Les Processus

Partie 1

Terminologie

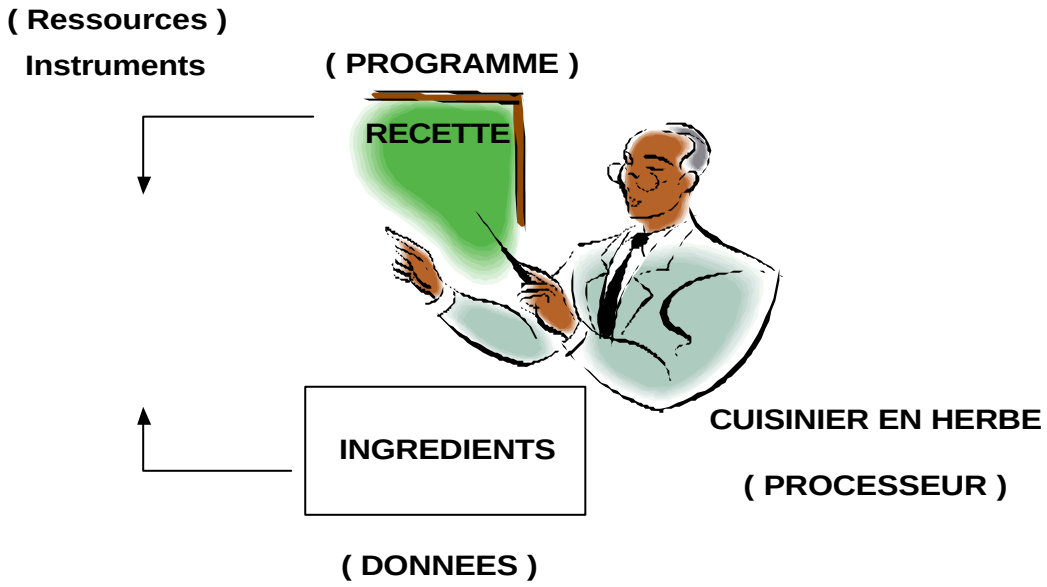
Partie 1a

Les Processus

- Définition :
  - Suite d’actions obtenues par l’exécution d’une séquence d’instructions dont le résultat constitue une des fonctions du système d’exploitation.
  - Entraîne l’exécution de plusieurs programmes et symétriquement un programme peut être impliqué dans plusieurs processus.

Les Processus

- Exemple : La recette du gâteau au chocolat
- Processus : Activité de transformation des ingrédients en gâteau



• Le processus représente l'abstraction d'un programme en cours d'exécution. Il possède un programme, des données en entrée et en sortie ainsi qu'un état courant.

Notion de Processus (1)

- **Programme** : décrit les actions à entreprendre
  - Ensemble de modules sources
  - Ensemble de modules objets
  - Résultat de l’édition des liens
- **Processeur :**
  - Entité matérielle capable d'exécuter des instructions
- **Processus :**
  - Entité dynamique correspondant à l’exécution d’une suite d’instruction
  - Concept abstrait
  - Le processeur fait évoluer le processus
- **Descendance des processus :**
  - Arbre des processus créés par un processus

Notion de Processus (2)

- **Système :**
  - Doit distinguer les différents processus
  - Doit représenter l'état d'un processus
- **Etat :**
  - Localisation du programme
  - Données propres
  - Variables
  - Registres
- **Ressource critique :**
  - Un seul processus peut y accéder à la fois
- **Ressource n point d'accès :**
  - Ressource attribuée n fois

Activation du Processus

- Un processus est activé par l'intermédiaire d'un agent appelé **processeur** qui exécute le programme associé**.**

- Le processus est vu comme **un objet logiciel** géré par le système d'exploitation au même titre que les fichiers.

