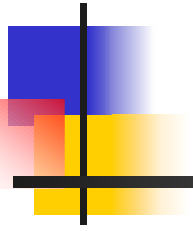
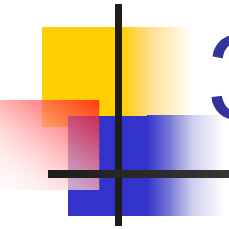
Chapitre III : La gestion des ressources
Notions fondamentales



- **Processus**
- **Ressources**
- **Concurrence**
- **Client Serveur**

La gestion des ressources

- La première fonction du SE est la gestion des ressources de l'ordinateur
- Nous allons étudier la gestion :
  - Du processeur
  - De la mémoire centrale
  - Des supports magnétiques (disques, bandes)
- Avant de rentrer dans les détails nous précisons quelques notions



3.1 Notion de processus

3.1.1 Définition

- La notion de processus est basée sur celle de programme (*terme apparu dans les années 60*)
- Un processus est un programme en exécution

3.1 Notion de processus

3.1.2 Évolutions

- La préhistoire :
  - Open shop : libre accès
  - Closed shop : opérateur
- Les moniteurs : petit SE permettant de lancer plus facilement les programmes
- Le traitement par lots
  - Un programme à la fois, les uns à la suite des autres
- Indépendance des E/S : améliore le rendement
- La multi-programmation : allocateur, ordonnanceur (scheduler) + mécanisme de pré-emption
- Le temps partagé : time sharing gérer plusieurs utilisateurs

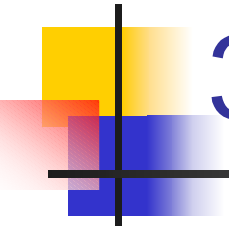
3.1 Notion de processus

3.1.3 Multi-programmation

- La multi-programmation consiste dans le traitement de plusieurs processus séquentiels entremêles
- Ce n'est pas réellement un traitement simultané
- Ce traitement simultané ne peut avoir lieu que si la machine possède plusieurs processeurs

3.1 Notion de processus

- On parle de multi-programmation pour dire que chaque processus (tâche) reçoit l'usage du processeur pendant une fraction de temps (moins d'une seconde)
- La rapidité de commutation des tâches fait que les utilisateurs ont l'impression d'une simultanéité dans l'exécution de la programme



3.1 Notion de processus

Problèmes :

- Comment partager le CPU entre plusieurs programmes
- Comment restaurer l'état de chaque programme
- Comment gérer la mémoire (limitée)
- Comment gérer les E/S
- Comment protéger les données en mémoire

3.1 Notion de processus

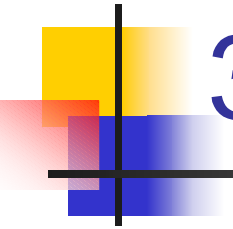
3.1.4 Application multi-utilisateurs

- Dans le cadre de systèmes de mono-programmation les programmes sont traités les uns à la suite des autres (file d'attente)
- Les anciens SE des micro-ordinateurs fonctionnaient ainsi jusque dans les années 90

3.1 Notion de processus

- L'utilisateur ne pouvait travailler qu'avec un seul logiciel à la fois (compilateur ou traitement de texte), jamais plusieurs en même temps
- Avec la multi-programmation, plusieurs programmes peuvent être traités en même temps.

3.1 Notion de processus



- Ce procédé permet l'emploi simultané de la machine par plusieurs utilisateurs
- La machine traite alors des processus qui sont la propriété de différents utilisateurs
- Grâce à ce mécanisme chaque utilisateur a l'impression d'avoir la machine pour lui seul (cf machine virtuelle Chapitre I)

3.1 Notion de processus

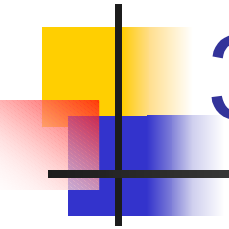
3.1.5 Les processus dans l'ordinateur

- Un certain nombre de processus existent aussi longtemps que l'ordinateur est en fonctionnement
- **Exemple :** la gestion des connexions utilisateur, gestion des disques
- A contrario, d'autres processus ont une durée de vie limitée
- **Exemple :** le processus de la commande `date` n'existe que le temps de la commande

