ISS - Initiation aux Systèmes d'exploitation et au Shell
LU2IN020
Examen du 19 janvier 2021

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

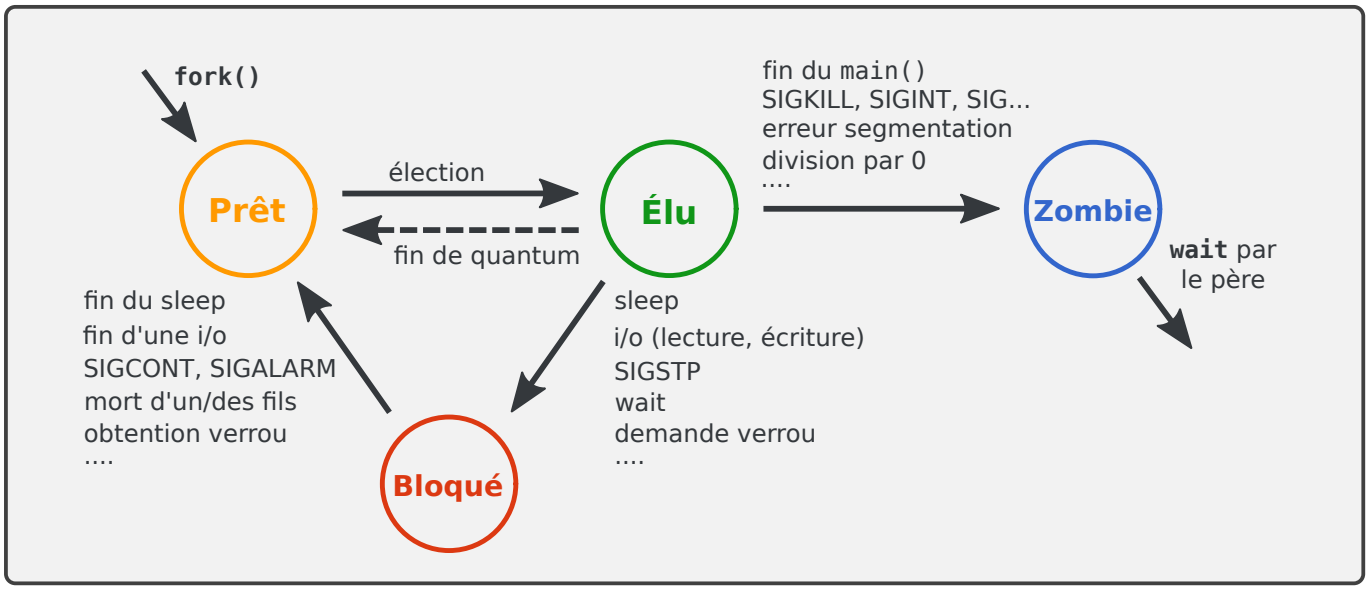
Aucun document autorisé pendant l'épreuve
Les téléphones portables, les montres connectées et autres appareils doivent être rangés dans votre sac.
Le barème n'est donné qu'à titre indicatif, pour vous permettre de juger de la difficulté des questions.
Attention : l'énoncé est imprimé recto-verso sur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

