

Programmation orientée objet

TD 1 - Identifications et ordonnancement des processus

Exercice 1 : Identification

Nous avons vu qu'un processus pouvait être identifié par son PID. De plus, grâce à son PPID, on peut également identifier son processus "parent".

Sur Windows dans la console on peut accéder à la liste des processus avec la commande "tasklist". Sur Linux c'est avec la commande "ps" que l'on peut observer la liste des processus. La commande "ps -ef" permet d'obtenir en plus, les PPID des processus.

Exemple de résultat suite à la commande "ps -ef" :

| UID | PID | PPID | C | STIME | TTY | TIME | CMD |
|-----|-------|-------|----|-------|-------|----------|---|
| ... | | | | | | | |
| pi | 6211 | 831 | 8 | 09:07 | ? | 00:01:16 | /usr/lib/chromium-browser/chromium-browser-v7 --disable-quit --enable-tcp-fast-open -- |
| pi | 6252 | 6211 | 0 | 09:07 | ? | 00:00:00 | /usr/lib/chromium-browser/chromium-browser-v7 --type=zygote --ppapi-flash-path=/usr/li |
| pi | 6254 | 6252 | 0 | 09:07 | ? | 00:00:00 | /usr/lib/chromium-browser/chromium-browser-v7 --type=zygote --ppapi-flash-path=/usr/li |
| pi | 6294 | 6211 | 4 | 09:07 | ? | 00:00:40 | /usr/lib/chromium-browser/chromium-browser-v7 --type=gpu-process --field-trial-handle=1 |
| pi | 6300 | 6211 | 1 | 09:07 | ? | 00:00:16 | /usr/lib/chromium-browser/chromium-browser-v7 --type=utility --field-trial-handle=10758 |
| pi | 6467 | 6254 | 1 | 09:07 | ? | 00:00:11 | /usr/lib/chromium-browser/chromium-browser-v7 --type=renderer --field-trial-handle=1075 |
| pi | 11267 | 6254 | 2 | 09:12 | ? | 00:00:15 | /usr/lib/chromium-browser/chromium-browser-v7 --type=renderer --field-trial-handle=1075 |
| pi | 12035 | 836 | 0 | 09:13 | ? | 00:00:00 | /usr/lib/libreoffice/program/oosplash --writer file:///home/pi/Desktop/mon_fichier.odt |
| pi | 12073 | 12035 | 2 | 09:13 | ? | 00:00:15 | /usr/lib/libreoffice/program/soffice.bin --writer file:///home/pi/Desktop/mon_fichier.c |
| pi | 12253 | 831 | 1 | 09:13 | ? | 00:00:07 | /usr/bin/python3 /usr/bin/sense_emu_gui |
| pi | 20010 | 6211 | 1 | 09:21 | ? | 00:00:00 | /usr/lib/chromium-browser/chromium-browser-v7 --type=utility --field-trial-handle=10758 |
| pi | 20029 | 6254 | 56 | 09:21 | ? | 00:00:28 | /usr/lib/chromium-browser/chromium-browser-v7 --type=renderer --field-trial-handle=1075 |
| pi | 20339 | 6254 | 4 | 09:21 | ? | 00:00:01 | /usr/lib/chromium-browser/chromium-browser-v7 --type=renderer --field-trial-handle=1075 |
| pi | 20343 | 6254 | 2 | 09:21 | ? | 00:00:00 | /usr/lib/chromium-browser/chromium-browser-v7 --type=renderer --field-trial-handle=1075 |
| pi | 20464 | 6211 | 17 | 09:22 | ? | 00:00:00 | /proc/self/exe --type=utility --field-trial-handle=1075863133478894917,6306120996223181 |
| pi | 20488 | 6254 | 14 | 09:22 | ? | 00:00:00 | /usr/lib/chromium-browser/chromium-browser-v7 --type=renderer --field-trial-handle=1075 |
| pi | 20519 | 676 | 0 | 09:22 | pts/0 | 00:00:00 | ps -ef |

En observant les PPID, on peut trouver le PID à l'origine de tous les processus concernant le navigateur Web (chromium-browser). Il suffit de "remonter" la liste des PPID pour trouver celui qui est à l'origine :

- On observe que la plupart des processus concernant le navigateur Web possèdent le même PPID : 6254
- Si on regarde le processus de PID 6254, celui-ci est toujours en rapport avec le navigateur Web, il faut donc continuer de "remonter". Le PPID du processus ayant pour PID 6254 est 6252.
- Idem pour ce processus, il est toujours en rapport avec le navigateur. On remonte donc grâce à son PPID : 6211
- Idem pour ce processus, il est toujours en rapport avec le navigateur. On remonte donc grâce à son PPID : 831. Nous n'avons pas plus d'informations sur ce processus.

