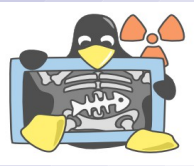
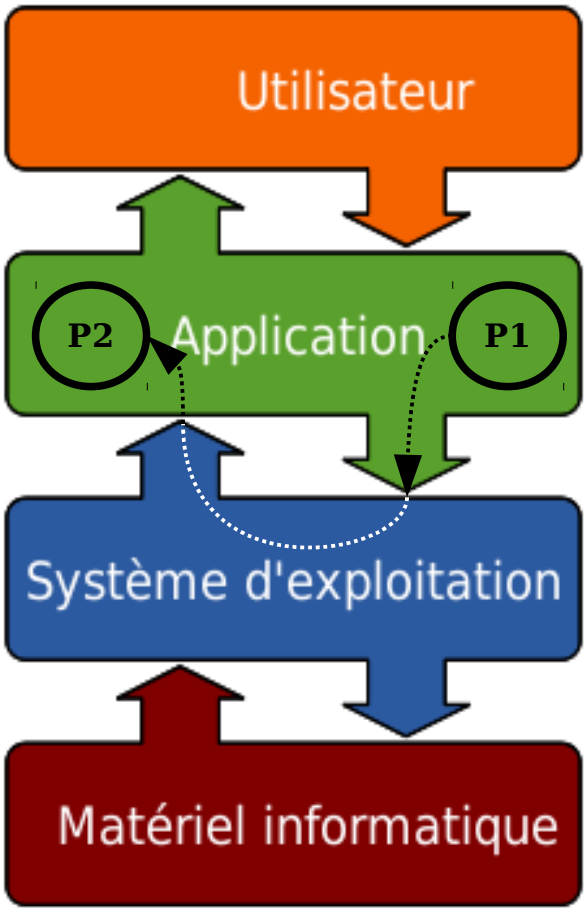
Programmation Système : les IPC



Les IPC (*Inter-Process Communication* ou communication inter processus) regroupent un ensemble de mécanismes permettant à des processus concurrents (ou distants) de communiquer.



La communication inter-processus

- Multi-tâches :
  Les processus ne sont pas isolés dans l'ordinateur puisqu'ils concourent tous à l'exploitation de travaux de l'utilisateur et puisqu'ils sont en compétition pour le partage de ressources qui sont limitées.

  ⇥ Ceci implique un besoin des processus de communiquer.

- Multi-programmation :
  L'idée (paralléliser des traitements) est de substituer un seul programme purement séquentiel en plusieurs pièces (processus ou tâches) pouvant logiquement se dérouler simultanément et en partageant des données.

  ⇥ Ceci implique un besoin des processus de communiquer.

Exclusion mutuelle

- Les ressources d'un système peuvent être réparties en deux catégories :
  - les ressources partageables, pouvant être utilisées simultanément par plusieurs processus (parfois limités en nombre),
  - les ressources non partageables, ne pouvant être utilisées que par un seul processus à un moment donné.

- Les ressources non partageables peuvent l'être pour deux raisons :
  - il est matériellement impossible de partager la ressource
  - le partage de la ressource entraînerait des risques d'interférence (imprimante, zone mémoire, etc...)

⇥ Le problème à résoudre consiste à s'assurer que les ressources non partageables ne sont attribuées qu'à un seul processus à la fois.

Section critique

- Une **section critique** est une portion de code dans laquelle il doit être garanti qu'il n'y aura jamais plus d'un processus ou *thread* simultanément.

  ⇥ Il est nécessaire d'utiliser des sections critiques lorsqu'il y a accès à des ressources partagées par plusieurs processus ou *thread*.

- Une section critique peut être protégée par un ***mutex*** (*mutual exclusion*), un **sémaphore** ou d'autres primitives de programmation concurrente.

Synchronisation et Interblocage

- Synchronisation : la vitesse relative de deux processus est imprévisible. Elle dépend de la fréquence des interruptions, de la durée du travail, de la fréquence d'attribution des processeurs aux processus. Les processus se déroulent donc de façon asynchrone et doivent ainsi synchroniser leurs activités à certains moment du traitement.

  ⇥ C'est le rôle du système d'exploitation de fournir un mécanisme pour cela.

- Inter blocage (*deadlock*) : lorsque plusieurs processus demandent à obtenir des ressources en même temps, une situation d'inter blocage peut se produire si les ressources requises par les uns sont occupées par les autres et vice versa.

  ⇥ Prévenir les effets de l'inter blocage est une fonction indispensable du système d'exploitation.

