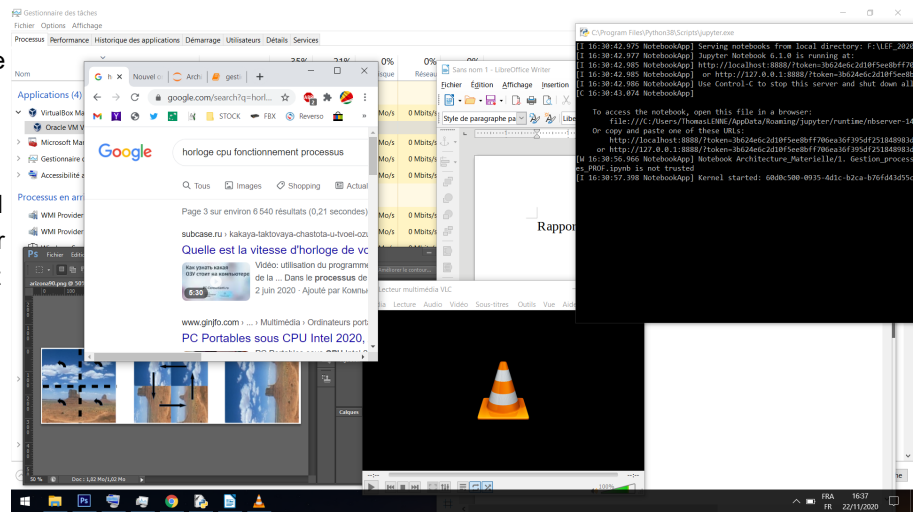


# Gestion des processus et des ressources par un système d'exploitation

- Considérons une utilisation banale d'un ordinateur: Un utilisateur rédige un rapport dans un traitement de texte, plusieurs onglets d'un navigateur sont ouverts pour effectuer des recherches, un logiciel d'édition d'image permet de modifier des illustrations pour le rapport, tout cela en écoutant de la musique via une application de l'ordinateur.
- Tous ces programmes donnent l'impression de s'exécuter en même temps.
- Pourtant, un programme est une suite d'instructions en langage machine qui sont exécutées une à une par le processeur. Comment exécuter en même temps les instructions du traitement de texte et du lecteur de musique ?
- Les systèmes d'exploitations modernes permettent les exécutions concurrentes de plusieurs programme grâce à l'ordonnanceur, ce que nous allons découvrir dans ce document après avoir rappelé les principes d'exécution d'un programme.
- Nous découvrirons également quelques commandes permettant d'afficher et arrêter des processus.



## 1. Exécution des programmes (Rappels)

- Un exécutable est un fichier qui contient une suite d'instructions en langage machine (appelées souvent des fichiers binaires de l'anglais *binary*). C'est donc une suite d'octets que le processeur est capable de décoder et exécuter.
- Ainsi lorsque l'on exécute un programme ( en double cliquant sur son icône ou à l'aide d'une console), le système d'exploitation effectue les actions suivantes :
  - Le fichier exécutable est copié dans la mémoire vive (RAM) à une certaine adresse.
  - Le système d'exploitation écrit la valeur de cette adresse dans un registre.
- Afin de synchroniser toutes les exécutions, le processeur dispose d'une horloge dont la fréquence tourne actuellement autour de 3 GHz. Cette fréquence indique le nombre de cycles que le processeur effectue par seconde, c'est à dire 3 milliards d'exécution par seconde !
- Il est important de comprendre qu'à chaque cycle, une seule instruction est exécutée. Ainsi, pour simplifier, un programme est exécuté instruction par instruction à chaque cycle.

## 2. L'ordonnanceur

- Cette description est néanmoins incomplète, car il faudrait donc attendre qu'un programme ait fini d'exécuter toutes ses instructions avant de pouvoir passer à un autre programme.

### Exemple :

Considérons ce petit programme python :

```
a=input('Entrez une valeur`')
print('Bravo')
```

Avec la description précédente, il serait impossible de faire autre chose pour l'utilisateur puisque le programme ne fait rien d'autre tant qu'il n'a pas saisi une valeur.

- Pour pallier à ce problème pouvoir exécuter plusieurs programmes "en même temps", les systèmes d'exploitation modernes utilisent des *interruptions* .

