# Exercices sur les processus

**Exercice 1 :**

**Partie A :** Pour chaque question, entourer la bonne réponse.

1) Parmi les commandes ci-dessous, laquelle permet d'afficher les processus en cours d'exécution ?
   a) `dir` b) `ps` c) `man` d) `ls`

2) Quelle abréviation désigne l'identifiant d'un processus dans un système d'exploitation de type UNIX ?
   a) PIX b) SIG c) PID d) SID

3) Comment s'appelle la gestion du partage du processeur entre différents processus ?
   a) L'interblocage b) L'ordonnancement
   c) La planification d) La priorisation

4) Quelle commande permet d'interrompre un processus dans un système d'exploitation de type UNIX ?
   a) `stop` b) `interrupt` c) `end` d) `kill`

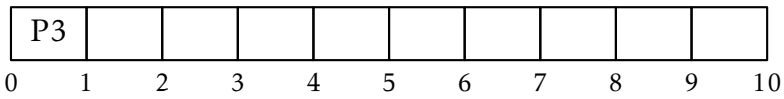
**Partie B :**

1) Un processeur choisit à chaque cycle d'exécution le processus qui doit être exécuté. Le tableau ci-dessous donne pour trois processus P1, P2, P3 :
   - la durée d'exécution (en nombre de cycles),
   - l'instant d'arrivée sur le processeur (exprimé en nombre de cycles à partir de 0),
   - le numéro de priorité.

   Le numéro de priorité est d'autant plus petit que la priorité est grande. On suppose qu'à chaque instant, c'est le processus qui a le plus petit numéro de priorité qui est exécuté, ce qui peut provoquer la suspension d'un autre processus, lequel reprendra lorsqu'il sera le plus prioritaire.

| Processus | Durée d'exécution | Instant d'arrivée | Numéro de priorité |
|---|---|---|---|
| P1 | 3 | 3 | 1 |
| P2 | 3 | 2 | 2 |
| P3 | 4 | 0 | 3 |

   Compléter le tableau ci-dessous en indiquant dans chacune des cases le processus exécuté à chaque cycle.




2) On suppose maintenant que les trois processus précédents s'exécutent et utilisent une ou plusieurs ressources parmi R1, R2 et R3.
   Parmi les scénarios suivants, lequel provoque un interblocage ? Justifier.

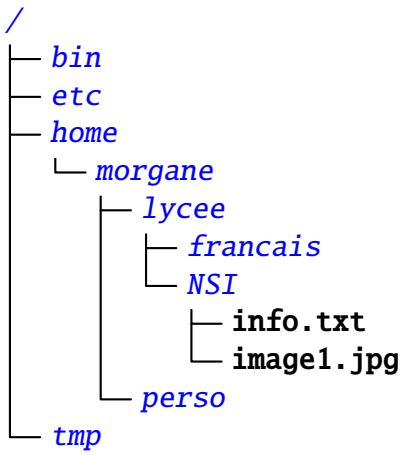
| Scénario 1 |
|---|
| P1 acquiert R1 |
| P2 acquiert R2 |
| P3 attend R1 |
| P2 libère R2 |
| P2 attend R1 |
| P1 libère R1 |

| Scénario 2 |
|---|
| P1 acquiert R1 |
| P2 acquiert R3 |
| P3 acquiert R2 |
| P1 attend R2 |
| P2 libère R3 |
| P3 attend R1 |

| Scénario 3 |
|---|
| P1 acquiert R1 |
| P2 acquiert R2 |
| P3 attend R2 |
| P1 attend R2 |
| P2 libère R2 |
| P3 acquiert R2 |

**Exercice 2 :** *Cet exercice porte sur les systèmes d'exploitation et la gestion des processus par un système d'exploitation.*
Cet exercice pourra utiliser des commandes de systèmes d'exploitation de type UNIX telles que `cd`, `ls`, `mkdir`, `rm`, `rmdir`, `mv` et `cat`.