- **Le système d'exploitation** s'occupe de la gestion des processus et peut être amené à commuté d'un processus à un autre en fonction des impératifs liés à l'exécution du système : *la vitesse d'exécution d'un processus n'est donc pas uniforme.*

Les Processus

- Exemple : La recette du gâteau au chocolat
- Evénement : « Piqûre d'abeille »
- Interruption du travail : « Enregistre l'endroit de la recette »
- Programme : « Livre de première urgence »
- Processus : « soin à apporter, calmer son fils »
- Reprise du travail de cuisinier
- Restitution de l'état du processus

- *Dans ce contexte, l'état du processus courant est sauvegardé.* ***Le processeur passe d'un processus plus prioritaire à un autre et chacun des processus possède un programme propre.***

Les Processus

- Exemple : La recette du gâteau au chocolat
- Evénement : « Nouveaux invités »
- Fabriquer deux gâteaux
- Deux processus indépendants sur le même programme
  - Un seul programme pour les deux processus chargé en mémoire

→ • *Programme réentrant*,

  - Le code ne se modifie pas lui-même
  - **Séparation des données propres de chacun des processus**
  - **Zone de variables distinctes**

→ *Processeur virtuel, temps virtuel*

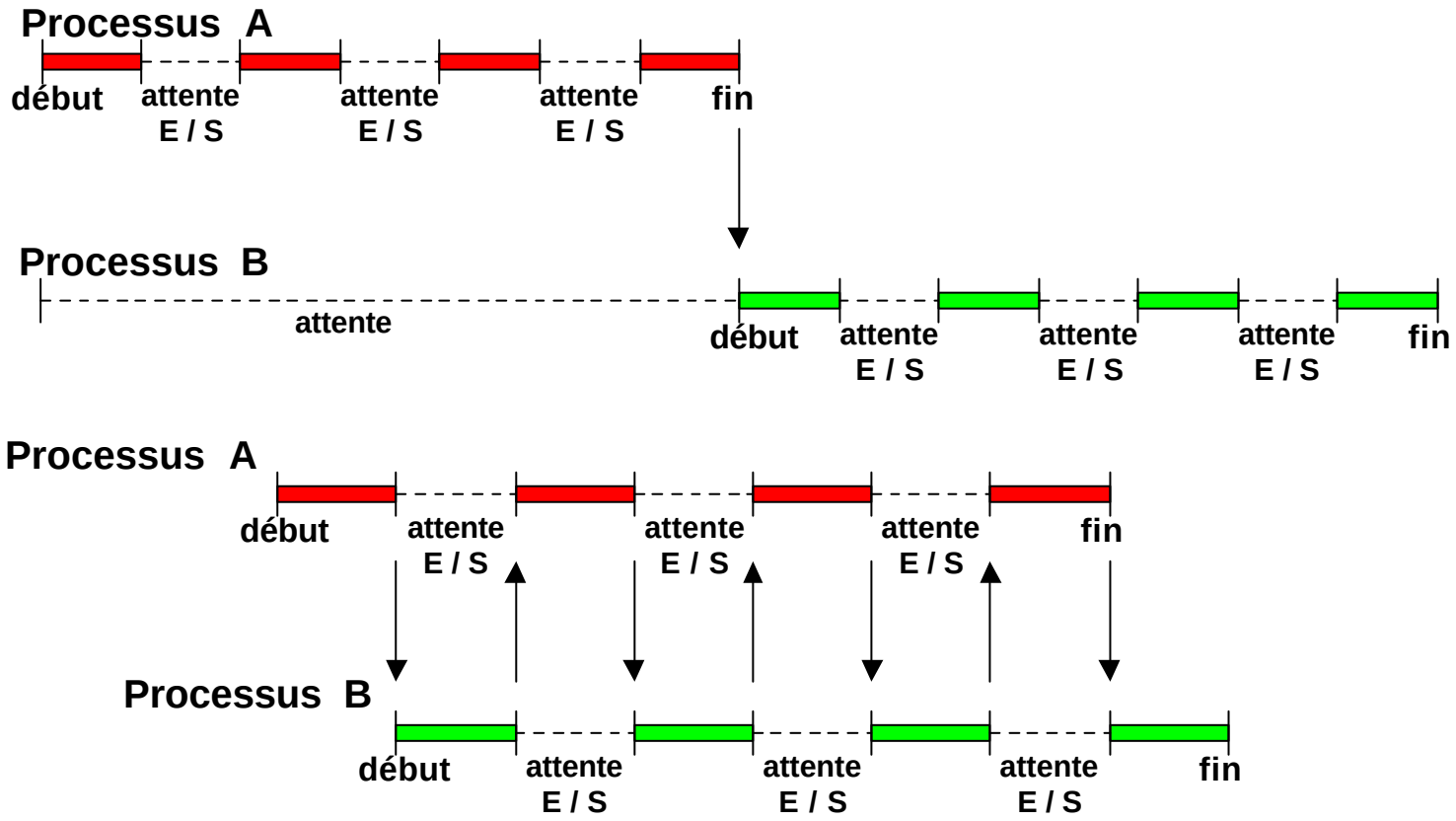
Processeur et Temps Virtuels

• Attribuer à chaque processus qu'il possède en propre, et qui exécute les instructions du programme pour le compte du processus

• Le système implante ces processeurs virtuels en attribuant le (ou les) processeur réel alternativement aux différents processeurs virtuels.