3.1 Notion de processus

- **Arborescence des processus :** de façon générale le SE alloue un processus pour l'interface utilisateur (shell) puis d'autres processus (fils du premier pour chaque commande utilisateur ou pour chaque programme lancé
- Ainsi, il se crée une hiérarchie de processus (père fils) prenant la forme d'un arbre
- **Exemple :** Word et Excel ainsi que Delphi sont lancés à partir de l'interface graphique ce qui crée trois processus du même père. Si vous imprimez depuis Word, celui-ci crée un nouveau processus (fils) pour exécuter cette tâche

3.1 Notion de processus



- Un ordinateur personnel gère au moins 2 processus
- Un gros système (serveur) en gère plusieurs centaines : au moins un par utilisateur connecté (shell)
- Attention le SE a des limites fixées par le noyau (1024 en général)

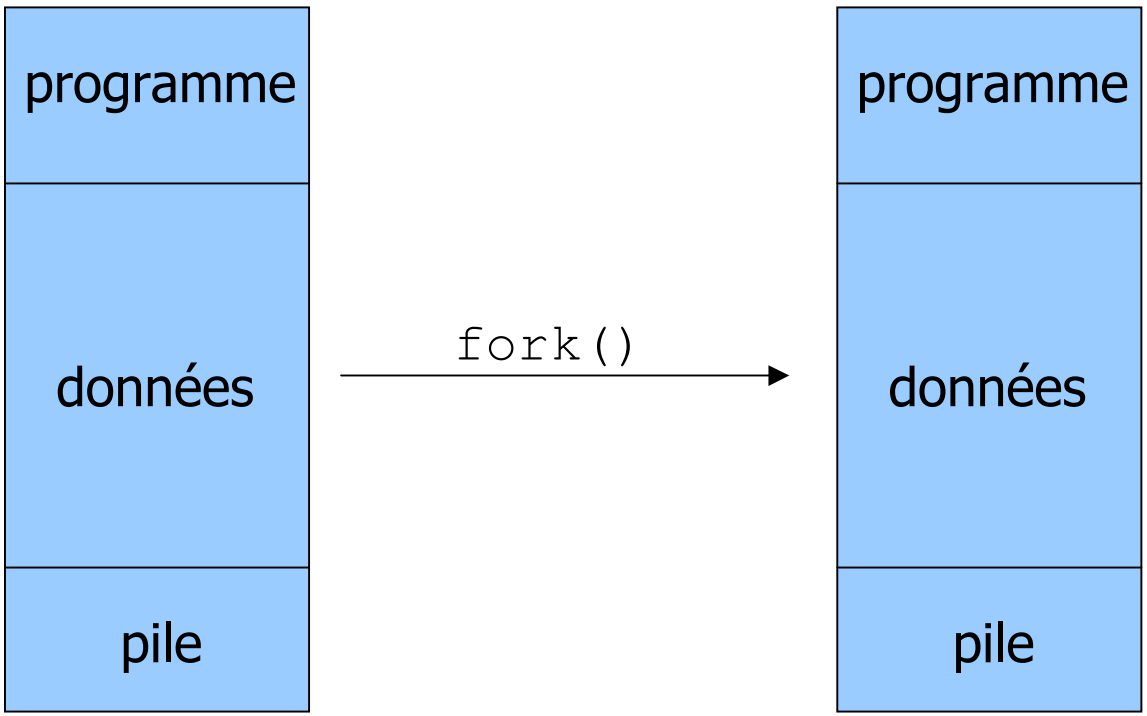
3.1 Notion de processus

3.1.6 Lancement des processus fils

- Un processus à trois façons de créer un processus fils :
  - Clonage : processus parallèle identique au père mais indépendant
  - Mise en route d'un processus fils avec interactions ou attente de terminaison de celui-ci
  - Recouvrement par le fils
- Permet de fournir une manière efficace pour organiser le travail de l'ordinateur
- La majorité des SE multi-tâches fonctionne sur ce principe