Avec les informations à disposition, on peut annoncer que le processus à l'origine des processus qui concernent le navigateur Web est le processus de PID 831.

1. En utilisant le même exemple, déterminer le processus à l'origine des processus concernant LibreOffice
2. Selon vous, si je ferme le navigateur puis le relance, les processus le concernant garderont-ils forcément les mêmes PID qu'avant la fermeture ou non ? Pourquoi ?
3. Si je ferme le navigateur et que je regarde la liste des processus (avec tasklist ou ps), verrais-je encore les PID des processus concernant le navigateur ? Pourquoi ?

Il est possible de forcer la suppression d'un processus grâce à la commande :

- taskkill /PID ??? (sur Windows)
- kill ??? (sur Linux)

Les “???” correspondent aux PID du processus que l'on souhaite supprimer.

On peut également avoir accès aux différents processus depuis le gestionnaire des tâches de Windows. (Ctrl + Shift + Echap ou chercher “gestionnaire des tâches” depuis le menu Démarrer).

Gestionnaire des tâches

Fichier Options Affichage

Processus Performance Historique des applications Démarrage Utilisateurs Détails Services

| Nom | Statut | 3% Processeur | 64% Mémoire | 1% Disque | 0% Réseau | Pr |
|--|--------|------------------|----------------|--------------|--------------|----|
| Applications (8) | | | | | | |
| > Discord (32 bits) (6) | | 0,3% | 180,9 Mo | 0 Mo/s | 0 Mbits/s | |
| > Explorateur Windows | | 0% | 35,6 Mo | 0 Mo/s | 0 Mbits/s | |
| > Gestionnaire des tâches | | 0,6% | 26,6 Mo | 0 Mo/s | 0 Mbits/s | |
| > Google Chrome (11) | | 0,5% | 921,2 Mo | 0,1 Mo/s | 0 Mbits/s | |
| > Interpréteur de commandes Wi... | | 0% | 7,4 Mo | 0 Mo/s | 0 Mbits/s | |
| > Notepad++ : a free (GNU) sourc... | | 0% | 1,8 Mo | 0 Mo/s | 0 Mbits/s | |
| > PyScripter Python IDE (4) | | 0,6% | 14,6 Mo | 0 Mo/s | 0 Mbits/s | |
| > WinGup for Notepad++ | | 0% | 0,5 Mo | 0 Mo/s | 0 Mbits/s | |
| Processus en arrière-plan (101) | | | | | | |
| AMD External Events Client Mo... | | 0,2% | 0,7 Mo | 0 Mo/s | 0 Mbits/s | |
| > AMD External Events Service M... | | 0% | 0,3 Mo | 0 Mo/s | 0 Mbits/s | |
| > Antimalware Service Executable | | 0% | 142,8 Mo | 0 Mo/s | 0 Mbits/s | |