• Chaque processus avance au rythme de son processeur virtuel, induisant **un *temps virtuel***, qui correspond au temps d'utilisation du processeur virtuel.

Multiprogrammation et Utilisation efficace du CPU



Le Contexte d’un processus

Partie 1b

Le Contexte d'un processus

- **Un processus correspond à l'exécution d'un programme**

- **Plusieurs processus peuvent utiliser le même programme**

- **Les Processus se distinguent par :**
  - **leur zone de données distinctes**
  - **leur contexte qui leur est propre**

Le Contexte d'un processus

• **Le contexte de processus est l'ensemble des données qui permettent de reprendre l'exécution d'un processus qui a été interrompu.**

• **Constitution :**

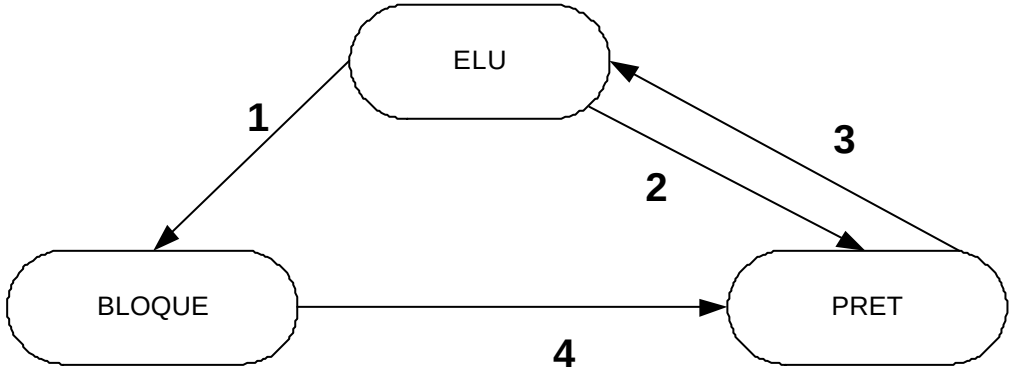
- son état,
- son mot d'état : - la valeur des registres actifs,
- - le compteur ordinal,
- la valeur des variables globales et dynamiques,
- son entrée dans la table des processus,
- sa zone U,
- les piles user et système,
- les zones de codes et de données.

Quand il y a changement de processus courant, il y a une commutation de mot d'état et d'un changement de contexte.

Le Contexte d'un processus

- **Etat des processus :**

- **Elu** : en cours d'exécution
  - *dispose de toutes les ressources dont il a besoin*
- **Bloqué** : en attente d'un événement extérieur pour pouvoir continué
  - *a besoin d'au moins une ressource autre que le processeur physique*
- **Prêt** : suspendu provisoirement pour permettre l'exécution d'un autre processus
  - *dispose de toutes les ressources à l'exception du processeur physique*