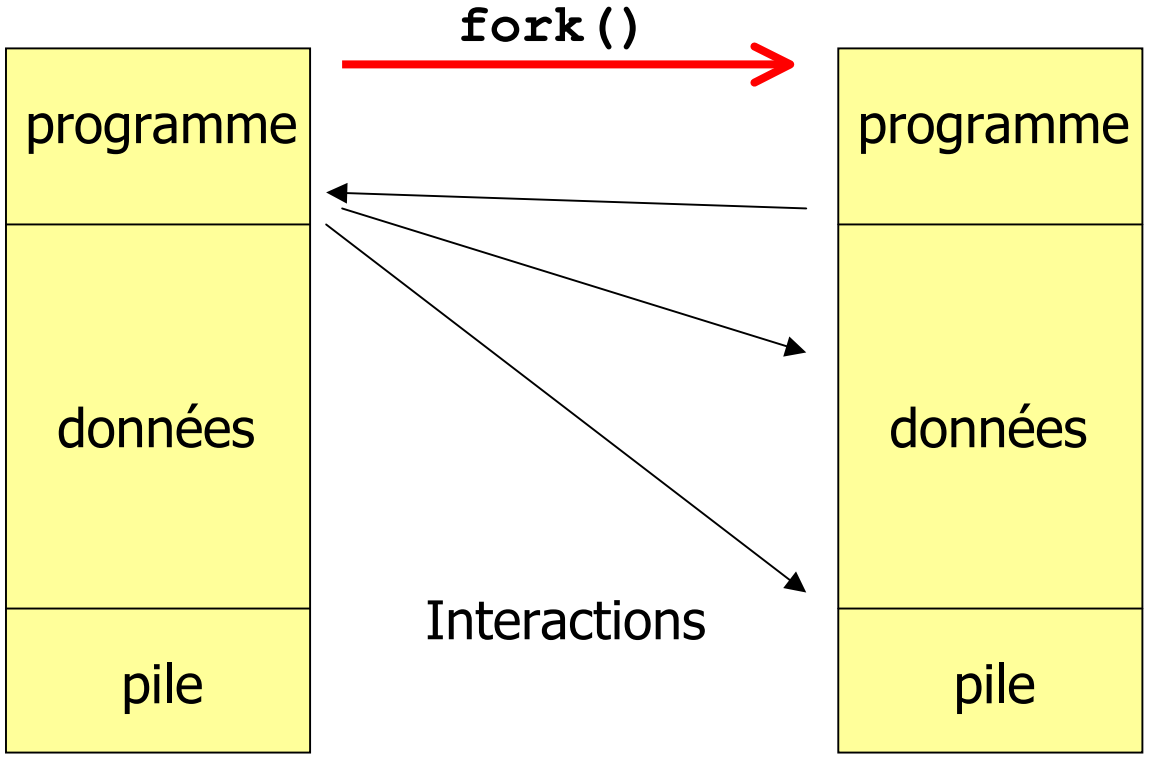
3.1 Notion de processus

Clonage



3.1 Notion de processus

Processus communiquant



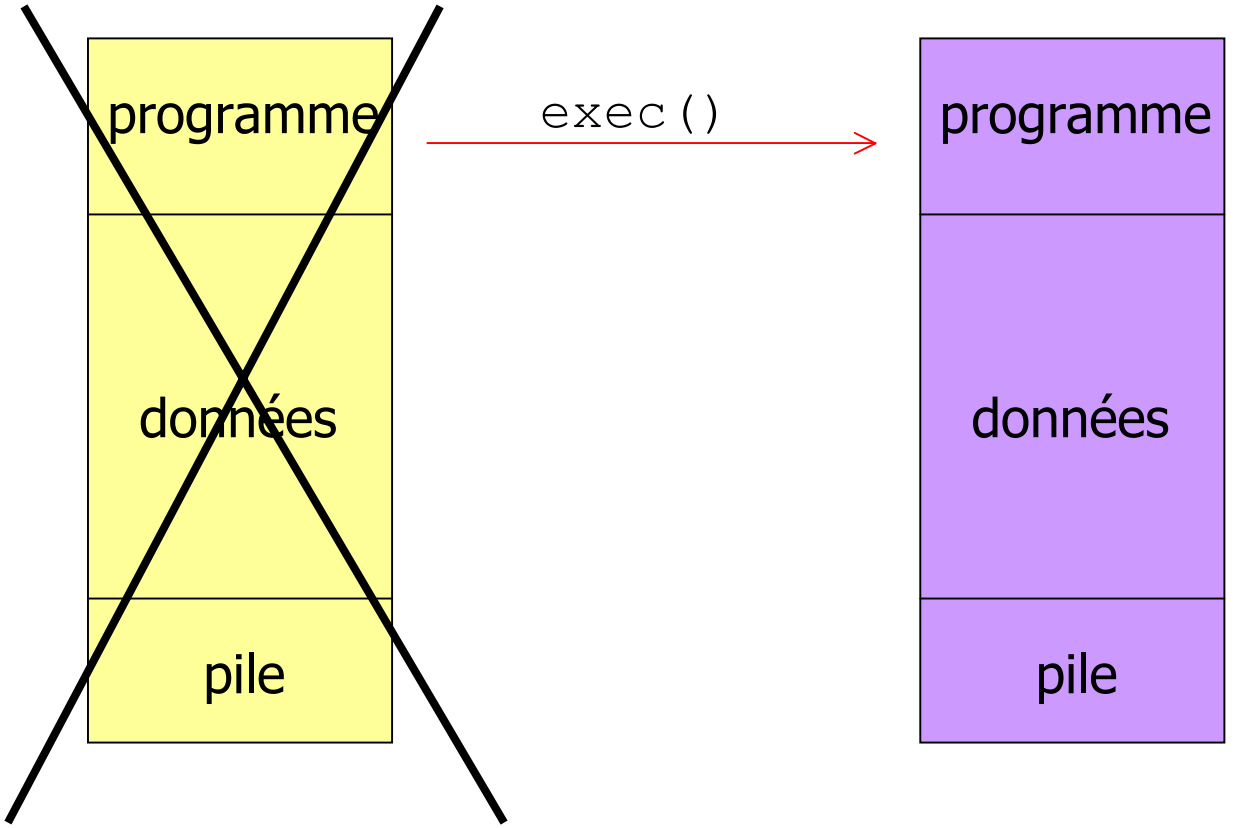
3.1 Notion de processus

- Exemple :

```
Id-fils:=fork();
Si Id_fils=0
  alors il s'agit du fils
  sinon il s'agit du père
```