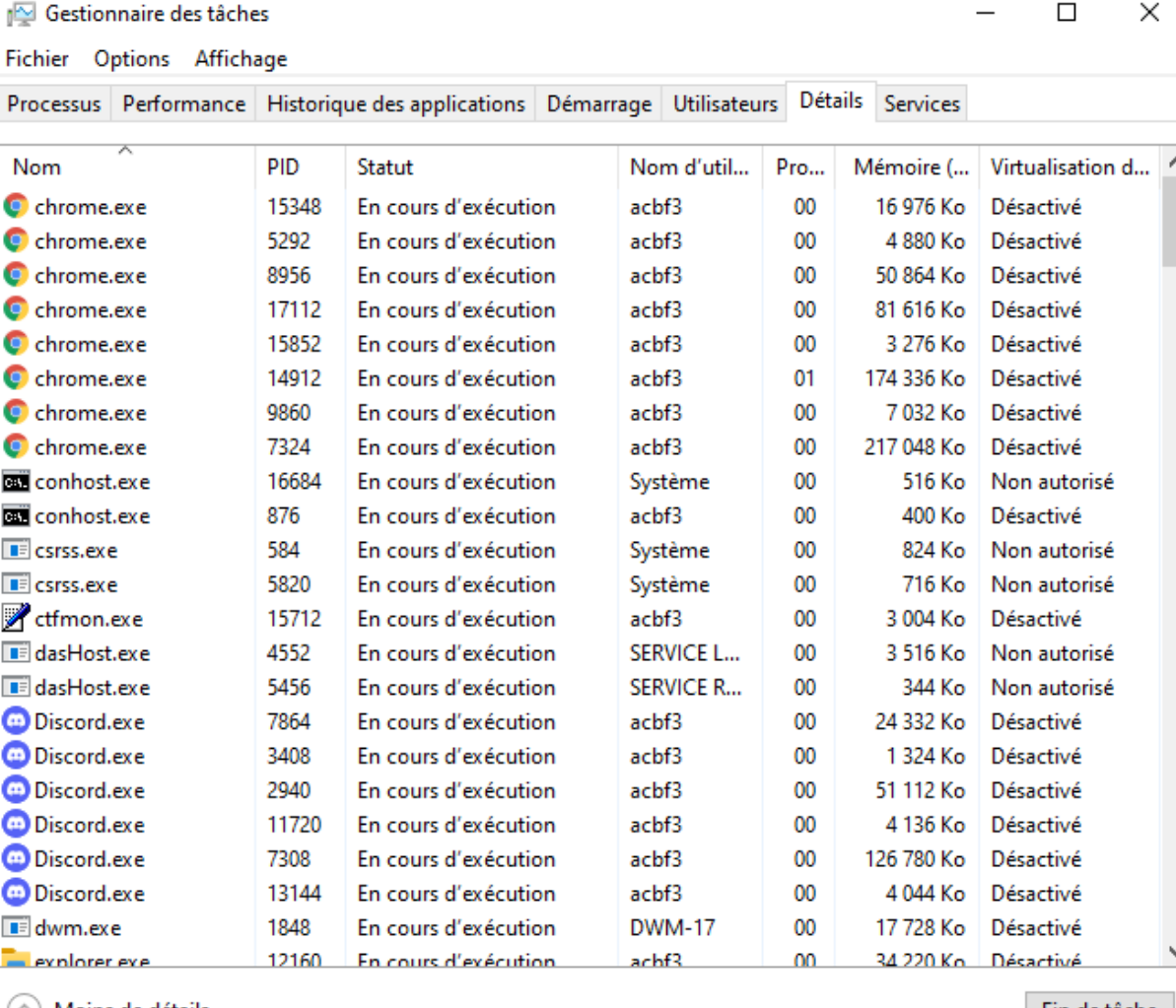
Moins de détails

Fin de tâche

Ce gestionnaire des tâches permet de voir de manière simplifiée les différents processus en cours d'exécution sur votre machine. De plus, vous pouvez forcer l'arrêt d'un processus en le sélectionnant et en cliquant sur le bouton "fin de tâche" (ce qui peut être pratique lorsqu'une application ne répond plus et ne veut plus se fermer conventionnellement).

D'autres informations sont disponibles comme les différents pourcentages concernant le processeur et la mémoire. Dans cet exemple, la mémoire de mon ordinateur est sollicitée à 64% et mon processeur à seulement 3%.

