

Les processus

Table des matières

1	Les processus	2
1.1	Introduction	2
1.2	Définition	2
1.2.1	Création d'un processus	2
1.3	Visualisation des processus	2
1.3.1	Sur Windows	2
1.3.2	Sur Linux	3
1.4	Terminer un processus	3
1.4.1	Sur Windows	3
1.5	Sur Linux	3
1.6	PID et PPID	4
2	Ordonnancement	4
3	Les interblocages	7
3.1	Quels interblocages en informatique ? Exemple de deux processus qui ont besoin de la même donnée de manière exclusive (pour la modifier par exemple)	7

CE QU'IL FAUT SAVOIR FAIRE À L'ISSUE DU CHAPITRE :

-

1 Les processus

1.1 Introduction

Afin de bien fonctionner, de nombreuses tâches ou application doivent être exécuté en même temps , par le système d'exploitation et les utilisateurs.

Pour permettre cela, et gérer les problèmes qui en découlent, le système d'exploitation va générer des processus, et les gère ensuite.

1.2 Définition

Définition 9.1

| Un **processus** est un programme en cours d'exécution sur un ordinateur.

⚠ La notion de programmes et de processus est différente. Le même programme exécuté plusieurs fois génère plusieurs processus.

1.2.1 Création d'un processus

La création d'un processus peut intervenir

- au démarrage du système
- par un appel d'un autre processus
- par une action d'un utilisateur (lancement d'application par exemple)

1.3 Visualisation des processus

1.3.1 Sur Windows

- Avec l'interface windows, il suffit d'appuyer simultanément sur les touches ctrl, alt et suppr.
- En ligne de commande : Cliquer sur "Windows"+"R", puis entre cmd puis Entrée.
 - *tasklist* liste les processus sur l' ordinateur
 - *taskkill* permet d'arrêter un processus.

1.3.2 Sur Linux

Il est possible de les visualiser grâce à la commande `ps -a -u -x`.

La colonne USER indique le nom de l'utilisateur qui a lancé le processus.

La colonne PID donne l'identifiant numérique du processus.

Les colonnes % CPU et % MEM donnent respectivement le taux d'occupation du processeur et de la mémoire.

Un caractère "?" indique que le processus n'a pas été lancé depuis un terminale.

La colonne STAT indique l'état du processus :

- R pour running (prêt ou en exécution)
- S pour sleeping (en attente)
- ☛ La commande `top` est très pratique, avec un affichage qui s'actualise toutes les secondes.

1.4 Terminer un processus

1.4.1 Sur Windows

Il suffit de cliquer sur fin de tâches, ou bien d'utiliser *taskkill* en ligne de commande.






1.5 Sur Linux

Pour tuer un processus, on lui envoie un signal de terminaison. On en utilise principalement deux :

- SIGTERM (15) : demande la terminaison d'un processus. Cela permet au processus de se terminer proprement en libérant les ressources allouées.
- SIGKILL (9) : demande la terminaison immédiate et inconditionnelle d'un processus. C'est une terminaison violente à n'appliquer que sur les processus récalcitrants qui ne répondent pas au signal SIGTERM.

A vous de jouer ! 9.1

Lancer une application et visualiser sur votre ordinateur les différents processus. Retrouver le processus correspondant à l'application que vous avez lancé. Puis supprimer-le en utilisant les lignes de commandes.