1) Dans un système d'exploitation de type UNIX, on considère l'arborescence des fichiers suivante dans laquelle les noms de dossiers sont en italique et ceux des fichiers sont en gras.
On souhaite grâce à l'utilisation du terminal de commande, explorer et modifier les répertoires et fichiers présents.
On suppose qu'on se trouve actuellement à l'emplacement `/home/morgane`.




   a) Parmi les quatres propositions suivantes, donner celle correspondant à l'affichage obtenu lors de l'utilisation de la commande `ls`.

   - **Proposition 1** : `lycee francais NSI info.txt image1.jpg perso`
   - **Proposition 2** : `lycee perso`
   - **Proposition 3** : `morgane`
   - **Proposition 4** : **`bin`** `etc home tmp`

   b) Écrire la commande qui permet, à partir de cet emplacement d'atteindre le répertoire `lycee`.

   On suppose maintenant qu'on se trouve dans le répertoire `/home/morgane/lycee/NSI`.

   c) Écrire la commande qui permet de créer à cet emplacement un répertoire nommé `algorithmique`.

   d) Écrire la commande qui permet, à partir de cet emplacement, de supprimer le fichier `image1.jpg`.

2) On rappelle qu'un processus est une instance d'application. Un processus peut être démarré par l'utilisateur, par un périphérique ou par un autre processus appelé **parent**.

```
UID          PID  PPID  C STIME TTY          TIME CMD
test         900   739  0 10:51 ?        00:00:00 /usr/lib/gvfs/gvfs-udisks2-vol
test         907   838  0 10:51 ?        00:00:00 /usr/lib/gvfs/gvfsd-trash -sp
test         913   739  0 10:51 ?        00:00:00 /usr/lib/gvfs/gvfsd-metadata
test         918   823  0 10:51 ?        00:00:00 /usr/lib/x86_64-linux-gnu/xfce
test         919   823  0 10:51 ?        00:00:00 /usr/lib/x86_64-linux-gnu/xfce
test         923     1  0 10:51 ?        00:00:02 xfce4-terminal
test         927   923  1 10:51 pts/0    00:00:00 bash
test        1036     1  0 11:18 ?        00:00:02 mousepad /home/test/Documents/
test        1058   923  0 11:22 pts/1    00:00:00 bash
test        1132     2  0 11:37 ?        00:00:00 [kworker/0:0-ata_sff]
test        1134     2  0 11:43 ?        00:00:00 [kworker/0:2-ata_sff]
test        1140   739  0 11:43 ?        00:00:00 /usr/lib/x86_64-linux-gnu/tumb
test        1149     2  0 11:43 ?        00:00:00 [kworker/u2:0-events_unbound]
test        1153   927  0 11:44 pts/0    00:00:00 vi
test        1154   927  0 11:44 pts/0    00:00:00 python3 prog.py
test        1155  1058  0 11:44 pts/1    00:00:00 ps -aef
```

   a) Donner le PID du parent du processus démarré par la commande `vi`.

   b) Donner le PID d'un processus enfant du processus démarré par la commande `xfce4-terminal`.

c) Citer le PID de deux processus qui ont le même parent.

d) Parmi tous les processus affichés, citer le PID des deux qui ont consommé le plus de temps de processeur.

3) On considère les trois processus $P_1$, $P_2$ et $P_3$, tous soumis à l'instant 0 dans l'ordre 1, 2, 3 :

| Nom du processus | Durée d'exécution en unité de temps | Ordre de soumission |
|---|---|---|
| $P_1$ | 3 | 1 |
| $P_2$ | 1 | 2 |
| $P_3$ | 4 | 3 |

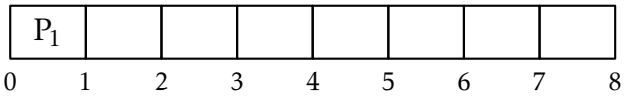
a) Dans cette question, on considère que les processus sont exécutés de manière concurrente selon la politique du tourniquet : le temps est découpé en tranches nommées *quatums de temps*.

Les processus prêts à être exécutés sont placés dans une file d'attente selon leur ordre de soumission.