3.1 Notion de processus

Recouvrement



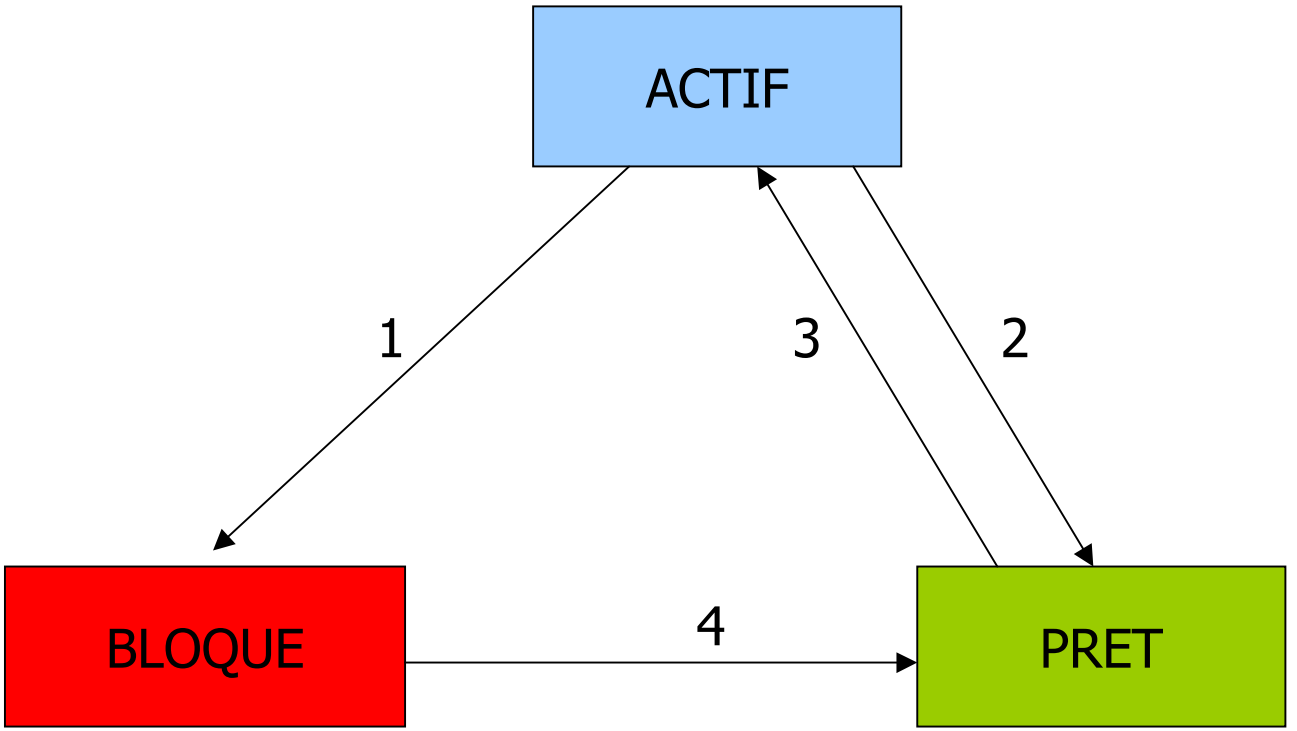
3.1 Notion de processus

3.1.6 Le cycle de vie d'un processus

- Un processus peut avoir trois états principaux :
  - En train d'être exécuté : on dit qu'il est **actif**
  - En attente d'être exécuté : il est **prêt** (toutes les ressources dont il a besoin sont disponibles)
  - En attente de ressources ou d'un signal provenant d'un périphérique : il est **bloqué**

- **Mort :** il n'existe plus et la place qu'il occupe dans les systèmes va être libérée

3.1 Notion de processus



Principaux états d'un processus

3.1 Notion de processus

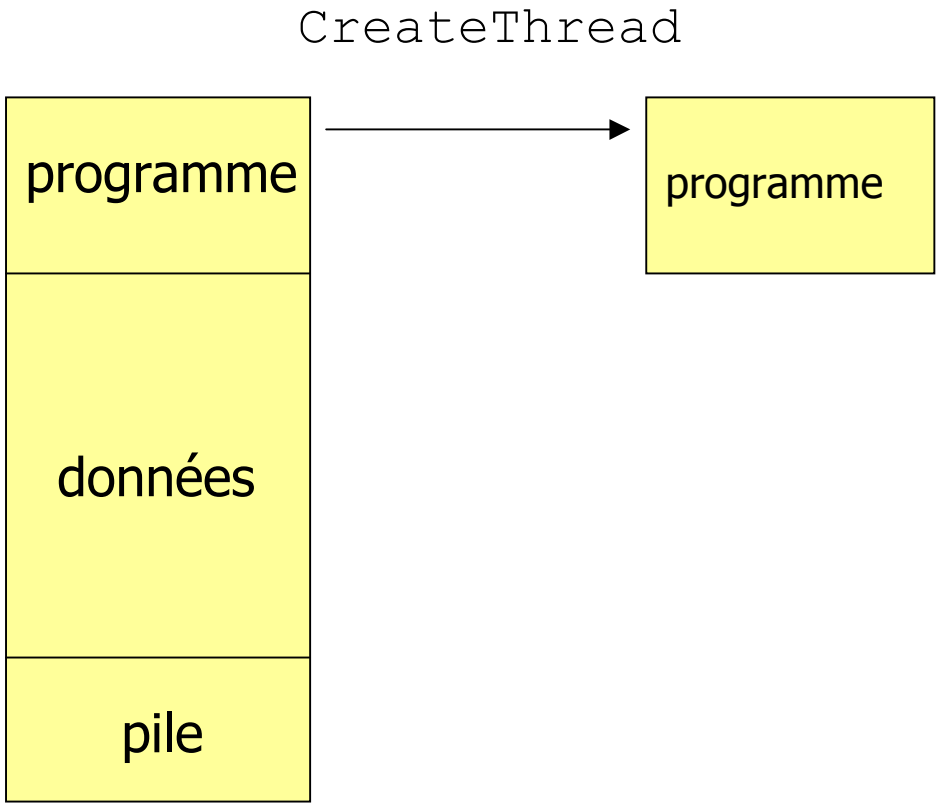
- Cas particulier du système UNIX :
  - **Les démons** : processus qui sont recréés automatiquement par le système (gestion des services réseau)
  - **Les zombies :** processus qui ne peuvent pas être tués (perte de père)