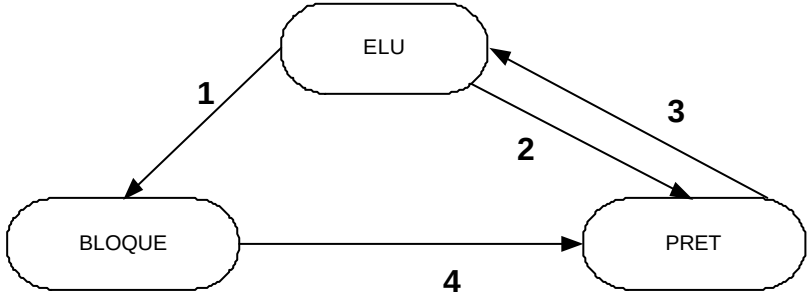
Le Contexte d'un processus

- **Transitions des Etats des processus :**

- **T1 : Elu ⇨ Bloqué**
  - *Exprime le besoin de disposer d'une nouvelle ressource*
    - explicite, sous forme **d'une demande au système**,
    - implicite, sous forme **d'un accès à cette ressource non allouée**

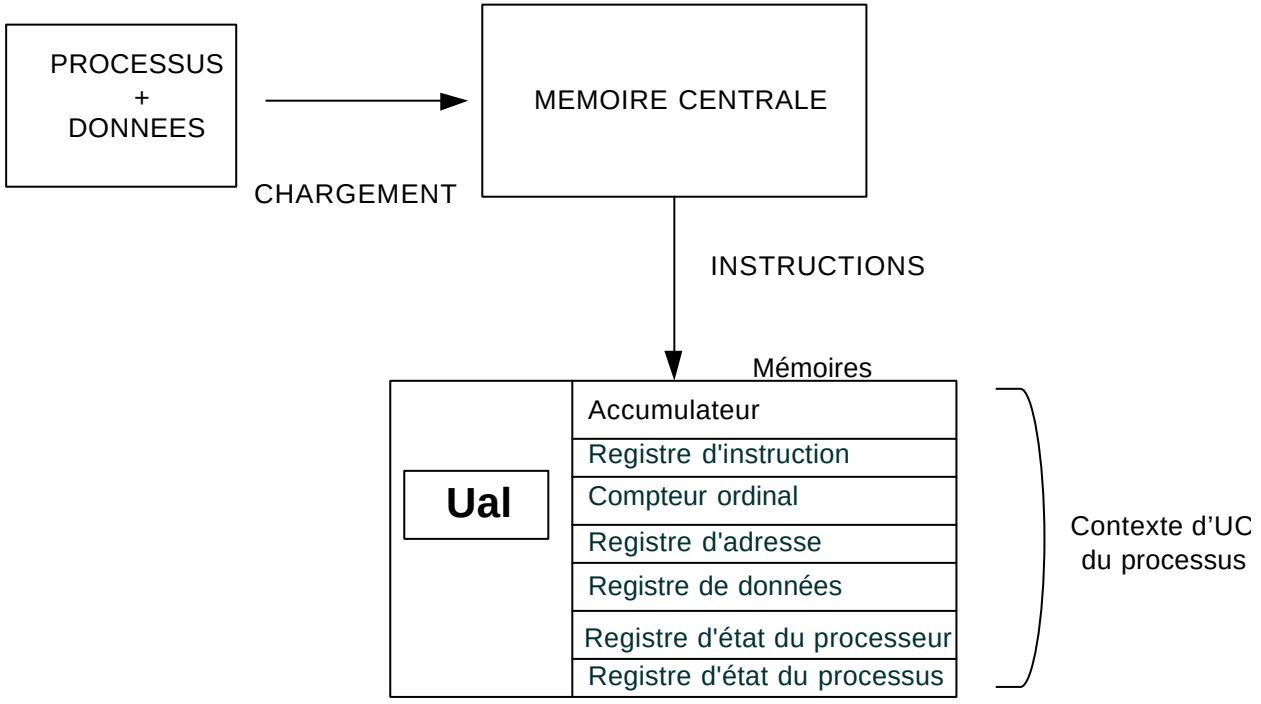
- **T4 : Bloqué ⇨ Prêt**
  - *Conséquence d'un événement extérieur au processus*