Lorsqu'un processus est élu, il s'exécute au plus durant un quantum de temps. Si le processus n'a pas terminé son exécution à l'issue du quantum de temps, il réintègre la file des processus prêts (côté entrée). Un processus, désormais en tête de la file (côté sortie) des processus prêts, est alors à son tour élu pour une durée égale à un quantum de temps maximum.

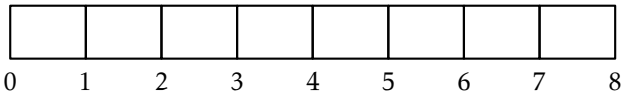
entrée $\longrightarrow$ $\overline{\text{P}_3 \quad \text{P}_2 \quad \text{P}_1}$ $\longrightarrow$ sortie

Reproduire le tableau ci-dessous sur la copie et indiquer dans chacune des cases le processus exécuté à chaque cycle. Le quantum correspond à une unité de temps.




b) Dans cette question, on considère que les processus sont exécutés en appliquant la politique du "plus court d'abord" : les processus sont exécutés complètement dans l'ordre croissant de leur temps d'exécution, le plus court étant exécuté en premier.

Reproduire le tableau ci-dessous sur la copie et indiquer dans chacune des cases le processus exécuté à chaque cycle.




4) On considère trois ressources $R_1$, $R_2$ et $R_3$ et trois processus $P_1$, $P_2$ et $P_3$ dont les files d'exécution des instructions élémentaires sont indiquées ci-dessous.

| Processus $P_1$ |
|---|
| Demande $R_1$ |
| Demande $R_2$ |
| Libère $R_1$ |
| Libère $R_2$ |

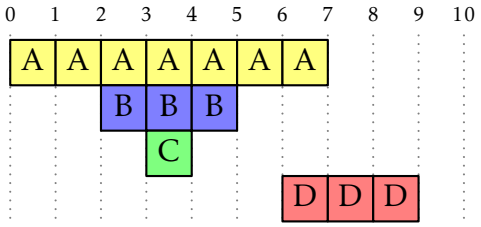
| Processus $P_2$ |
|---|
| Demande $R_2$ |
| Demande $R_3$ |
| Libère $R_2$ |
| Libère $R_3$ |

| Processus $P_3$ |
|---|
| Demande $R_3$ |
| Demande $R_1$ |
| Libère $R_3$ |
| Libère $R_1$ |

a) Rappeler les différents états d'un processus et expliquer pourquoi il y a ici risque d'interblocage, en proposant un ordre d'exécution des instructions élémentaires le provoquant.

b) Proposer une modification dans l'ordre des instructions des processus qui permet d'éviter tout problème d'interblocage.

**Exercice 3 :** Le schéma représente l'heure d'arrivée et la durée de 4 processus. L'unité de temps est le quantum.

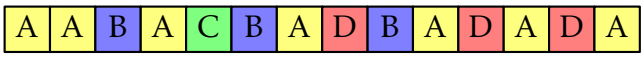



Cela veut dire que le processus B va arriver au temps 2 et durer 3 quantums.
On considère 3 stratégies d'ordonnancement :

- **La file (FIFO) :** le premier processus arrivé est le premier à être exécuté et ce n'est qu'à sa fin que le processus suivant peut commencer.
  On obtient alors :


- **Le plus rapide à finir en premier (*Shortest Remaining Time First*) :** on exécute en priorité le processus dont le temps d'exécution restant est le plus petit.
  On obtient alors :


- **Le tourniquet (*Round-Robin*) :** on attribue un quantum de temps identique à chaque tâche extraite de la file d'attente, pour l'y replacer dès que le quantum est échu.
  On obtient alors :




1) Expliquer le résultat de chaque stratégie d'ordonnancement de l'exemple.

2) Donner le résultat des 3 stratégies pour les processus suivants :

| nom | arrivée | durée |
|---|---|---|
| A | 0 | 8 |
| B | 3 | 4 |

**FIFO :**

**SRTF :**

**Round-Robin :**

3) Donner le résultat des 3 stratégies pour les processus suivants :

| nom | arrivée | durée |
|---|---|---|
| A | 0 | 5 |
| B | 1 | 5 |
| C | 2 | 2 |

**FIFO :**

**SRTF :**

**Round-Robin :**