

Numérique et Sciences Informatiques
Chapitre IX - Gestion des processus et des ressources
par un système d'exploitation
Travaux Dirigés 18

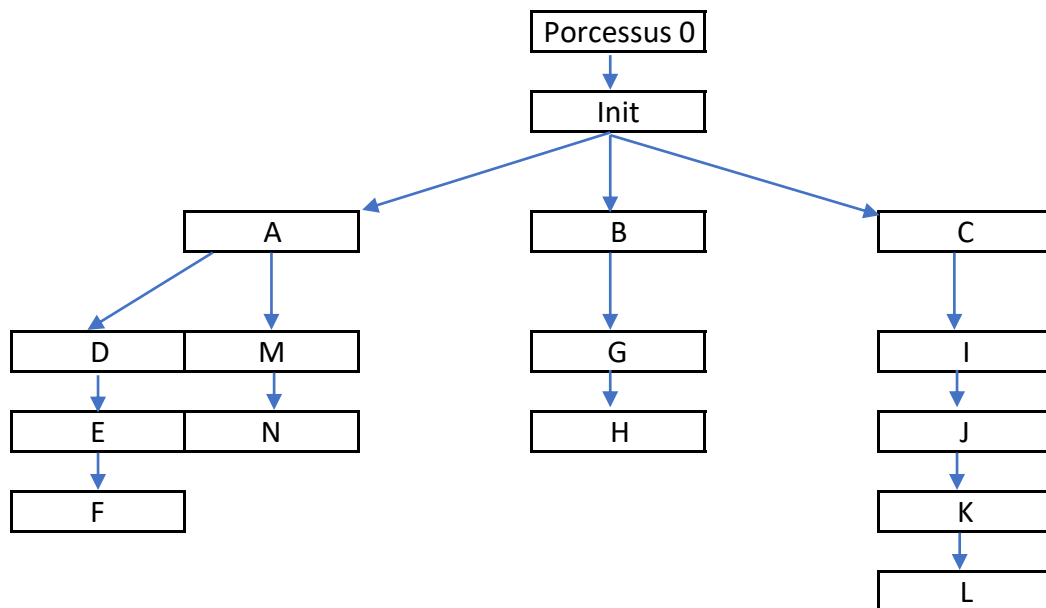
I. Où trouve-t-on les processus

1. Exécuter le logiciel *Mozilla FireFox* ou *Google Chrome*.
2. Aller à l'adresse suivante : <https://www.offidocs.com/index.php/desktop-online-utilitaires-apps-fr-fr/xlinux-linux-en-ligne-fr-fr>.
3. Cliquer sur le bouton **Enter** . Une console virtuelle linux apparaît.
4. Exécuter la commande :

```
1 ps
```

La liste des processus en cours d'exécution apparaît.

5.
 - (a) Dans quel état peuvent être ces processus ?
 - (b) Quel est le dernier processus exécuté ?
 - (c) Quel est le PID de ce dernier processus ?
 - (d) Exécuter à nouveau le processus **ps** . Que remarquez-vous quant au dernier processus exécuté ?
 - (e) Regarder la liste des PID. Que remarquez-vous ?
 - (f) Combien de processus se sont-ils lancé jusqu'à présent ?
 - (g) Combien de processus sont-ils en cours d'exécution ?
 - (h) Que dire des autres processus ?
6. Voici un arbre représentant des processus créés et les liens entre ces derniers.



- (a) Rappeler le PID du processus 0.
- (b) Rappeler le PID et le PPID du processus **Init** .
- (c) En parcourant cet arbre en profondeur préfixe, on retrouve l'ordre de création des processus. Quel est le PID et le PPID du processus *M* ?

II. L'interblocage

Certains processus nécessitent un ensemble de ressources pour s'exécuter.

L'utilisation d'une ressource passe par trois étapes :

- la demande de la ressource : si la demande n'est pas satisfaite, il faut attendre (le processus reste à l'état *Bloqué*).
- l'utilisation de la ressource : le processus peut utiliser la ressource (le processus doit être à l'état *Elu*).
- la libération de la ressource : le processus libère la ressource demandée (le processus doit être à l'état *Elu*).

Imaginons deux processus $P1$ et $P2$ nécessitant deux ressources $R1$ et $R2$ pour leur exécution.

- Premier cas : $P1$ et $P2$ demandent simultanément les ressources dans le même ordre, par exemple, ils demandent $R1$ en premier et $R2$ en deuxième.
Etablir la liste des états de $P1$ et $P2$. Justifier à chaque étape.
- Deuxième cas : $P1$ et $P2$ demandent simultanément les ressources dans un ordre différent, par exemple, $P1$ demande $R1$ en premier et $R2$ en deuxième alors que $P2$ demande $R2$ en premier et $R1$ en deuxième.
Etablir la liste des états de $P1$ et $P2$. Justifier à chaque étape.