3.1 Notion de processus

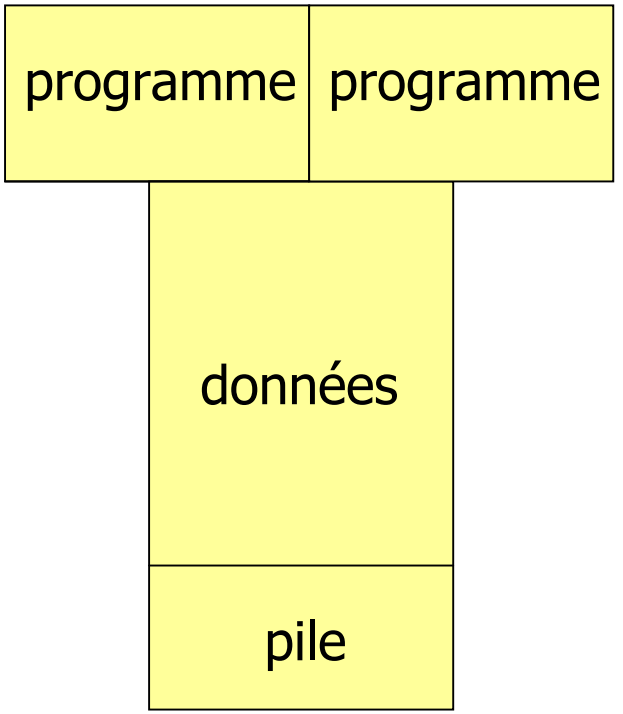
3.1.7 Les processus légers : threads

- Certains systèmes proposent en plus de la notion de processus, un mécanisme de processus légers
- Ce mécanisme :
  - Permet de créer plus rapidement des processus fils
  - Permet une communication facile entre un processus père et ses enfants

3.1 Notion de processus



3.1 Notion de processus



3.1 Notion de processus

Les threads existent dans :

- Les SE de la famille UNIX
- Windows NT et 2000
- WIN 9x et Me mais dans cette dernière famille, la notion de processus n'existe pas
  - Les programmes lancés depuis l'interface graphique sont des threads
  - C'est pour cette raison que le système peut être instable si un programme se plante (tous les threads partagent une zone de mémoire commune)

3.2 Notion de ressource

- On appelle ressource, tout élément qui peut être utile au déroulement d'un processus
- Une ressource peut être :
  - **Physique :** processeur, mémoire, périphérique
  - **Logique :** fichier, des données stockées
- Plusieurs processus peuvent avoir besoin des mêmes ressources

3.2 Notion de ressource

Suivant le type de la ressource, plusieurs cas peuvent se présenter :

- **Ressource partageable :** elle peut être attribuée en même temps à plusieurs processus.
  - **Par exemple :** mémoire centrale, fichiers
  - Partageable par un nombre max de processus

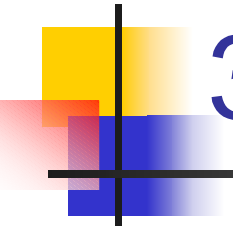
3.2 Notion de ressource

- **Ressource non partageable :**
- si elle ne peut être utilisée que par un seul processus à la fois (imprimante)
- Le processus qui a obtenu la ressource la libère une fois son travail terminé (imprimante) ou doit libérer la ressource après un certain temps d'utilisation (processeur)
- Processus en exclusion mutuelle

3.3 Notion de concurrence

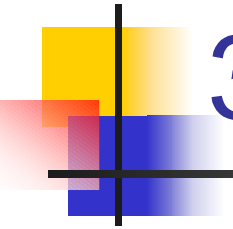
- Pour accéder à une ressource non partagée, il faut demander à un arbitre l'accès à cette ressource (moniteur)
- Lorsqu'une ressource est partagée, si plusieurs processus accèdent à la ressource, des **incohérences** peuvent en résulter

