# ISS - Initiation aux Systèmes d'exploitation et au Shell
# LU2IN020
# Partiel du 20 novembre 2021

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

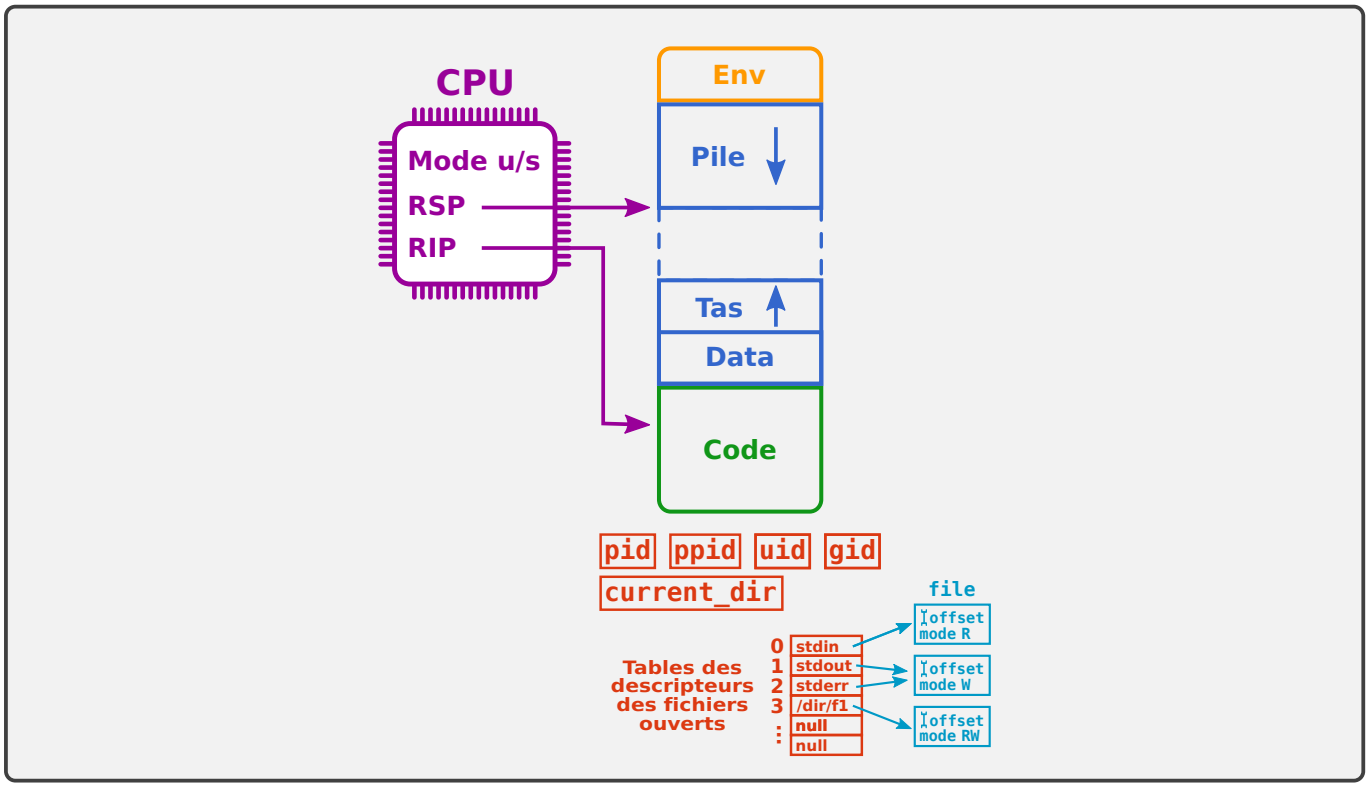
**Aucun document autorisé pendant l'épreuve**
Les téléphones portables, les montres connectées et autres appareils doivent être rangés dans votre sac.
Le barème n'est donné qu'à titre indicatif, pour vous permettre de juger de la difficulté des questions.
Attention : l'énoncé est imprimé recto-verso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