- **T1 et T2 : Elu ⇨ Prêt / Prêt ⇨ Elu**
  - *Le système alloue le processeur physique.*



Le Contexte d’un processus

- **Commutation de mot d’état**
  - **Pour être exécutés, un processus et ses données sont chargés en mémoire centrale**
  - **Les instructions du programme sont transférées de la mémoire centrale vers l’unité centrale où elles sont exécutées.**



Le Contexte d'Unité Centrale du processus

Les quelques registres spécialisés rappelés dans le tableau ci-dessous forment le **contexte d'unité centrale d'un processus.**

Registre	Descriptif
L'accumulateur	Il reçoit le résultat d'une instruction. Sur les machines à registres multiples, le jeu d'instructions permet souvent d'utiliser n'importe lequel des registres comme accumulateur.
Le registre d'instruction	Il contient l'instruction en cours.
Le compteur ordinal	Ce compteur contient l'adresse de la prochaine instruction mémoire à exécuter. Il change au cours de la réalisation d'une instruction pour pointer sur la prochaine instruction à exécuter. La majorité des instructions ne font qu'incrémenter ce compteur. Les instructions de branchement réalisent des opérations plus complexe sur ce compteur : affectation, incrémentation ou décrémentation plus importantes.
Le registre d'adresse	Il permet de spécifier une adresse en mémoire.
Les registres de données	Ils sont utilisées pour lire ou écrire une donnée à une adresse spécifiée en mémoire.
Les registres d'état du processeur	Le processeur dispose de plusieurs registres : actif, mode (user/system), vecteur d'interruption, etc.
Les registres d'état du processus	Droits, adresses, priorités, etc.

La table des processus ou BCP

- **Elle permet de sauvegarder les informations relatives au processus à l'exception du contenu de son propre espace mémoire**

- **Tableau ou Liste chaînée de structures**

- **Chaque processus en cours à sa structure qui lui est propre**

- **Un processus se caractérise donc par :**

  - **Adresse mémoire (image du noyau)**

  +

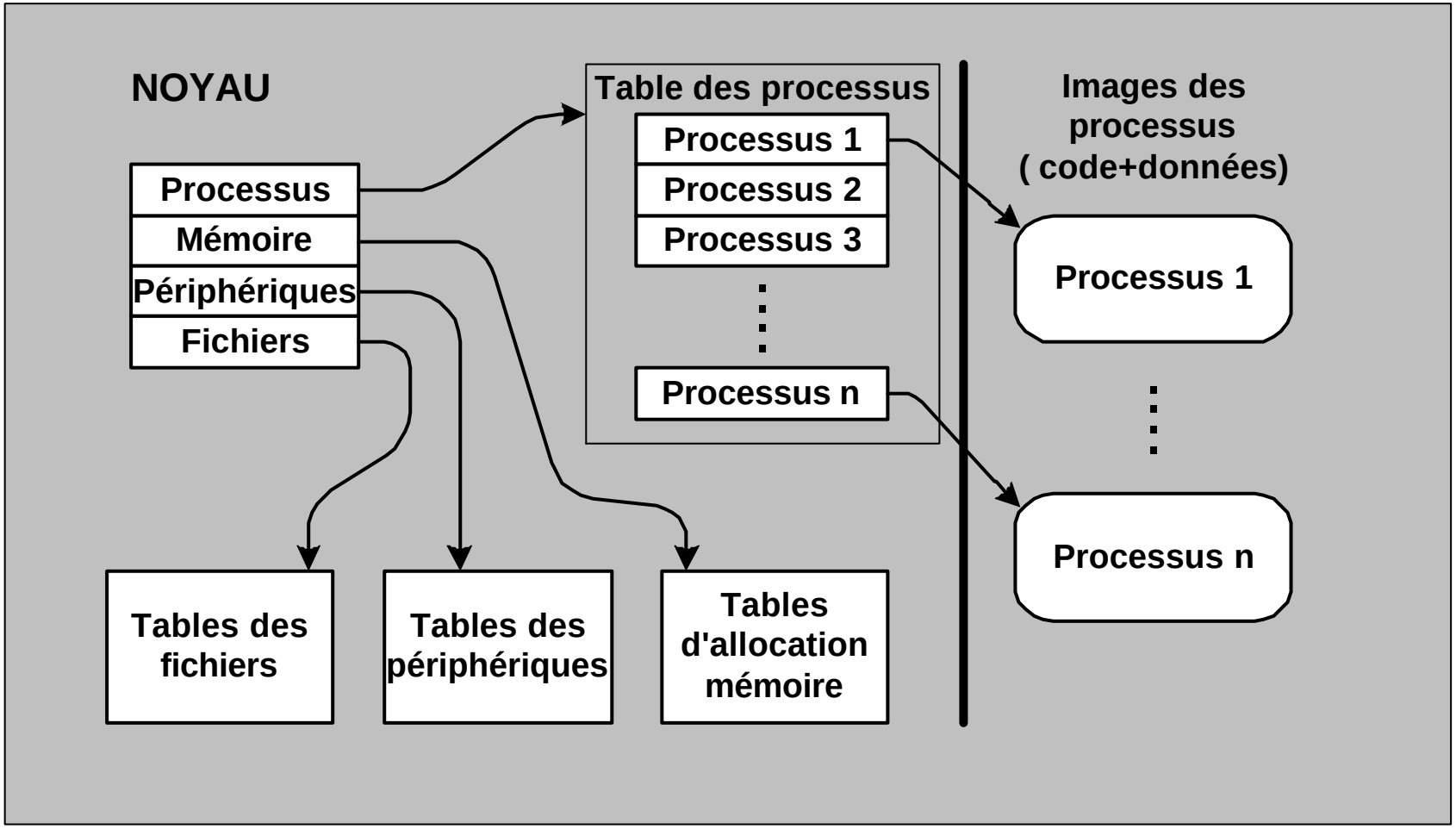
  - **Entrée dans la table des processus**