Processus Performance Historique des applications Démarrage Utilisateurs Détails Services						
^		5%	87%	0%	0%	2%
Nom	Statut	Processe...	Mémoire	Disque	Réseau	Processe...
Applications (12)						
>  Adobe Acrobat Reader DC (32 ...		0,1%	5,1 Mo	0 Mo/s	0 Mbits/s	0%
>  Explorateur Windows		0%	26,0 Mo	0 Mo/s	0 Mbits/s	0%
>  Gestionnaire des tâches		0,4%	28,4 Mo	0 Mo/s	0 Mbits/s	0%
>  GNU Image Manipulation Progr...		0%	41,7 Mo	0 Mo/s	0 Mbits/s	0%
>  Google Chrome (24)		0,1%	545,9 Mo	0,1 Mo/s	0 Mbits/s	0%

```
gimp-2.10.exe      5660 Console      1  110 844 Ko
```

```
C:\Users\DUTHOIT>taskkill /IM gimp-2.10.exe
Opération réussie : un signal de fin a été envoyé au processus "gimp-2.10.exe" de PID 5660.
```

Remarque

Vous remarquerez qu'une grande partie de la mémoire, dans mon exemple précédent, est utilisée par le navigateur Google Chrome et ses 24 onglets ouverts !

1.6 PID et PPID

Un processus est caractérisé par un identifiant unique : son PID (Process Identifier). Lorsqu'un processus engendre un fils, l'OS génère un nouveau numéro de processus pour le fils. Le fils connaît aussi le numéro de son père : le PPID (Parent Process Identifier).

⚠ Comme une ressource (le processeur ou un périphérique) ne peut pas être partagée, c'est son temps d'utilisation qui va l'être : le temps d'utilisation d'une ressource est partagé en intervalles très courts, pendant lesquels l'ordonnanceur l'alloue à un seul utilisateur.

2 Ordonnancement

Dans un système multitâche, plusieurs processus sont actifs simultanément, mais un processeur (simple coeur) ne peut exécuter qu'une instruction à la fois !

🗨 Nous allons donc avoir l'impression que le processeur gère les processus de façon simultanée, mais il en est rien.

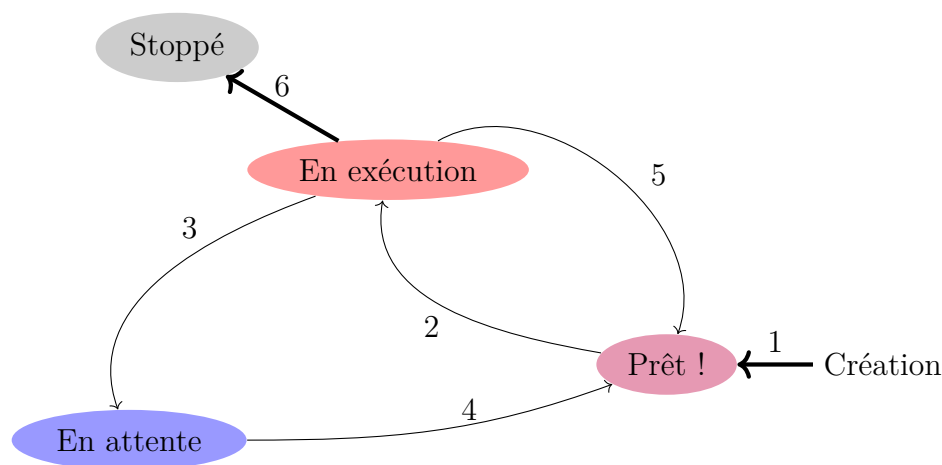
Il va donc falloir partager le temps de processeur disponible entre tous les processus : c'est le travail de **l'ordonnanceur**. Ce dernier a pour tâche de sélectionner le processus suivant à exécuter parmi ceux qui sont prêts.

Définition 9.2

L'ordonnanceur gère les processus, ayant pour objectif de partager le temps d'utilisation du processeur.

Afin de gérer cet ordonnancement, les processus se voient affectés d'états différents :

- prêt (ready) : le processus attend son tour pour prendre la main
- (running) : le processus a accès au processeur pour exécuter ses instructions
- (sleeping) : le processus attend qu'un événement se produise (saisie clavier, réception d'une donnée par le réseau ou le disque dur ...) en exécution .
- (stopped) : le processus a fini son travail ou a reçu un signal de terminaison (SIGTERM, SIGKILL, ...). Il libère les ressources qu'il occupe.