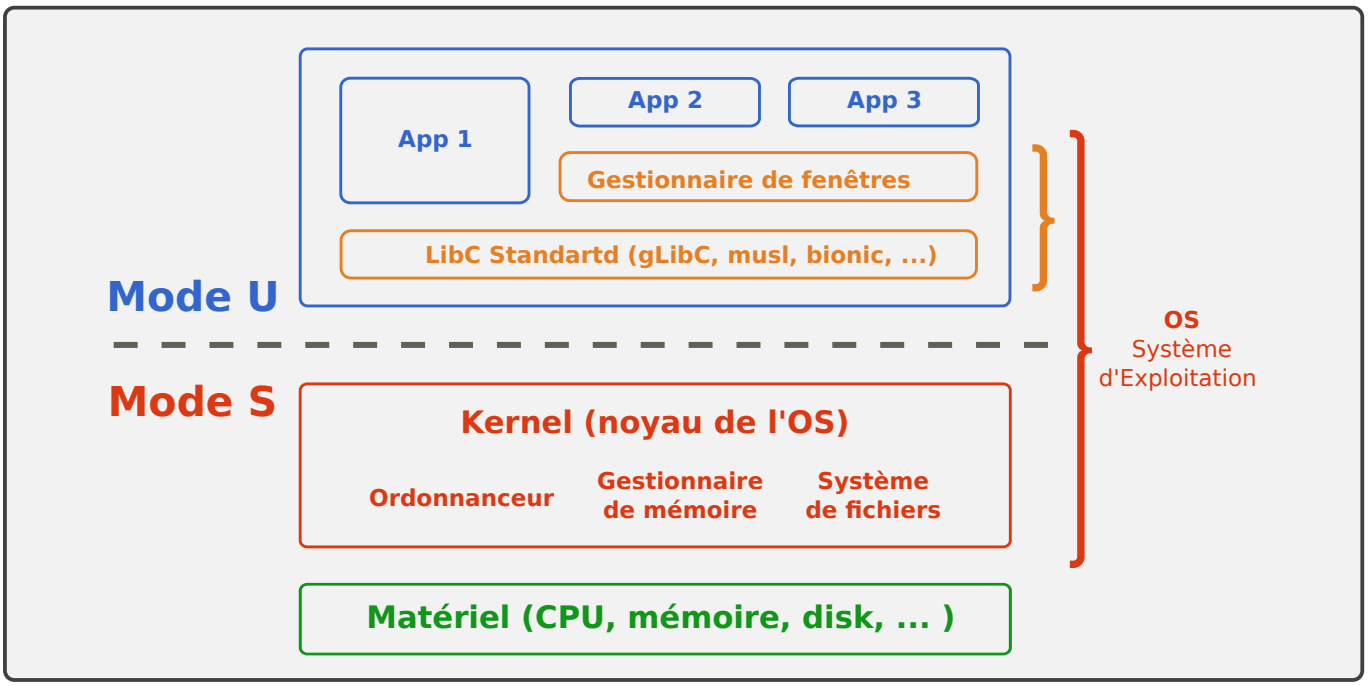
### Question 1 – 1,5 point

Dessinez, 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 apparaitre 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 ?

Cet entier est un descripteur de fichier. Il correspond à l'index dans la table des fichiers ouverts du kernel. On le voit en *Bash* dans l'opérateur `2>` où 2 correspond à `stderr`.

### Question 4 – 0,5 point

Quelle est la définition du cours du mot processus ?

Un processus c'est un espace d'adressage.Il contient un ou plusieurs fils d'exécution (thread).

### Question 5 – 0,5 point

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

Le noyau d'un système d'exploitation est composé de l'ensemble de son code qui s'exécute en mode S.

### 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 ?

Lors de l'exécution d'un appel système le processeur passe en mode S et à tous les pouvoirs, cependant le code de cet appel système n'a pas été écrit par l'utilisateur et n'est pas modifiable par lui. Il a donc tous les pouvoirs, mais uniquement pour faire ce qui a été prévu.

### Question 7 – 0,5 point

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

Cette adresse correspond à une adresse virtuelle et non une adresse physique. Sa limite ne dépend donc pas de la taille de la RAM, mais uniquement de celle du *mot mémoire* de l'architecture.

### Question 8 – 0,5 point

Qui définit la valeur du *shebang* ?

Le *shebang* qui s'écrit en `#!` quel que soit le langage de script utilisé est en fait un nombre magique : (`0x23 0x21 -> 100011100001 -> 2273`). Il est fixé en dur par le kernel, mais peut être changé en le recompilant. Il est utilisé dans l'appel système `sys_execve()` pour distinguer un exécutable d'un script à interpréter.