La table des processus

• Cette structure contient les informations qui doivent toujours être accessibles par le noyau et qui permettent de reprendre l'exécution d'un processus interrompu.

- • ***Etat :*** ce champ permet au noyau de prendre des décisions sur les changements d'état à effectuer sur le processus.
- • ***Adresse,*** **Taille et Localisation** en mémoire (centrale, secondaire). Ces informations permettent de transférer un processus en ou hors mémoire centrale.
- •***UID propriétaire du processus***, permet de savoir si le processus est autorisé à envoyer des signaux et à qui il peut les envoyer.
- •***PID, PPID*** l'identificateur du processus et de son père. Ces deux valeurs sont initialisées pendant l'appel système fork().
- • ***Evénement*** un descripteur de l'événement attendu quand le processus est dans un mode endormi.
- • ***Priorités*** **:** Plusieurs paramètres sont utilisés par l'ordonnanceur pour sélectionner l'élu parmi les processus prêts.
- • ***Vecteur d'interruption*** **du processus** : Ensemble des signaux reçus par le processus mais pas encore traites.
- •***Divers*** des compteurs utilises par le système
- •***Adresse de la zone u***

La table des processus



La zone U

• Cette structure contient des informations sur les données privées du processus, uniquement manipulables par le noyau.

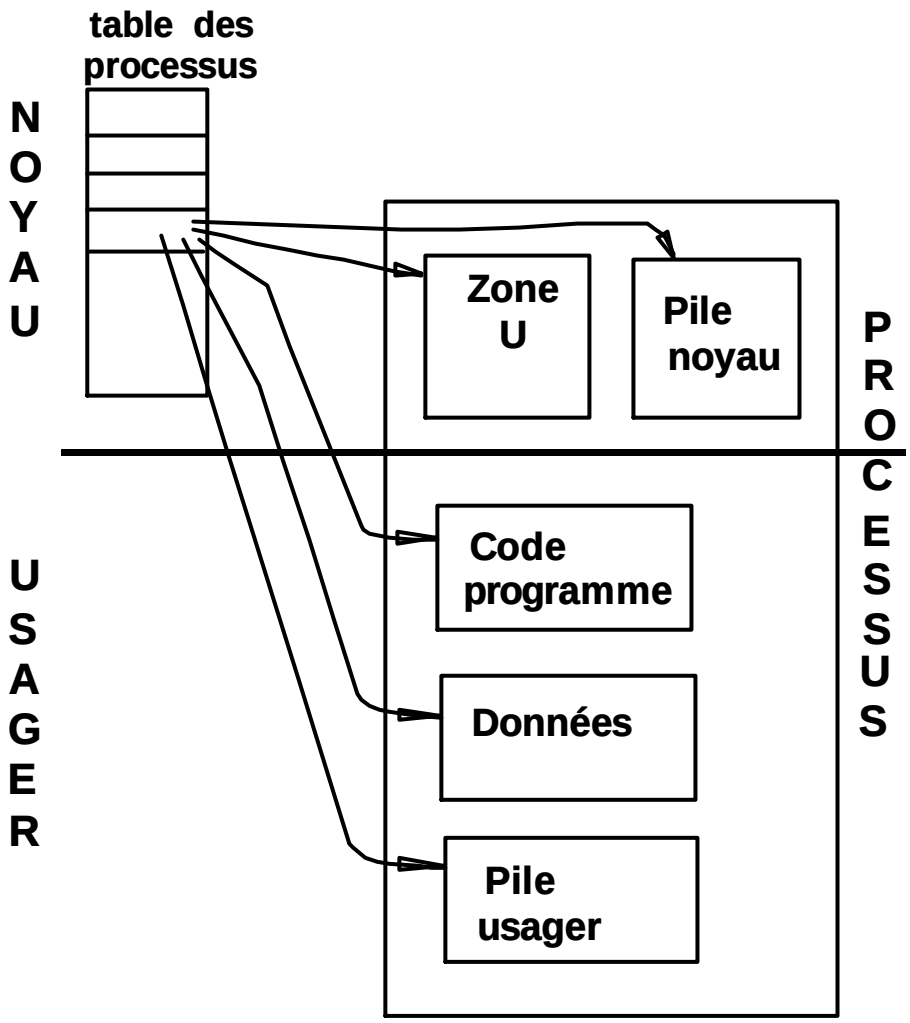
• Contenu de la zone u :

- ***Pointeur*** sur la structure de processus de la table des processus.
- ***UID de l'utilisateur*** qui détermine les divers privilèges, donne au processus les droits d'accès à un fichier, les changements de priorité, etc.