- 1 Le processus est créé ; il est "prêt"
- 2 **Eléction** L'ordonnanceur choisit ce processus. Il entre en exécution.
- 3 Le processus se met en attente d'un événement.
- 4 L'événement attendu se produit. Le processus est donc "prêt"
- 5 L'ordonnanceur décide de suspendre le processus afin de donner la main à un autre processus.
- 6 Le processus est terminé.

☞ Il est vraiment important de bien comprendre que c'est le système d'exploitation qui attribue aux processus les différents états "En exécution, En attente, prêts".
On dit que le système d'exploitation gère l'ordonnancement des processus (tel processus sera prioritaire sur tel autre...)

Afin d'élire le processus qui va repasser en mode exécution, l'ordonnanceur applique un algorithme prédéfini lors de la conception de l'OS.

Le choix de cet algorithme va impacter directement la réactivité du système et les usages qui pourront en être fait. C'est un élément critique du système d'exploitation.

Parmi les algorithmes les plus répandus :

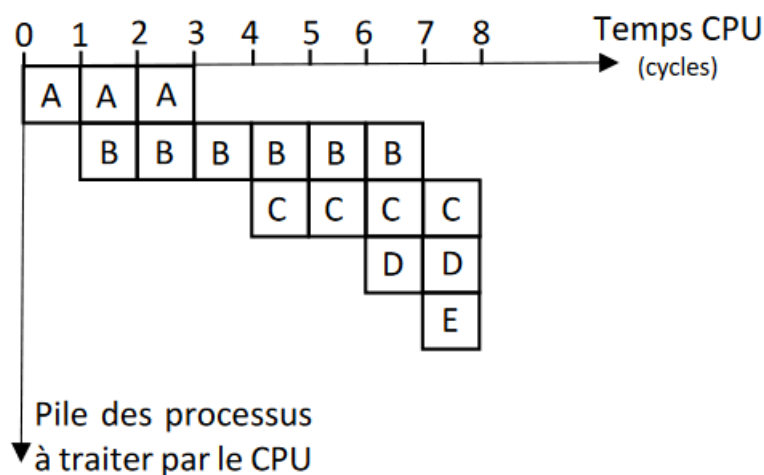
- Le **tourniquet** : La ressource est affectée à chaque processus à tour de rôle. Cette méthode à plusieurs avantages : simplicité, rapidité de gestion, robustesse.
- Mise en place d'un système de priorité : l'ordre de l'affectation de la ressource sera alors fonction de la priorité des tâches.
⚠ Attention au réglage du niveau de priorité, qui doit être "équitable" et "objectif".
- Gestion du premier entrée, premier sortie (FIFO). Comme par exemple la file d'impression des documents sur une imprimante.
- Le plus court d'abord. Assez efficace, mais la difficulté réside dans le fait d'estimer la durée du processus en amont, avant qu'il ne commence.

Exercice 9.1

Considérons cinq processus notés A, B, C, D et E, dont les temps d'exécution et leurs arrivages respectifs sont donnés dans le tableau ci-contre.

Processus	Temps d'exécution	Temps d'arrivée
A	3	0
B	6	1
C	4	4
D	2	6
E	1	7

On peut également représenter le tableau précédent de la manière suivante afin de faciliter la compréhension de l'ordonnancement des processus :