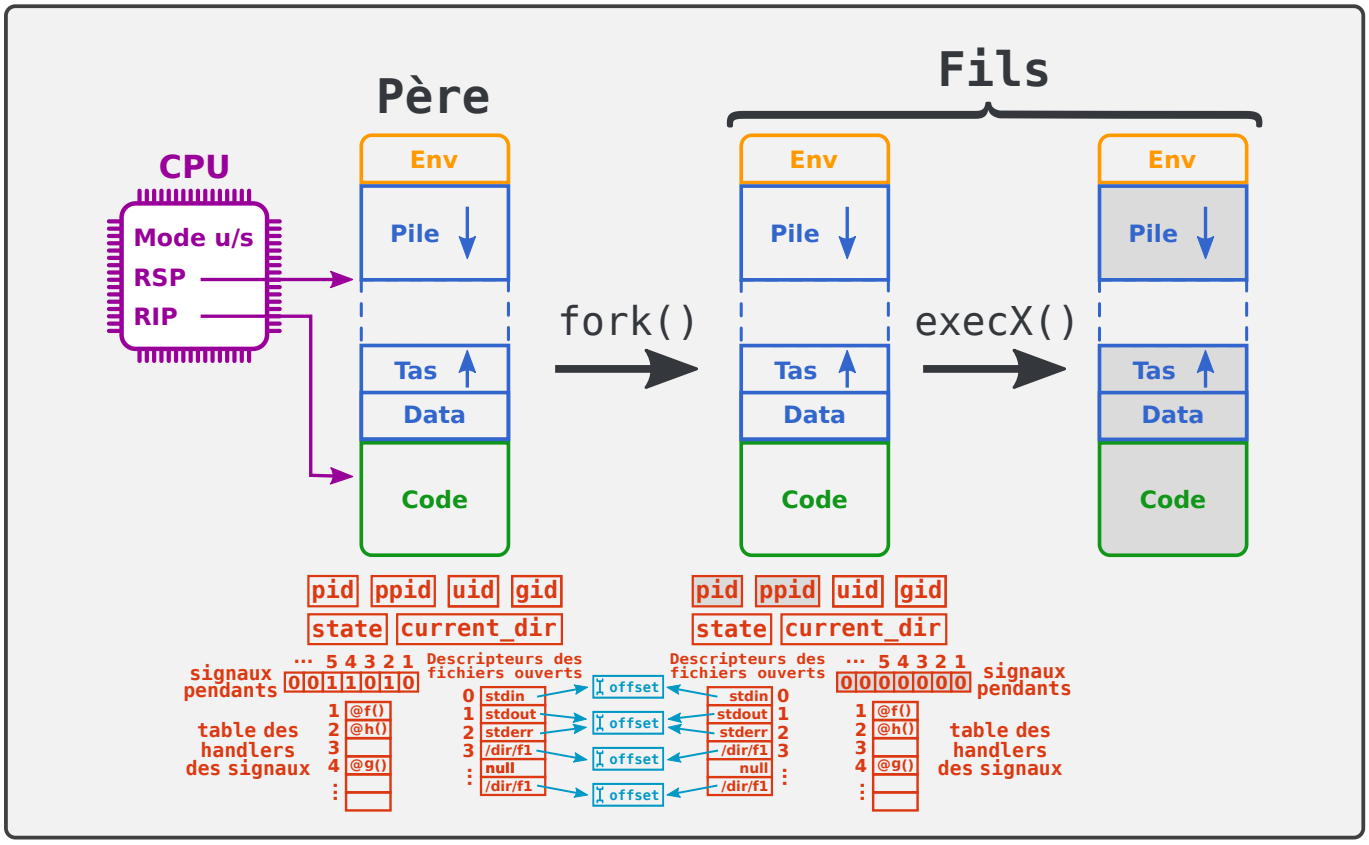
Question 1 – 1,5 point

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 – 3 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 – 1 point

En déroulant son exécution, donnez le résultat de la dernière commande :

```
moi@pc /home/moi $ pwd
/home/moi
moi@pc /home/moi $ chmod u+x maCommande.sh
moi@pc /home/moi $ cat maCommande.sh
#! /bin/bash

cd ..

if [ $# -eq 1 ] ; then
 $0
fi
moi@pc /home/moi $ ./maCommande.sh 3
```