Les communications inter-processus (IPC)

- Les communications inter processus (*Inter-Process Communication* ou IPC) regroupent un ensemble de mécanismes permettant à des processus concurrents (ou distants) de communiquer.

- Ces mécanismes peuvent être classés en trois catégories :

  - les outils permettant aux processus de s'échanger des données ;

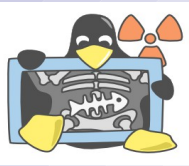
  - les outils permettant de synchroniser les processus, notamment pour gérer le principe de section critique ;

  - les outils offrant directement les caractéristiques des deux premiers (échanger des données et synchroniser des processus).

Échange de données

- Les **fichiers** peuvent être utilisés pour échanger des informations entre deux, ou plusieurs processus.
- La **mémoire** (principale) d'un système peut aussi être utilisée pour des échanges de données. Suivant le type de processus, les outils utilisés ne sont pas les mêmes :
  - Dans le cas des processus « classiques », l'espace mémoire du processus n'est pas partagé. On utilise alors des mécanismes de **mémoire partagée**, comme les segments de mémoire partagée pour Unix.
  - Dans le cas des processus légers (*threads*) l'espace mémoire est partagé, la mémoire peut donc être utilisée directement.
- Quelle que soit la méthode utilisée pour partager les données, ce type de communication pose le problème des sections critiques.

Synchronisation



- Les mécanismes de synchronisation sont utilisés pour résoudre les problèmes de sections critiques et plus généralement pour bloquer et débloquer des processus suivant certaines conditions :
  - Les **verrous** permettent de bloquer tout ou une partie d'un fichier.
  - Les **sémaphores** (et les **mutex**) sont un mécanisme plus général, ils ne sont pas associés à un type particulier de ressource et permettent de limiter l'accès concurrent à une section critique à un certain nombre de processus.
  - Les **signaux** (ou les **évènements**) permettent de communiquer entre processus : réveiller, arrêter ou avertir un processus d'un événement.
- L'utilisation des mécanismes de synchronisation est difficile et peut entraîner des problèmes d'interblocage (tous les processus seront alors bloqués).

Échange de données et synchronisation

- Ces outils regroupent les possibilités des deux autres et sont plus simples d'utilisation.
- L'idée est de communiquer en utilisant le **principe des *files*** (notion de boite aux lettres), les processus voulant envoyer des informations (notion de messages) les placent dans la *file* ; ceux voulant les recevoir les récupèrent dans cette même *file*. Les opérations d'écriture et de lecture dans la *file* sont bloquantes et permettent donc la synchronisation.
- Ce principe est utilisé par :
  - les files d'attente de message (*message queue*) sous Unix,
  - les *sockets* Unix ou Internet,
  - les tubes (nommés ou non), et
  - la transmission de messages (*Message Passing*).
- *Remarques : CORBA, DCOM, SOAP et MPI sont des exemples de systèmes de passage de messages.*

IPC System V (1)

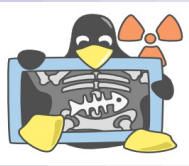
- Les IPC Système V recouvrent 3 mécanismes de communication entre processus (**Inter Processus Communication**) :

- Les **files de messages**, dans lesquelles un processus peut glisser des données ou en extraire. Les messages étant typés, il est possible de les lire dans un ordre différent de celui d'insertion, bien que par défaut la lecture se fasse suivant le principe de la file d'attente.

- Les **segments de mémoire partagée**, qui sont accessibles simultanément par deux processus ou plus, avec éventuellement des restrictions telles que la lecture seule.

- Les **sémaphores**, qui permettent de synchroniser l'accès à des ressources partagées.

IPC System V (2)



- Les IPC permettent de faire communiquer deux processus d'une manière asynchrone, à l'inverse des tubes.

- C'est à dire qu'au moyen des IPC, un processus pourra avoir accès à une information alors que le processus qui a généré l'information est terminé depuis longtemps.

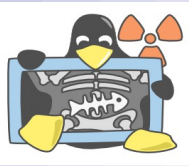
- La sauvegarde des informations est assurée par le système mais celui-ci ne garantit ces informations que s'il ne subit pas d'arrêt (car il n'y a pas de sauvegarde sur disque dans ce cas).

- Dans tous les cas, ces outils de communication peuvent être partagés entre des processus n'ayant pas immédiatement d'ancêtre commun.