## 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 chaines 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`.

**ping.sh**

```bash
#! /bin/bash

echo "ping"

[ "$1" -gt 1 ] && $(dirname $0)/pong.sh $(($1 - 1))
```

**pong.sh**

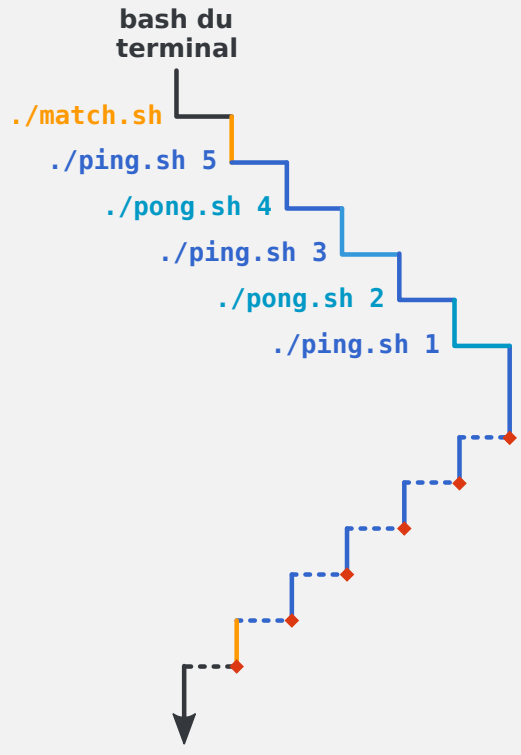
```bash
#! /bin/bash

echo "pong"

[ "$1" -gt 1 ] && $(dirname $0)/ping.sh $(($1 - 1))
```

## 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.

**match_v2.sh**

```bash
#! /bin/bash

export NB_ECHANGES=5

./mesScripts/ping_v2.sh
```

**ping_v2.sh**

```bash
#! /bin/bash

echo "ping"

NB_ECHANGES=$((NB_ECHANGES - 1))

[ "$NB_ECHANGES" -gt 0 ] && $(dirname $0)/pong_v2.sh
```

**pong_v2.sh**

```bash
#! /bin/bash

echo "pong"

NB_ECHANGES=$((NB_ECHANGES - 1))

[ "$NB_ECHANGES" -gt 0 ] && $(dirname $0)/ping_v2.sh
```

## 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.

**my_tee.sh**

```bash
#! /bin/bash

[ $# -eq 0 ] && echo "il faut au moins un nom de fichier" && exit 1

while read line ; do
    for f in $@ ; do
      echo $line >> $f
    done
    echo $line
done
```

## 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
```

La chaine de tube produit le comportement suivant :
- Le premier `my_tee.sh` ajoute `"LU2IN020 - Partiel - Amphi 1"` à la fin des fichiers `etudiant_A` et `etudiant_B`
- Le tr transforme la chaine en `"LU2IN020 - Partiel - Amphi 2"`
- Le premier `my_tee.sh` ajoute `"LU2IN020 - Partiel - Amphi 1"` à la fin des fichiers `etudiant_A` et `etudiant_B`
- la redirection finale vers `/dev/null` évite d'afficher la chaine sur le terminal

## 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_examen 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.

**convocations.sh**

```
#! /bin/bash

while read ue examens ; do
  for exam in $examens ; do
    echo "$ue - $exam - Amphi 1" | ./my_tee.sh $(cat $2) | tr 1 2 \
                                 | ./my_tee.sh $(cat $3)  > /dev/null
  done
done < $1
```

## 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.

```
moi@pc /home/moi $ tail -n +2 comptes.txt | cut -d / -f1 | sort | uniq | wc -l
```

### Question 2 – 1 point

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

```
moi@pc /home/moi $ tail -n +2 comptes.txt | cut -d : -f 2 | sort | tail -n 1
```

### Question 3 – 1 point

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

```
moi@pc /home/moi $ grep -E ',([0-9]+):\1' comptes.txt | cut -d , -f 1
```

### 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).

```
#! /bin/bash

file=comptes.txt

for y in $(tail -n +2 $file | cut -d / -f 1 | sort | uniq); do
    [ "$(grep $y $file | wc -l)" -ne 12 ] && echo -n "$y "
done
echo
```

## 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.

**match.sh**

```
#! /bin/bash

./mesScripts/king.sh 5
```

**ping.sh ou pong.sh ou king.sh ou kong.sh**

```
#! /bin/bash

basename $0 | cut -d . -f 1

[ "$1" -gt 1 ] && $(dirname $0)/$(basename $0 | tr io oi) $(($1 - 1))
```