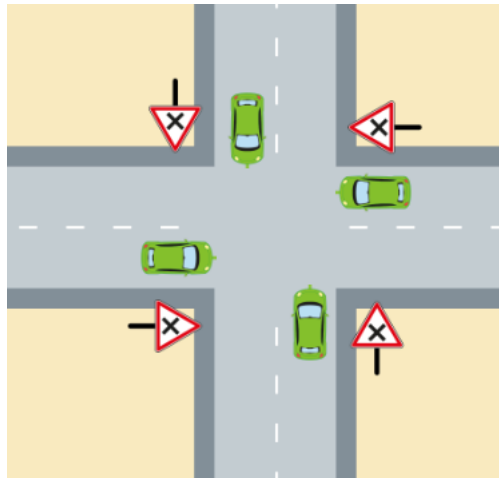
Représenter pour chaque algorithme le schéma d'exécution. Calculer le temps de séjours, le temps moyen de séjour, le temps d'attente (soustraction entre le temps de séjour et le temps d'exécution) et enfin le temps d'attente moyen

1. Algorithme "Premier arrivé, premier servi"
2. Algorithme "Le plus court avant".

**

3 Les interblocages

Les interblocages sont des situations de la vie quotidienne !



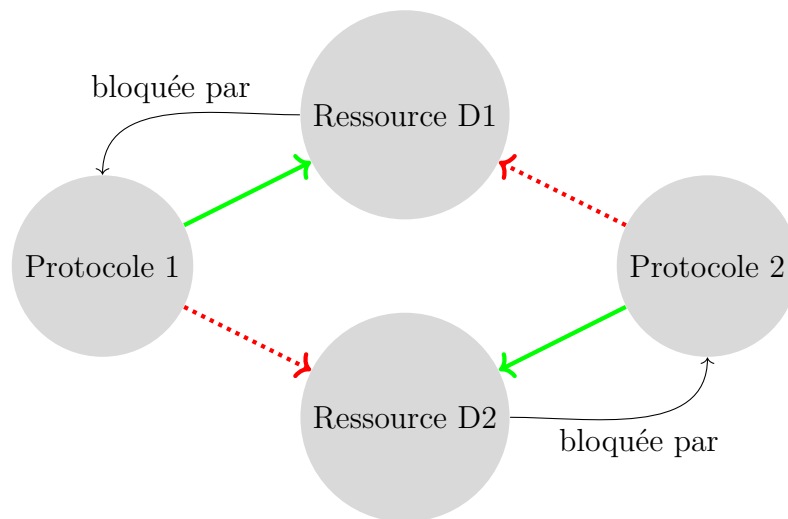
Qui doit passer ?

3.1 Quels interblocages en informatique ? Exemple de deux processus qui ont besoin de la même donnée de manière exclusive (pour la modifier par exemple)

Su le schéma ci-dessous, P1 et P2 ont tous les deux besoin des ressource D1 et D2
Voici un scénario possible :

- Le processus P1 commence son exécution (état élu), il demande la ressource D1. Il obtient satisfaction puisque D1 est libre
- P1 passe ensuite l'état "prêt".
- Pendant ce temps, le système a passé P2 en exécution : P2 commence son exécution et demande la ressource D2. Il obtient immédiatement D2 puisque cette ressource était libre.
- P2 poursuit son exécution
- P2 demande la ressource D1, il se retrouve dans un état bloqué puisque la ressource D1 a été attribuée à P1
- P1 est dans l'état prêt, il n'a pas eu l'occasion de libérer la ressource R1 puisqu'il n'a pas eu l'occasion d'utiliser R1
- P1 passe donc à l'état d'exécution . Afin de libérer D1, P1 a besoin de D2. Problème, car D2 n'est pas disponible ! Donc P1 ne peut pas libérer D1 et passe à bloqué.

Résumons la situation à cet instant : P1 possède la ressource D1 et se trouve dans l'état bloqué, P2 possède la ressource D2 et se trouve dans l'état bloqué (attente de D1)
Cette situation est qualifiée d'interblocage (deadlock en anglais).



Face à cette problématique, deux situations sont envisageables :

- Essayer d'éviter les interblocages.
- Détecter quand un interblocage est apparu, et le solutionner

☞ La plupart des systèmes d'exploitation ont choisi de ne pas essayer de les éviter, mais plutôt de détecter et de les solutionner.