Il est également possible d'avoir un affichage plus détaillé et un accès au PID des processus dans l'onglet "Détails"



The screenshot shows the Windows Task Manager window with the 'Détails' (Details) tab selected. The window title is 'Gestionnaire des tâches'. The menu bar includes 'Fichier', 'Options', and 'Affichage'. The tabs at the top are 'Processus', 'Performance', 'Historique des applications', 'Démarrage', 'Utilisateurs', 'Détails' (active), and 'Services'.

| Nom | PID | Statut | Nom d'util... | Pro... | Mémoire (...) | Virtualisation d... |
|--------------|-------|----------------------|---------------|--------|---------------|---------------------|
| chrome.exe | 15348 | En cours d'exécution | acbf3 | 00 | 16 976 Ko | Désactivé |
| chrome.exe | 5292 | En cours d'exécution | acbf3 | 00 | 4 880 Ko | Désactivé |
| chrome.exe | 8956 | En cours d'exécution | acbf3 | 00 | 50 864 Ko | Désactivé |
| chrome.exe | 17112 | En cours d'exécution | acbf3 | 00 | 81 616 Ko | Désactivé |
| chrome.exe | 15852 | En cours d'exécution | acbf3 | 00 | 3 276 Ko | Désactivé |
| chrome.exe | 14912 | En cours d'exécution | acbf3 | 01 | 174 336 Ko | Désactivé |
| chrome.exe | 9860 | En cours d'exécution | acbf3 | 00 | 7 032 Ko | Désactivé |
| chrome.exe | 7324 | En cours d'exécution | acbf3 | 00 | 217 048 Ko | Désactivé |
| conhost.exe | 16684 | En cours d'exécution | Système | 00 | 516 Ko | Non autorisé |
| conhost.exe | 876 | En cours d'exécution | acbf3 | 00 | 400 Ko | Désactivé |
| csrss.exe | 584 | En cours d'exécution | Système | 00 | 824 Ko | Non autorisé |
| csrss.exe | 5820 | En cours d'exécution | Système | 00 | 716 Ko | Non autorisé |
| ctfmon.exe | 15712 | En cours d'exécution | acbf3 | 00 | 3 004 Ko | Désactivé |
| dasHost.exe | 4552 | En cours d'exécution | SERVICE L... | 00 | 3 516 Ko | Non autorisé |
| dasHost.exe | 5456 | En cours d'exécution | SERVICE R... | 00 | 344 Ko | Non autorisé |
| Discord.exe | 7864 | En cours d'exécution | acbf3 | 00 | 24 332 Ko | Désactivé |
| Discord.exe | 3408 | En cours d'exécution | acbf3 | 00 | 1 324 Ko | Désactivé |
| Discord.exe | 2940 | En cours d'exécution | acbf3 | 00 | 51 112 Ko | Désactivé |
| Discord.exe | 11720 | En cours d'exécution | acbf3 | 00 | 4 136 Ko | Désactivé |
| Discord.exe | 7308 | En cours d'exécution | acbf3 | 00 | 126 780 Ko | Désactivé |
| Discord.exe | 13144 | En cours d'exécution | acbf3 | 00 | 4 044 Ko | Désactivé |
| dwm.exe | 1848 | En cours d'exécution | DWM-17 | 00 | 17 728 Ko | Désactivé |
| explorer.exe | 12160 | En cours d'exécution | acbf3 | 00 | 34 220 Ko | Désactivé |

At the bottom left, there is a button 'Moins de détails' with an upward arrow icon. At the bottom right, there is a button 'Fin de tâche'.

Exercice 2 : Ordonnancement

Comme vu précédemment, de nombreux processus s'exécutent en parallèle sur une machine. L'ordonnanceur est là pour veiller au bon déroulement de l'ensemble des processus. Il existe plusieurs méthodes pour déterminer quel processus doit passer dans l'état élu et lesquels doivent attendre avant de pouvoir transmettre leurs instructions au processeur.

1. FIFO (First In First Out)

Les processus vont entrer dans une sorte de file d'attente. Le premier de la file sera le premier à exécuter toutes ses instructions. Une fois celui-ci terminé, le suivant sera élu et ainsi de suite.

Exemple :

| Processus 1 | Processus 2 | Processus 3 |
|--------------------|--------------------|--------------------|
| Instruction 1 (i1) | Instruction 1 (i1) | Instruction 1 (i1) |
| Instruction 2 (i2) | Instruction 2 (i2) | Instruction 2 (i2) |
| Instruction 3 (i3) | Instruction 3 (i3) | |
| | Instruction 4 (i4) | |
| | Instruction 5 (i5) | |

Prenons le cas où le processus 1 est le premier de la file suivi du processus 2 suivi du processus 3. L'ordre d'arrivée sera identique à l'ordre d'exécution ce qui donne :

| Ordre d'exécution | i1 | i2 | i3 | i1 | i2 | i3 | i4 | i5 | i1 | i2 |
|-------------------|----|----|----|----|----|----|----|----|----|----|
|-------------------|----|----|----|----|----|----|----|----|----|----|

2. Tourniquet

Le processeur va attribuer à chaque processus un quantum de temps durant lequel il va pouvoir exécuter ses instructions. Les processus seront encore dans une file mais cette fois un roulement va s'effectuer. Les processus n'ayant pas terminé toutes leurs instructions retourneront au début de la file en attendant leur tour.

