

<p style="text-align: center;"><b>info507 : Systèmes d'exploitation</b> <b>TD 1 : appels système, processus et ordonnancement</b></p>
-------------------------------------------------------------------------------------------------------------------------------------------

Pierre Hyvernats, François Boussion, Gérald Cavallini

Pierre.Hyvernats@univ-smb.fr

Francois.Boussion@univ-smb.fr

Gerald.Cavallini@univ-smb.fr

### Exercice 1 : Appels système

*Question 1.* Est-il important pour un programmeur de savoir quelles fonctions provoquent des appels système ?

Comment savoir si une fonction provoque des appels système ?

*Question 2.* Parmi les fonctions suivantes, quelles sont celles qui provoquent des appels système ?

- `strcmp()`, pour comparer des chaînes de caractères,
- `malloc()`,
- `execl()`,
- `memcpy()`, pour copier une zone de mémoire vers une autre,
- `clock()`, pour avoir le temps d'utilisation (approximatif) du processeur pour le processus,
- `getpid()`,
- `kill()`,
- `fork()`,
- `time()`, pour lire l'heure,
- `wait()`,
- `random()`, pour obtenir un nombre aléatoire,
- `sched_yield()`,
- `printf()`.

### Exercice 2 : Les états possibles d'un processus

Lors de sa vie, un processus peut se trouver dans plusieurs états :

- nouveau
- prêt ou bloqué
- en exécution
- terminé

De plus, un processus peut se trouver soit en mémoire principale, soit en mémoire secondaire (en "swap").

*Question 1.* Quels sont les états qui peuvent être en mémoire principale ? Et ceux qui peuvent être en mémoire secondaire ?

À quoi correspond chaque état ? Donnez des exemples des situations où un processus se retrouve dans chaque état.

*Question 2.* Essayez de décrire toutes les transitions possibles entre états.

Commencez par traiter le cas où tous les processus sont en mémoire principale.

### Exercice 3 : Ordonnement simple (non préemptif)

On considère les huit processus suivants :

processus	temps d'arrivée	durée	priorité
$P_1$	0	3	0
$P_2$	$1 - \varepsilon$	24	1
$P_3$	$1 - \varepsilon$	8	2
$P_4$	$7 - \varepsilon$	5	2
$P_5$	$8 - \varepsilon$	4	1
$P_6$	$10 - \varepsilon$	2	4
$P_7$	$15 - \varepsilon$	7	4
$P_8$	$16 - \varepsilon$	2	2

Un processus qui arrive au temps  $2 - \varepsilon$  est un processus qui est présent juste avant le temps 2, mais qui n'était pas présent au temps 1.

*Question 1.* Donnez l'ordre d'exécution des processus pour la politique d'ordonnement "FIFO".

*Question 2.* Idem pour la politique d'ordonnement "FIFO avec priorités" (sans réquisition). Remarque : 0 est plus prioritaire que 1, etc.

*Question 3.* Idem pour la politique d'ordonnement "plus court temps d'exécution". Qu'en pensez-vous ?

*Question 4.* Discutez sur ce que serait un ordonnement "optimal" pour ces processus.

### Exercice 4 : Ordonnement préemptif

On reprend les processus de l'exercice précédent...

*Question 1.* Donner l'ordonnement des tâches en suivant la politique "tourniquet" avec un quantum de temps de deux unités.

*Question 2.* Idem en utilisant un tourniquet avec priorités. La priorité du processus actif diminue (jusqu'à 5) lorsqu'il est remplacé.

Quel problème est-ce que cette politique peut poser ?