à 17 ans,VAXZEVRIA AstraZeneca,00%,00%
...
07375725,de 65 à 74 ans,COVID-19 Vaccine Janssen,04%,04%
07375725,de 65 à 74 ans,COMIRNATY Pfizer-BioNTech pédiatrique,00%,00%
07375725,de 65 à 74 ans,Tous vaccins,95%,94%
06358175,75 ans et plus,COMIRNATY Pfizer-BioNTech,74%,73%
06358175,75 ans et plus,COVID-19 Vaccine Moderna,09%,09%
06358175,75 ans et plus,VAXZEVRIA AstraZeneca,06%,06%
06358175,75 ans et plus,COVID-19 Vaccine Janssen,02%,02%
06358175,75 ans et plus,COMIRNATY Pfizer-BioNTech pédiatrique,00%,00%
06358175,75 ans et plus,Tous vaccins,92%,90%
```

Question 1 – 1 point

Proposez une commande qui affiche la classe d'âge de ce document la moins nombreuse.

```
sort /tmp/stats.csv | head -n1 | cut -d, -f2
```

Question 2 – 1 point

Proposez une commande qui affiche le nombre de vaccins présents dans ce document.

```
cut -d, -f3 /tmp/stats.csv | sort | uniq | grep -v "Tous␣vaccins" | wc -l
```

Question 3 – 1 point

Proposez une commande qui permet de vérifier s'il existe une classe d'âge pour laquelle, tous vaccins confondus, toutes les personnes ayant commencé un cycle de vaccination l'ont terminé.

```
grep -E 'Tous␣vaccins,.*(..)%,\1%' /tmp/stats.csv
```

Question 4 – 2 points (Bonus)

Et pour terminer ce semestre en beauté, proposez une commande qui affiche la classe d'âge qui, tous vaccins confondus, a en proportion le plus de personnes ayant terminé leur cycle de vaccination.

```
sed -n -E 's/(.*),(..)%/\2,\1/;␣/Tous␣vaccins/p' /tmp/stats.csv | sort | tail -1 | cut -d, -f3
```