## Interruptions

- Une interruption est un signal envoyé au processeur lorsqu'un événement se produit. Les interruptions peuvent être générées par un matériel (le disque dur signale qu'il a fini d'écrire des octets, une carte réseau signale l'arrivée de données, etc...).
- Le processeur qui reçoit ce signal interrompt l'exécution en cours, mémorise l'état de cette exécution, puis choisit un autre programme à exécuter parmi ceux qui sont en attente.
- Ce sont ces interruptions qui permettent aux programmes de s'exécuter de façon concurrente ("en même temps").

## Vocabulaire

Avant d'aller plus loin, quelques termes qui seront utilisés par la suite :

- **Exécutable :**
  - Un fichier binaire contenant des instructions machines directement exécutables par un processeur.
- **Processus :**
  - Un programme en cours d'exécution. Chaque processus est identifié par un numéro de manière unique par le processeur.
- **Thread ou tâche :**
  - Exécution d'une suite d'instructions d'un processus :
    - Deux processus = Deux programmes (par exemple un navigateur et un traitement de texte)
    - Deux threads peuvent être issus d'un même processus et s'exécutent donc de manière concurrente (par exemple dans un navigateur : télécharger un fichier et afficher un texte dans un onglet)
- **Exécution concurrente :**
  - Deux processus ou deux threads s'exécutent de manière concurrentes lorsqu'ils essaient de "s'exécuter en même temps".
- **Exécution parallèle :**
  - Deux processus ou deux threads s'exécutent de manière parallèles lorsqu'ils s'exécutent réellement en même temps. Ceci n'est possible qu'avec la présence de plusieurs processeurs sur la machine (architecture multicoeurs).
- **Ressource:**
  - Tout objet informatique matériel ou logiciel nécessaire à l'exécution d'un programme. Exemples de ressources :
    - fichiers, mémoire, processus, imprimante, carte son ,...

## Ordonnancement

- Le système d'exploitation va donc gérer l'ordre d'exécution des processus, à l'aide d'un programme appelé *gestionnaire d'interruption*.
- Voici un exemple de déroulement :

1. Le programme traitement de texte est en cours d'exécution( saisie de texte, mise en page,...)
2. Une interruption d'horloge se déclenche.
3. Le gestionnaire d'interruption est appelé, prend la main, et sauvegarde l'état du traitement de texte à un endroit particulier de la mémoire.
4. Il choisit dans la liste des processus un autre processus, par exemple le navigateur web.
5. Il restaure l'état de ce processus dans lequel il était au moment de sa dernière interruption.
6. Le gestionnaire d'interruption rend la main au navigateur web, jusqu'à la prochaine interruption.

- Afin de pouvoir choisir parmi tous les processus à lequel exécuter lors de la prochaine interruption, le système d'exploitation conserve entre autre pour chaque processus :
  - Son numero unique , le PID (Process ID).
  - L'état dans lequel se trouve le processus (en attente/en cours d'exécution)
  - La valeur des registres lors de sa dernière interruption (contenu des variables,...)
  - ...
- Les processus peuvent ainsi être placés dans une file(au sens de celui déjà abordé en programmation). Le premier processus dans la file sera le prochain à reprendre son exécution, puis sera placé à la fin de la file à la prochaine interruption.Ceci évite qu'un processus monopolise les ressources de calcul du processeur.
- On peut visualiser les processus présents dans un système à l'aide de son gestionnaire de tâches. Ci-dessous, les gestionnaires de tâches de Windows 10(à gauche) et d'une distribution Linux Debian(à droite) en mode graphique ou console :