III. Ordonnancement Round-robin

III.1. Un exemple

On donne dans le tableau ci-dessous quatre processus qui doivent être exécutés par un processeur. Chaque processus a un instant d'arrivée et une durée, donnés en quantum.

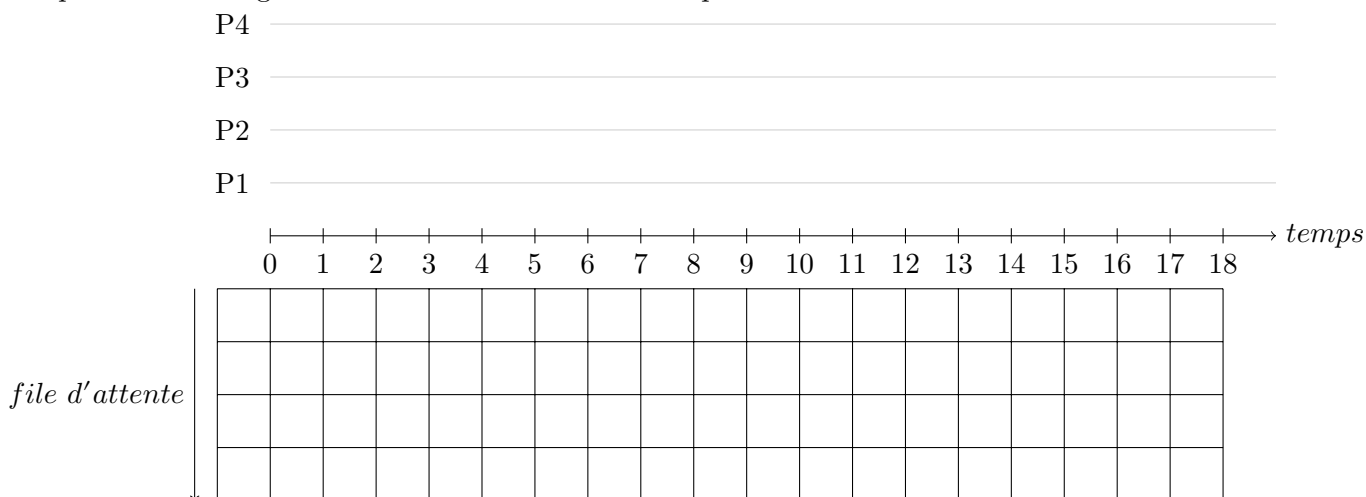
Processus	P1	P2	P3	P4
Instant d'arrivée	0	2	3	7
Durée	8	6	2	2

Les processus sont placés dans une file d'attente en fonction de leur instant d'arrivée.

Les processus sont élus pendant 1 quantum de temps.

Un processus qui devient bloqué à un instant t arrivera dans la file d'attente après un processus nouveau qui arrive au même instant t .

Compléter le chronogramme et les files d'attente à chaque instant.



III.2. Implémentation

On considère les classes **Processus** et **Ordonnaceur**.

La première crée une instance de type **Processus** dont on peut connaître le nom, la durée d'exécution restante et l'état.

La seconde simule un ordonnanceur.

1. La classe `Processus`

- (a) Compléter les sélecteurs `donne_nom`, `donne_duree` et `donne_etat` de la classe `Processus`.
- (b) Compléter les méthodes `bloquage`, `election` et `terminaison` de la classe `Processus`. Elles changent l'état du processus.
- (c) Compléter la méthode `election` de la classe `Processus`. Elle change l'état du processus et réduit de 1 la durée d'exécution restante, en ne descendant jamais en dessous de 0.

2. La classe `Ordonnanceur`

- (a) Créer la méthode `ajoute_proc` qui prend en paramètres un nom de processus `nom`, un instant d'arrivée `arrivee` et une durée d'exécution `duree`. Cette méthode ajoute à l'attribut `__processus_restants` le couple (`arrivee`, `processus`) où `processus` est une instance d'un processus de nom et durée donnés.
- (b) Créer la méthode `ajoute_proc_file` qui enfile dans l'attribut `__file` un processus si l'instant correspond à l'arrivée d'un processus parmi les processus restants.
- (c) Créer la méthode `elir` qui prend en paramètre un processus. Cette méthode simule l'élection et donc l'exécution du processus durant un quantum de temps. De plus, si le processus a fini son exécution complète, il devra se terminer.
- (d) Créer la méthode `termine` qui prend en paramètre un processus. Cette méthode simule la terminaison du processus.
- (e) Créer la méthode `preempte` qui prend en paramètre un processus. Cette méthode simule la préemption du processus.
- (f) Créer la méthode `lancement` qui exécute l'ordonnanceur et donc fait défiler le temps. Elle affichera à chaque instant le nom du processus « élu » et la durée restante d'exécution, s'il y en a un jusqu'à ce que tous les processus aient fini leur exécution.