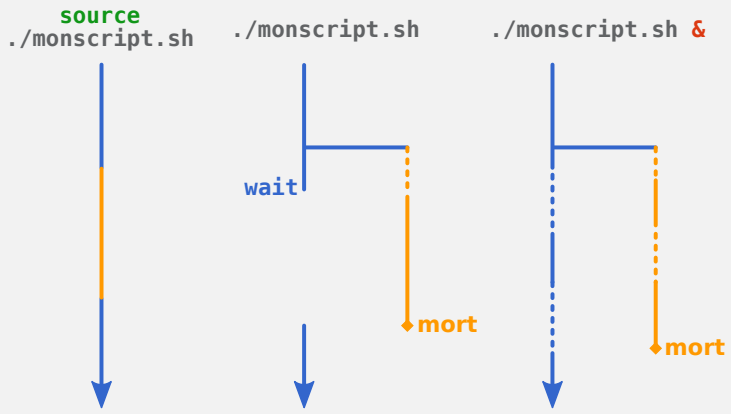
Il n'y a pas de problèmes de droits d'exécution. La commande s'exécute donc une fois et la condition est vraie puisqu'il y a un paramètre. Malheureusement cet appel échoue, on a changé de répertoire et la variable $0 est un chemin relatif.

```
moi@pc /home/moi $ ./maCommande.sh 1
./maCommande.sh: ligne 6: ./maCommande.sh: Aucun fichier ou dossier de ce type
```

Question 4 – 1 point

Faites 3 chronogrammes illustrant la différence entre les trois utilisations de `monScript.sh` suivantes. Vous supposerez ici que ces trois commandes sont faites sur un ordinateur n'ayant qu'un seul cœur de calcul :

1. `./monScript.sh`
2. `./monScript.sh &`
3. `source ./monScript.sh`



Question 5 – 1 point

Parmi celles présentées en cours, citez deux raisons de ne pas utiliser des liens physiques ?

Les liens physiques présentent plusieurs limites dont :
— impossibilité de faire des liens physiques entre deux partitions ;
— impossibilité de faire plus d'un lien physique vers un répertoire ;
— difficulté pour recenser tous les liens physiques d'un fichier en vue de sa suppression.

Question 6 – 1,5 point

On considère ici un système dont la variable `rootdir` pointe vers l'inode 8. Le contenu des fichiers correspondants aux inodes 8, 12, 18 et 23 est donné dans la figure ci-dessous.

En modifiant ce schéma, donnez l'état du système de fichiers après l'exécution des trois commandes suivantes. Vous pouvez le cas échéant rayer une ligne ou choisir librement un nouveau numéro d'inode.

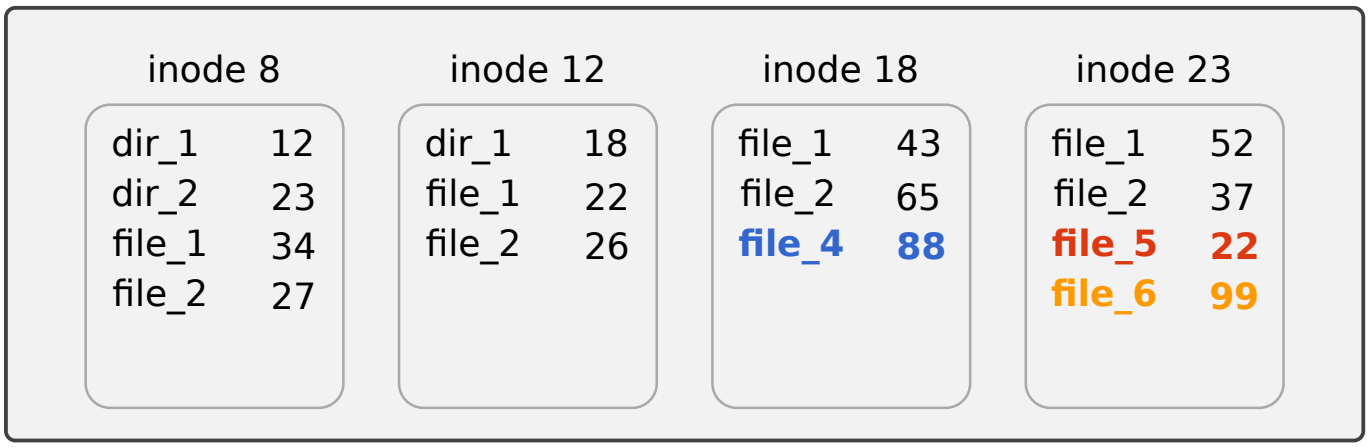
```
moi@pc / $ cp /file_1 /dir_1/dir_1/file_4
moi@pc / $ ln /dir_1/file_1 /dir_2/file_5
moi@pc / $ ln -s /dir_1/file_2 /dir_2/file_6
```