Windows Task Manager - Gestionnaire des tâches

Nom	Statut	Processeur	Mémoire	Disque	Réseau	Processus
<b>Applications (9)</b>						
Adobe Photoshop CS6 (2)		1,9%	188,1 Mo	0 Mo/s	0 Mo/s	0%
Explorateur Windows		0%	39,2 Mo	0 Mo/s	0 Mo/s	0%
Gestionnaire des tâches		0,6%	37,8 Mo	0 Mo/s	0 Mo/s	0%
Google Chrome (21)		0%	766,3 Mo	0 Mo/s	0 Mo/s	0%
Interpréteur de commandes W...		0%	21,4 Mo	0 Mo/s	0 Mo/s	0%
LibreOffice (32 bits)		0%	9,4 Mo	0 Mo/s	0 Mo/s	0%
MSI True Color		0%	1,2 Mo	0 Mo/s	0 Mo/s	0%
VirtualBox Manager		0%	7,9 Mo	0 Mo/s	0 Mo/s	0%
VirtualBox Manager		0,9%	85,5 Mo	0 Mo/s	0 Mo/s	0%
<b>Processus en arrière-plan (92)</b>						
Adobe Acrobat Update Service ...		0%	0,7 Mo	0 Mo/s	0 Mo/s	0%
Antimalware Service Executable		0,1%	74,4 Mo	0 Mo/s	0 Mo/s	0%
Application Frame Host		0%	3,6 Mo	0 Mo/s	0 Mo/s	0%
Application sous-système spou...		0%	0,1 Mo	0 Mo/s	0 Mo/s	0%
AppVSNHost		0%	0,1 Mo	0 Mo/s	0 Mo/s	0%
AppVSNHost		0%	0,1 Mo	0 Mo/s	0 Mo/s	0%
Bonjour Service		0%	0,3 Mo	0 Mo/s	0 Mo/s	0%
Chargeur CTF		0%	4,2 Mo	0 Mo/s	0 Mo/s	0%
COM Surrogate		0%	1,3 Mo	0 Mo/s	0 Mo/s	0%
COM Surrogate		0%	0,4 Mo	0 Mo/s	0 Mo/s	0%
Component Package Support Se...		0%	0,1 Mo	0 Mo/s	0 Mo/s	0%

Linux Task Manager - htop

Tâche	PID	RSS	Processeur
applet.py	25788	33,1 Mio	0%
at-spi2-registr...	25810	6,7 Mio	0%
at-spi-bus-lau...	25787	6,2 Mio	0%
bash	25946	5,1 Mio	0%
blueman-applet	25765	50,2 Mio	0%
dbus-daemon --config-file=/usr/share/defaults/at-spi2/accessibility.conf --nofork --print-address 3	25798	4,2 Mio	0%
dbus-daemon --session --address=systemd: --nofork --nopidfile --systemd-activation --syslog-only	25649	4,8 Mio	0%
dconf-service	25778	4,9 Mio	0%
Firefox	25985	243,4 Mio	0%
Gestionnaire de tâches	26452	26,6 Mio	3%
gvfs-fc-volume-monitor	25909	7,5 Mio	0%
gvfsd	25797	7,0 Mio	0%

Nom de l'image	PID	Nom de la sessio	Numéro de s	Utilisation	PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
System Idle Process	0	Services	0	8 Ko	792	tlenne	20	0	2791296	233012	113768	S	4,0	23,0	0:04.87	firefox+
System	4	Services	0	124 Ko	977	tlenne	20	0	838000	195044	100144	S	0,3	19,3	0:00.82	soffice+
Registry	124	Services	0	49 912 Ko	996	tlenne	20	0	349504	40960	32332	S	0,3	4,1	0:00.25	xfce4-t+
smss.exe	452	Services	0	1 212 Ko	1	root	20	0	103944	8108	6104	S	0,0	0,8	0:01.35	systemd
csrss.exe	672	Services	0	5 628 Ko	2	root	20	0	0	0	0	S	0,0	0,0	0:00.00	kthreadd
wininit.exe	780	Services	0	6 904 Ko	3	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	rcu_gp
services.exe	852	Services	0	11 376 Ko	4	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	rcu_par+
lsass.exe	896	Services	0	21 092 Ko	5	root	20	0	0	0	0	I	0,0	0,0	0:00.04	kworker+
svchost.exe	472	Services	0	3 448 Ko	6	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	kworker+
svchost.exe	700	Services	0	33 188 Ko	7	root	20	0	0	0	0	I	0,0	0,0	0:00.00	kworker+
fontdrvhost.exe	920	Services	0	13 232 Ko	8	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	mm_perc+
MUDFHost.exe	1052	Services	0	6 320 Ko	9	root	20	0	0	0	0	S	0,0	0,0	0:00.04	ksoftir+
svchost.exe	1156	Services	0	15 588 Ko	10	root	20	0	0	0	0	I	0,0	0,0	0:00.00	rcu_sch+
svchost.exe	1208	Services	0	8 860 Ko	11	root	20	0	0	0	0	I	0,0	0,0	0:00.00	rcu_bh
svchost.exe	1280	Services	0	11 600 Ko	12	root	rt	0	0	0	0	S	0,0	0,0	0:00.00	migrati+
					13	root	20	0	0	0	0	I	0,0	0,0	0:00.00	kworker+
					14	root	20	0	0	0	0	S	0,0	0,0	0:00.00	cpuhp/0