IPC System V (3)

- La mise en oeuvre des IPC présente des points communs :
- les primitives de création et d'ouverture d'un objet se présentent sous la forme **`xxxget()`**. Elles attendent une clef comme l'un des paramètres et renvoient un identificateur. La clef est une donnée de type **`key_t`** qui doit être connue de tous les processus utilisant l'IPC.
- les primitives de contrôle de l'objet se présentent sous la forme **`xxxctl()`**. Elles permettent généralement d'obtenir les caractéristiques de l'objet, modifier ses caractéristiques ou détruire l'objet.
- Tout objet, géré par le noyau, est identifié par :
- une identificateur interne correspondant à l'entrée dans une table système et contenant les caractéristiques de l'objet,
- un identificateur externe, appelé la clé, utilisé par les processus utilisateurs pour manipuler l'objet.

IPC System V (4)



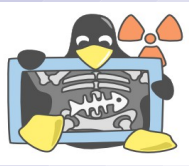
- Il existe deux types d'IPC : System V et POSIX. Les IPC System V sont une ancienne API (orientée UNIX).

	Files de messages	Mémoire partagée	Sémaphores
Headers	sys/types.h sys/ipc.h sys/msg.h	sys/types.h sys/ipc.h sys/shm.h	sys/types.h sys/ipc.h sys/sem.h
Création / Ouverture	msgget()	shmget()	semget()
Contrôle	msgctl()	shmctl()	semctl()
Opérations	msgsnd() msgrcv()	shmat() shmdt()	semop()
Structures associées	msqid_ds	shmid_ds	semid_ds

- Les IPC POSIX fournissent une interface bien mieux conçue (portable et "thread safe") que celles de System V. D'un autre coté, les IPC POSIX sont moins largement disponibles (particulièrement sur d'anciens systèmes) que ceux de System V.

- Plus de détails : **`man mq_overview`**, **`man sem_overview`**, **`man shm_overview`**

Segment de mémoire partagée



- Les processus peuvent avoir besoin de partager de l'information. Le système Unix fournit à cet effet un ensemble de routines permettant de créer et de gérer un segment de mémoire partagée.

- Un segment de mémoire partagée est identifié de manière externe par un nom qui permet à tout processus possédant les droits, d'accéder à ce segment. Lors de la création ou de l'accès à un segment mémoire, un numéro interne est fourni par le système.

- Parmi les mécanismes de communication entre processus, l'utilisation de segments de mémoire partagée est la technique la plus rapide, car il n'y a pas de copie des données transmises. Ce procédé de communication est donc parfaitement adapté au partage de gros volumes de données entre processus distincts.

Sémaphores

- Si deux processus écrivent et lisent "en même temps" dans une même zone de mémoire partagée, il ne sera pas possible de savoir ce qui est réellement pris en compte.

- D'où l'obligation d'utiliser des sémaphores pour ne donner l'accès à cette zone qu'à un processus à la fois.

tp-sys-ipc.pdf

Séquence 1 : Segment de mémoire partagée

Séquence 2 : Les sémaphores

Files de messages (1)

- Les files de messages sont des listes chaînées gérées par le noyau dans lesquelles un processus peut déposer des données (messages) ou en extraire. Elle correspond au concept de **boîte aux lettres**.

- Un message est une structure comportant un nombre entier (le type du message) et une suite d'octets de longueur arbitraire, représentant les données proprement dites.

- Les messages étant typés, il est possible de les lire dans un ordre différent de celui d'insertion. Le processus récepteur peut choisir de se mettre en attente soit sur le premier message disponible, soit sur le premier message d'un type donné.

Files de messages (2)

- Un processus peut émettre des messages même si aucun processus n'est prêt à les recevoir. Les messages déposés sont conservés après la mort du processus émetteur, jusqu'à leur consommation ou la destruction de la file.

- Les messages ne contiennent pas à priori le numéro du processus émetteur ni celui du destinataire. C'est au processus utilisateur de décider de l'interprétation de chaque message.

- Une propriété essentielle de la file de messages est que le message forme un tout. On le dépose ou on l'extrait en une seule opération.

Files de messages (3)

- Les **avantages** principaux de la file de message (par rapport aux tubes et aux tubes nommés) sont :
  - un processus peut émettre des messages même si aucun processus n'est prêt à les recevoir
  - les messages déposés sont conservés, même après la mort de l'émetteur, jusqu'à leur consommation ou la destruction de la file.
- Le principal **inconvénient** de ce mécanisme est la limite de la taille des messages ainsi que celle de la file.

tp-sys-ipc.pdf

Séquence 3 : Files de messages