inode 8	
dir_1	12
dir_2	23
file_1	34
file_2	27

inode 12	
dir_1	18
file_1	22
file_2	26

inode 18	
file_1	43
file_2	65

inode 23	
file_1	52
file_2	37



Exercice 2 : Pour quelques btc de plus (3 points)

L'équipe pédagogique d'ISS souhaite se lancer dans le business des banques en ligne. Elle va donc devoir manipuler des transactions bancaires contenues dans le fichier `bank_records.csv` dont voici un extrait :

bank_records.csv

```
Date,Description,Deposits,Withdrawls,Location
18-Jan-2021,EDF,00,"109,10",Bank
18-Jan-2021,Steam,00,60,Web
18-Jan-2021,Cheque,250,00,Toulouse
18-Jan-2021,Sorbonne,00,"480,60",Paris5
18-Jan-2021,CROUS,"550,10",00,Paris5
```

Question 1 – 0,5 point

Pour des raisons de confidentialité, on souhaite ne conserver que les colonnes 'Deposits' et 'Withdrawls'. Expliquez pourquoi la commande suivante donne des résultats inattendus.

```
moi@pc /home/moi $ cut -d',' -f 3,4 bank_records.csv
```

Certaines valeurs numériques contiennent des virgules. Cela va fausser le découpage en colonne. Sur l'exemple on obtient :

bank_records.err.csv

```
Deposits,Withdrawls
00,"109
00,60
250,00
00,"480
"550,10"
```

Question 2 – 2,5 points

Proposez un script `anonym.sh` qui prend en paramètre un nom de fichier suffixé par `.csv` qu'il anonymisera en préservant uniquement les colonnes citées précédemment. Le résultat devra être stocké dans un fichier de même nom, mais dont le suffixe `.csv` a été remplacé par `.ano.csv`.

Votre implémentation devra, en outre, vérifier la présence du paramètre et des droits en lecture sur le fichier. En cas d'erreur il devra afficher un message d'usage du script.

Sur le fichier donné en exemple le résultat devrait être :

bank_records.ano.csv

```
Deposits,Withdrawls
00,"109,10"
00,60
250,00
00,"480,60"
"550,10",00
```

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

if [ $# -ne 1 -o ! -r $1 ] ; then
  echo "Usage : $0 <name.csv>"
  exit -1
fi

SRC=$1
DST=${SRC%.csv}.ano.csv

sed -E 's/("[^"]*),([^"]*")/\1.\2/g' $SRC | \
   cut -d',' -f 3,4  | \
   sed -E 's/("[^"]*)\.([^"]*")/\1,\2/g' > $DST
```

Exercice 3 : C'est pas sorcier (9 points)

Avec le confinement, un étudiant désœuvré décide de mener des expérimentations pour vérifier et approfondir ce qui a été vu en cours d'ISS.

Afin de pouvoir les reproduire rapidement, chaque expérience devra être lancée par un unique script qui déroulera l'expérience et affichera le résultat à la fin de celle-ci. Toutes les techniques apprises ce semestre pourront être librement utilisées : sous processus, variables, environnement, récupération de la sortie standard d'une commande, redirection, création et/ou écriture dans un fichier, signaux, ...

Les questions sont indépendantes et leurs solutions sont assez simples. Mais attention les fausses pistes peuvent être nombreuses. Prenez donc le temps de réfléchir avant de vous lancer, tout en sachant qu'une solution partielle reste mieux que pas de solution du tout.

Question 1 – 3 points

Pour commencer, il se demande s'il est possible de vérifier l'impossibilité de compter les signaux reçus, telle que présentée en cours.

Implémentez un script `lostSig.sh` qui enverra un certain nombre de signaux à un fils qui exécutera, entre autres, cinq `sleep` d'une seconde. Une fois le fils terminé, `lostSig.sh` affichera suivant les constatations l'une des deux phrases suivantes :

— `"Le fils a bien traité tous les signaux envoyés"`
— `"Le fils n'a pas traité tous les signaux envoyés"`

inherit.sh

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

testFile=/tmp/test

rm -f $testFile

trap "touch $testFile" SIGTERM

./inherit-child-with_exit.sh &

sleep 1

kill -SIGTERM $!

wait

if [ -f $testFile ] ; then
  echo "les handlers du père sont hérités par le fils"
else
  echo "le fils démarre avec les handlers par défaut"
fi
```