Exemple :

Dans cet exemple, considérons que le temps d'exécution de chaque instruction est le même. Le processeur va attribuer un quantum de temps correspondant à l'exécution de 2 instructions.

| Processus 1 | Processus 2 | Processus 3 |
|--------------------|--------------------|--------------------|
| Instruction 1 (i1) | Instruction 1 (i1) | Instruction 1 (i1) |
| Instruction 2 (i2) | Instruction 2 (i2) | Instruction 2 (i2) |
| Instruction 3 (i3) | Instruction 3 (i3) | |
| | Instruction 4 (i4) | |
| | Instruction 5 (i5) | |

Prenons le cas où le processus 1 est le premier de la file suivi du processus 2 suivi du processus 3. L'ordre d'arrivée sera identique à l'ordre d'exécution ce qui donne :

| | | | | | | | | | | |
|-------------------|----|----|----|----|----|----|----|----|----|----|
| Ordre d'exécution | i1 | i2 | i1 | i2 | i1 | i2 | i3 | i3 | i4 | i5 |
|-------------------|----|----|----|----|----|----|----|----|----|----|

- Le processus P1 va pouvoir exécuter 2 instructions, il n'aura pas terminé son exécution complète il repassera à la fin de la file.
- Le processus P2 va commencer son exécution puis va aller au fond de la file.
- Le processus P3 commence et termine son exécution puisqu'il n'a que 2 instructions à exécuter. Il sort de la file.
- P1 finit son exécution. Il ne lui reste qu'une seule instruction, il l'exécute et sort de la file laissant sa place au processus suivant.
- P2 est le seul restant dans la file, il termine son exécution.

Prenons maintenant l'exemple où le processeur attribue un quantum de temps correspondant à l'exécution d'une instruction.

Prenons le cas où le processus 1 est le premier de la file suivi du processus 2 suivi du processus 3. L'ordre d'arrivée sera identique à l'ordre d'exécution ce qui donne :

| | | | | | | | | | | |
|-------------------|----|----|----|----|----|----|----|----|----|----|
| Ordre d'exécution | i1 | i1 | i1 | i2 | i2 | i2 | i3 | i3 | i4 | i5 |
|-------------------|----|----|----|----|----|----|----|----|----|----|

Après une instruction, le processus repasse au fond de la file et laisse sa place au suivant.

3. Plus court d'abord

Ici l'objectif est de favoriser les processus prenant le moins de temps d'exécution. Le problème de cette méthode est qu'il est parfois compliqué d'estimer le temps nécessaire à l'exécution d'un processus.

Exemple :

Dans cet exemple, considérons que le temps d'exécution de chaque instruction est le même.

| Processus 1 | Processus 2 | Processus 3 |
|--------------------|--------------------|--------------------|
| Instruction 1 (i1) | Instruction 1 (i1) | Instruction 1 (i1) |
| Instruction 2 (i2) | Instruction 2 (i2) | Instruction 2 (i2) |
| Instruction 3 (i3) | Instruction 3 (i3) | |
| | Instruction 4 (i4) | |
| | Instruction 5 (i5) | |

Avec la méthode d'ordonnancement du "plus court d'abord" on obtiendra :

| Ordre d'exécution | i1 | i2 | i1 | i2 | i3 | i1 | i2 | i3 | i4 | i5 |
|-------------------|----|----|----|----|----|----|----|----|----|----|
|-------------------|----|----|----|----|----|----|----|----|----|----|

4. Par priorités

Cette méthode permet d'attribuer un ordre de priorité à chaque processus. Le processus ayant le plus grand niveau s'exécutera et laissera sa place au processus suivant (celui ayant le plus grand ordre de priorité après lui).