## Etats des processus

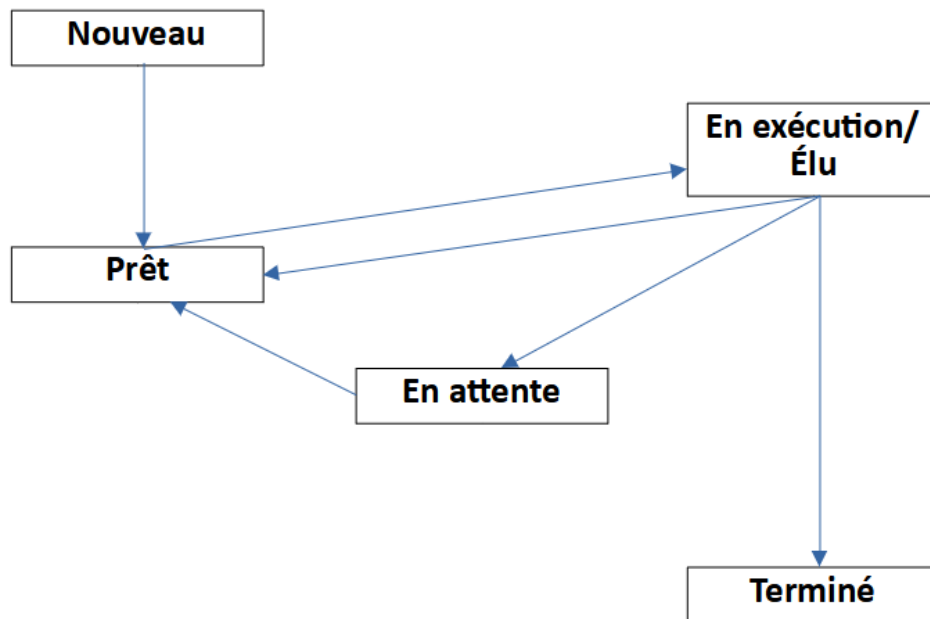
Les processus peuvent donc être dans différents états :

- **Nouveau** : Le processus est en cours de création, le système d'exploitation vient de copier l'exécutable en mémoire et initialise ses informations.
- **Prêt** : Le processus peut être le prochain à s'exécuter, il est dans la file des processus qui "attendent" leur tour.
- **En exécution (ou élu)** : Le processus est en train de s'exécuter.
- **En attente** : Le processus a été interrompu et est en attente d'une prochaine interruption.

- **Terminé:** Le processus est terminé et le système d'exploitation l'enlève des processus identifiés.

## Cycle de vie d'un processus

On peut résumer ces différents états dans le schéma ci-dessous :



### Remarque :

- Les états "nouveau" et "terminé" sont éphémères. Le plus souvent, l'état d'un processus varie entre "prêt", "en attente" et "en cours d'exécution".
- Quelque soit son état, un processus peut se terminer de façon anormale (erreur matérielle, intervention de l'utilisateur...)
- Les interruptions peuvent être déclenchées par le processeur à intervalles réguliers, mais aussi par un matériel (une carte son qui demande un accès au micro,...)

**Exercice 1 :** Il se trouve dans le dossier [Exercices \(Exercices/Processus\\_Exercices.pdf\)](#).

### 3. Quelques commandes

	Windows	Linux
Afficher les processus	tasklist	ps ou top
Tuer des processus	taskkill	kill
Interface graphique	« Gestionnaire de tâches »	

Il est possible à l'utilisateur d'identifier les différents processus à l'oeuvre dans le système voire d'agir dessus, que ce soit à l'aide d'une interface graphique où à l'aide de commandes.

#### Afficher les processus

##### Exemple 1 :

- Voici un résultat de la commande `ps -a -u -x` dans un environnement Linux.
  - L'argument `-a` permet d'afficher les processus de tous les utilisateurs.
  - `-u` d'afficher le nom de l'utilisateur plutôt qu'un identifiant numérique.
  - `-x` d'afficher les processus principaux.