3.3 Notion de concurrence



- Concurrence d'accès aux fichiers ou aux données
- Exemple des guichets automatiques de retrait d'argent

3.3 Notion de concurrence



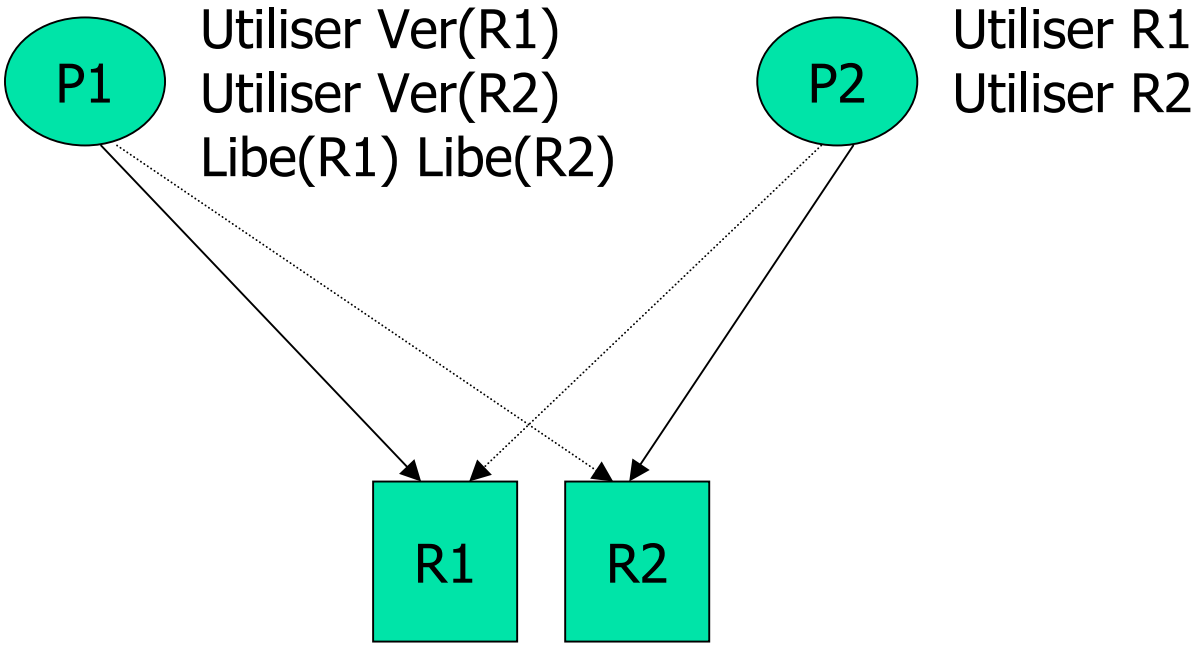
- Pour résoudre ce problème on utilise la notion de **verrou** (lock)
- Permet de résoudre l'exclusion mutuelle
- 2 opérations :
  - `Verouiller(v)` permet à un processus d'acquérir un verrou, s'il n'est pas disponible, le processus est bloqué en attente du verrou
  - `Déverouiller(v)` permet de libérer un verrou

3.3 Notion de concurrence

- Il existe aussi des sémaphore (Dijkstra 1965)
- C'est un distributeur de jeton (n jetons)
- Si un seul jeton alors le sémaphore est un verrou
- 2 opérations :
  - P(s) obtenir un jeton
  - V(s) restituer un jeton
- Création d'une file d'attente de processus

3.3 Notion de concurrence

- Le problème des verrous mortels



3.3 Notion de concurrence

Solution aux verrous mortels (interblocage) :

- Politique de l'autruche ?
- Détection guérison
- Prévention

3.4 Le paradigme Client Serveur

Communication entre processus :

- Schéma producteur consommateur
- Le producteur dépose dans un entrepot (tampon)
- Le producteur ne peut pas deposer s'il n'y a plus de place
- Le consommateur ne peut pas prendre un message qui est en train d'être déposé
- Le consommateur ne peut pas retirer d'éléments s'il n'y en a plus