inherit-child.sh

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

echo "Debut du fils"

sleep 3

echo "Fin du fils"
```

Question 2 – 3 points

Dans un deuxième temps, l'étudiant se demande ce qu'il advient des handlers de signaux après un `fork`. Sont-il hérités du père ou sont-ils replacés aux handlers par défaut.

Implémentez vous aussi des scripts `inherit.sh` et `inherit-child.sh` qui permettent de répondre à la question. Comme précédemment, l'expérience sera lancée au travers d'un script (`inherit.sh`) qui sera chargé d'afficher la conclusion c'est-à-dire l'une des deux phrases suivantes :

— `"Les handlers du père sont hérités par le fils"`
— `"Le fils démarre avec les handlers par défaut"`

lostSig.sh

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

NB_SIG=5
FILE=/tmp/nbSigFile

./lostSig-child.sh > $FILE &

pid_fils=$!

sleep 1

for i in $(seq 1 $NB_SIG) ; do
  kill -SIGTERM $pid_fils
done

wait

cat $FILE

if grep "^$NB_SIG\$" $FILE ; then
  echo "Le fils a bien traité tous les signaux"
else
  echo "Le fils n'a pas traité tous les signaux"
fi
```

lostSig-child.sh

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

cpt=0

trap "cpt=$((cpt+1))" SIGTERM

for i in {1..5} ; do
  sleep 1
done

echo $cpt
```

Question 3 – 3 points

Pour finir, l'étudiant s'interroge sur la présence de processus adoptés sur sa machine. Il va donc implémenter un script `adopt.sh` qui affichera le nombre de ces processus.

Ayant suivi attentivement son cours, il sait que sur certaines machines ces adoptions ne sont pas faites par le processus `init`. Son script d'expérimentation cherche donc, dans un premier temps, le `pid` du processus en charge des adoptions.

Donnez à votre tour une solution complète qui affiche sur la console le nombre de processus adoptés. Votre programme pourra utiliser la commande `ps` avec l'option `ppid:1,cmd:1`, dont voici un exemple d'exécution :

```
moi@pc /home/moi $ ps o ppid:1,cmd:1
PPID CMD
569 /bin/sh /usr/bin/startx
304 bash
743 ./unTest.sh &
743 ./unTest.sh &
1 /usr/bin/gnome-keyring-daemon --start --components=pkcs11,secrets,ssh
11596 pavucontrol
14012 soffice 2020-partiel-notes.ods
16962 make 2020-examen-correction.pdf
30488 bash
28321 ps o ppid:1,cmd:1
```

adopt.sh

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

./adopt-child.sh

pid_adopteur=$(ps o ppid:1,cmd:1 | grep -v grep |\
                                   grep 'adopt-subchild.sh' |\
                                   head -n 1 |\
                                   cut -d ' ' -f1)

ps o ppid:1,cmd:1 | grep "^$pid_adopteur " | wc -l # ou grep -c
```

adopt-avec_pgrep.sh

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

./adopt-child.sh

pid_adopteur=$(ps o ppid:1 $(pgrep -f 'adopt-subchild.sh') | tail -n 1)

pgrep -c --parent $pid_adopteur
```

adopt-child.sh

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

./adopt-subchild.sh &
```

adopt-subchild.sh

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

/bin/sleep 10
```