Pour bien illustrer la chose, nous allons définir un ordre de priorité à chaque processus, un nombre d'instructions et un moment d'arrivée dans la liste des processus en attente.

Exemple :

Plus un processus a un petit ordre de priorité, plus il est prioritaire.

| | Processus P1 | Processus P2 | Processus P3 |
|-----------------------|--------------|--------------|--------------|
| Ordre de priorité | 2 | 3 | 1 |
| Nombre d'instructions | 3 | 5 | 2 |
| Moment d'arrivée | 1 | 0 | 3 |

Voici donc le résultat obtenu par rapport au temps. Quand un processus ayant une plus grande priorité arrive dans la file, le processus actif laisse immédiatement sa place si celui-ci a une plus faible priorité.

| Ordre d'exécution | P2 | P1 | P1 | P3 | P3 | P1 | P2 | P2 | P2 | P2 |
|-------------------|----|----|----|----|----|----|----|----|----|----|
| Temps t | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

- t = 0 : P2 entre dans la file et commence son exécution
- t = 1 : P1 entre dans la file. P1 a une plus forte priorité que P2. P2 lui cède sa place et P1 commence son exécution.
- t = 2 : P1 est toujours le processus ayant la plus forte priorité, il continue son exécution.
- t = 3 : P3 entre dans la file. P3 a une plus forte priorité que P1. P1 lui cède sa place et P3 commence son exécution.
- t = 4 : P3 est toujours le processus ayant la plus forte priorité, il continue son exécution. Il finit son exécution et sort donc de la file.
- t = 5 : P1 et P2 sont dans la file, P1 est celui ayant la plus forte priorité, il reprend donc son exécution et la termine. Il sort de la file.
- t = 6, 7, 8, 9 : P2 est le seul processus restant dans la file, il continue donc son exécution jusqu'à la fin.

Voici un exemple de processus avec leurs nombres d'instructions, leurs ordre de priorité, et leur moment d'arrivée :

| | Processus P1 | Processus P2 | Processus P3 | Processus P4 | Processus P5 |
|------------------------------|--------------|--------------|--------------|--------------|--------------|
| Ordre de priorité | 4 | 3 | 1 | 2 | 5 |
| Nombre d'instructions | 2 | 3 | 2 | 3 | 4 |
| Moment d'arrivée | 3 | 2 | 4 | 5 | 0 |

On suppose que le temps d'exécution de chaque instruction est le même.

1. Donnez l'ordre d'exécution avec la méthode FIFO.
2. Donnez l'ordre d'exécution avec la méthode du tourniquet. Le processeur va accorder un quantum de temps égal au temps d'exécution de 2 instructions
3. Donnez l'ordre d'exécution avec la méthode du plus court d'abord. En cas d'égalité sur le nombre d'instructions, le processus sera choisi aléatoirement.
4. Donnez l'ordre d'exécution avec la méthode par priorités.

Aucune justification n'est demandée.

Exercice 3 : Interblocage (deadlock)

L'interblocage est une situation où deux processus se bloquent mutuellement car ils sont dans l'attente d'une ressource réservée par l'autre.

Déterminez les scénarios dans lesquels il y a une situation d'interblocage et justifiez :

Les Processus commenceront par la lettre "P" et les ressources par la lettre "R"

Scénario 1 :

| |
|---------------|
| P1 réserve R2 |
| P2 réserve R3 |
| P3 réserve R1 |
| P1 attend R3 |
| P3 libère R1 |
| P3 attend R2 |

Scénario 2 :

| |
|---------------|
| P1 réserve R1 |
| P2 réserve R2 |
| P3 attend R1 |
| P2 libère R2 |
| P2 attend R1 |
| P1 libère R1 |

Scénario 3 :

| |
|---------------|
| P1 réserve R1 |
| P2 réserve R3 |
| P3 réserve R2 |
| P1 attend R2 |
| P2 libère R3 |
| P3 attend R1 |

Scénario 4 :

| |
|---------------|
| P1 réserve R1 |
| P2 réserve R2 |
| P3 attend R2 |
| P1 attend R2 |
| P2 libère R2 |
| P3 réserve R2 |