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	835	0.0	0.8	258796	9064	?	Ssl	22:07	0:00	/usr/lib/upower
tlenne	840	0.0	0.6	240452	6860	?	Ssl	22:07	0:00	/usr/lib/gvfs/g
tlenne	848	0.0	0.4	159208	5048	?	Sl	22:07	0:00	/usr/lib/dconf/
tlenne	865	0.0	1.7	286904	17192	?	Ssl	22:07	0:00	/usr/lib/x86_64
tlenne	881	0.0	1.6	65388	16504	?	S	22:07	0:00	/usr/lib/x86_64
tlenne	882	0.0	1.9	67192	19688	?	S	22:07	0:00	/usr/lib/x86_64
tlenne	887	0.0	0.9	280752	10052	?	Ssl	22:07	0:00	/usr/lib/gvfs/g
tlenne	894	0.0	0.7	388168	7536	?	Sl	22:07	0:00	/usr/lib/gvfs/g
tlenne	900	0.0	0.5	164880	5528	?	Ssl	22:07	0:00	/usr/lib/gvfs/g
tlenne	904	0.0	0.5	175844	5720	?	Sl	22:07	0:00	/usr/lib/libred
tlenne	921	0.3	13.3	793192	134828	?	Sl	22:07	0:01	/usr/lib/libred
tlenne	938	2.1	22.7	2673536	229924	?	Sl	22:07	0:07	/usr/lib/firefd
tlenne	983	0.1	8.7	2384900	88592	?	Sl	22:07	0:00	/usr/lib/firefd
tlenne	1022	0.1	10.3	2400900	104176	?	Sl	22:07	0:00	/usr/lib/firefd
tlenne	1060	0.1	12.3	2412080	124908	?	Sl	22:07	0:00	/usr/lib/firefd
tlenne	1088	0.0	4.0	265584	41040	?	Sl	22:08	0:00	parole
tlenne	1094	0.4	4.1	497400	41548	?	Sl	22:08	0:01	xfce4-terminal
tlenne	1102	0.0	0.4	7680	4500	pts/0	Ss	22:08	0:00	bash
root	1186	0.0	0.0	0	0	?	I	22:12	0:00	[kworker/0:0-at
tlenne	1190	0.0	0.3	10916	3204	pts/0	R+	22:14	0:00	ps -a -u -x --f

- Pour en savoir plus sur cette commande et ses résultats, on pourra consulter l'aide à l'aide de la commande `man ps`
- Quelques observations :
  - La colonne `USER` indique le nom de l'utilisateur pour lequel a été lancé le processus.
  - La colonne `PID` donne l'identifiant numérique du processus.
  - Les colonnes `%CPU` et `%MEM` indiquent respectivement le taux d'occupation du processeur et de la mémoire par le processus/
  - La colonne `STAT` indique l'état du processus (la première lettre en majuscule):
    - `R` : *running* ou *runnable*, respectivement en exécution ou prêt (la commande `ps` ne fait pas la différence).
    - `S` : *sleeping*, en attente.
  - Les colonnes `START` et `TIME` indiquent respectivement la date/heure à laquelle le programme a été lancé et le temps total pendant lequel le processus a été dans l'état "en exécution".
  - La colonne `COMMAND` indique le nom du programme, plus précisément la commande qui a lancé le programme.

#### Terminer des processus

L'utilisateur peut demander l'interruption d'un processus à l'aide de la commande `kill`, en spécifiant le ou les numéros de processus (c'est à dire le(s) PID).

##### Exemple 2 :

- La commande `kill 921 938` envoie un signal de terminaison aux deux processus listés par les numéros données :
  - Cette demande peut être interceptée et réalisée par l'application concernée (ici le traitement de texte libre office), cela équivaut à fermer la fenêtre de celle-ci. Dans ce cas, il sera demandé à l'utilisateur s'il veut sauvegarder son travail.

- Il est également possible de terminer le processus sans passer par l'application concernée (ici avec l'option `-9` en saisissant `kill -9 921 938`). Dans ce cas, c'est le système qui termine le processus sans que l'utilisateur puisse sauvegarder l'état. C'est ce qui se passe par exemple lorsque l'application "ne répond plus".

## Remarques :

- Sous Linux, on peut aussi utiliser la commande `top` pour afficher les processus.
- Dans un environnement Windows, on peut utiliser la commande `tasklist` pour afficher les processus et `taskkill` pour terminer un processus.

```
C:\Users\ThomasLENNE>tasklist
```

Nom de l'image	PID	Nom de la session	Numéro de session	Utilisation
System Idle Process	0	Services	0	8 Ko
System	4	Services	0	132 Ko
Registry	124	Services	0	94 444 Ko
smss.exe	476	Services	0	1 216 Ko
csrss.exe	664	Services	0	5 560 Ko
wininit.exe	748	Services	0	7 044 Ko
csrss.exe	756	Console	1	6 040 Ko
services.exe	820	Services	0	11 396 Ko
lsass.exe	840	Services	0	19 392 Ko
svchost.exe	964	Services	0	3 428 Ko
svchost.exe	988	Services	0	32 484 Ko
fontdrvhost.exe	1012	Services	0	13 852 Ko
svchost.exe	624	Services	0	14 580 Ko
svchost.exe	860	Services	0	8 652 Ko
winlogon.exe	1048	Console	1	12 820 Ko
fontdrvhost.exe	1104	Console	1	17 200 Ko

**Exercice 2 :** Il se trouve dans le dossier [Exercices \(Exercices/Processus\\_Exercices.pdf\)](#)

## 4. Interblocage

- Ainsi , les processus d'un système s'exécutent de manière concurrente, dans un ordre hors de leur contrôle, défini par l'ordonnanceur du système d'exploitation. Ce fonctionnement possède de nombreux avantages :
  - Cela permet à un grand nombre de programmes de fonctionner "en même temps" à l'aide d'un seul processeur.
  - Les ressources sont optimisées (économie de calcul, d'énergie...)
- Cependant, lorsqu'un processus est interrompu, ce n'est pas de son fait, ce n'est pas lui qui décide. Il reprend plus tard son exécution , dans l'état où il était avant l'interruption :
  - Cela ne pose pas de problème tant que le processus n'a pas besoin de ressources externes à lui même.
  - Mais lorsque'un processus doit accéder à des ressources partagées (écrire dans un fichier comme dans l'exercice 1, ou accéder à un matériel comme un carte son), cela peut poser problème :
    - En particulier, lorsqu'une même ressource est attendue par plusieurs processus et qu'elle ne peut être utilisée par un seul, un blocage est alors possible: On appelle cela le phénomène d'interblocage, ou verrou mortel (*deadlock*)

### Exemple 3 :

Certaines ressources matérielles sont en accès exclusif: Cela signifie qu'elle ne peuvent être utilisée que par au plus un seul processus. Dans un ordinateur, c'est le cas de la carte son.

Imaginons deux processus :

- `record` : C'est un processus qui accède au micro de la carte son, et qui écrit le son enregistré dans une mémoire tampon de 10 secondes.
- `play` : C'est un processus qui accède au haut parleur de la carte son, et qui lit le contenu qu'il reçoit en entrée.

Classiquement, on utilise le processus `record` pour enregistrer du son, mémorisé ensuite dans un fichier, que nous appellerons `sound.mp3` . Puis on utilise le processus `play` pour lire le fichier et diffuser le son.

Supposons maintenant que l'on essaie d'exécuter le processus `record` en le dirigeant directement vers le processus `play` , sans passer par l'écriture de `sound.mp3` :

- Le processus `record` accède à la carte son et enregistre du son.
- Le processus `play` essaie alors d'accéder à la carte son mais se retrouve bloqué en attente que le processus `record` libère la carte son et ne peut jouer aucun son.
- Le processus `record` se retrouve également bloqué car il ne peut pas enregistrer plus que le contenu de la mémoire tampon qui se retrouve pleine.

Les deux processus sont alors en interblocage:

- `play` attend que la carte son soit libre pour pouvoir lire ce qu'il reçoit en entrée.
- `record` attend que sa mémoire tampon se vide pour libérer la carte son.

## Conditions de Coffman

- L'interblocage est le danger de la concurrence des processus dans un système monoprocesseur.
- L'informaticien Edward Grady Coffman Jr (1934-...) a le premier, en 1971, décrit quatre conditions nécessaires à la présence d'un interblocage, appelée conditions de coffman:
  - Exclusion mutuelle : Au moins une ressource est en accès exclusif
  - Rétention et attente : Un processus détient une ressource et demande une autre ressource détenue par un autre processus.
  - Non préemption : une ressource ne peut être rendue que par un processus qui la détient (sinon le système prendrait lui même la décision de libérer cette ressource)
  - Attente circulaire : les processus s'attendent mutuellement.
- Plusieurs stratégies existent pour éviter, détecter ou résoudre l'interblocage. Dans la plupart des utilisations des systèmes d'exploitation, la solution la plus souvent utilisée est de terminer les processus concernés lorsqu'ils sont en interblocage (par exemple à l'aide de `kill` )



**Exercice 3 :** Il se trouve dans le dossier [Exercices \(Exercices/Processus\\_Exercices.pdf\)](#)