- ***Compteurs des temps*** (users et system) consommés par le processus
- **Masque de signaux**
- ***Valeur de la dernière erreur*** rencontrée pendant un appel système.
- ***Valeur de retour du dernier appel système***.
- ***Les structures associées aux entrées-sorties***,
- ***Les paramètres utilises par la bibliothèque standard***, adresses des buffers, tailles et adresses de zones à copier, etc.
- ***"." et "/"*** le répertoire courant et la racine courante
- ***Limites*** de la taille des fichiers de la mémoire utilisable etc. ..

Table des processus

ESPACE MÉMOIRE

D'UN PROCESSUS



La zone U doit être en mémoire seulement quand le processus est choisi pour s'exécuter

La gestion de la mémoire

• Permettre aux différents programmes en cours d'exécution de se partager cette ressource toujours trop limitée.
• Découpage de la mémoire en page i.e en bloc d’octets contigus
• Utilisation de **l’adressage linéaire** qui consiste à proposer comme espace d’adressage une entité linéaire et connexe
• Utilisation du **swap** qui consiste à allouée à un processus une page mémoire qui n'a pas été utilisée depuis un certain temps.
• Avantage : Plusieurs processus peuvent se **partager la même page mémoire** physique, sans même s'en rendre compte.

Swap + Adressage linéaire = Adressage Virtuel

Les interruptions

• Un interruption est une **commutation de mot d'état** provoquée par un signal produit par le matériel.

• Conséquence d'un événement extérieur ou intérieur

⇨ Modification d'un indicateur « le vecteur d'interruption » qui contient les adresses de procédure de traitement de chacune des interruptions.

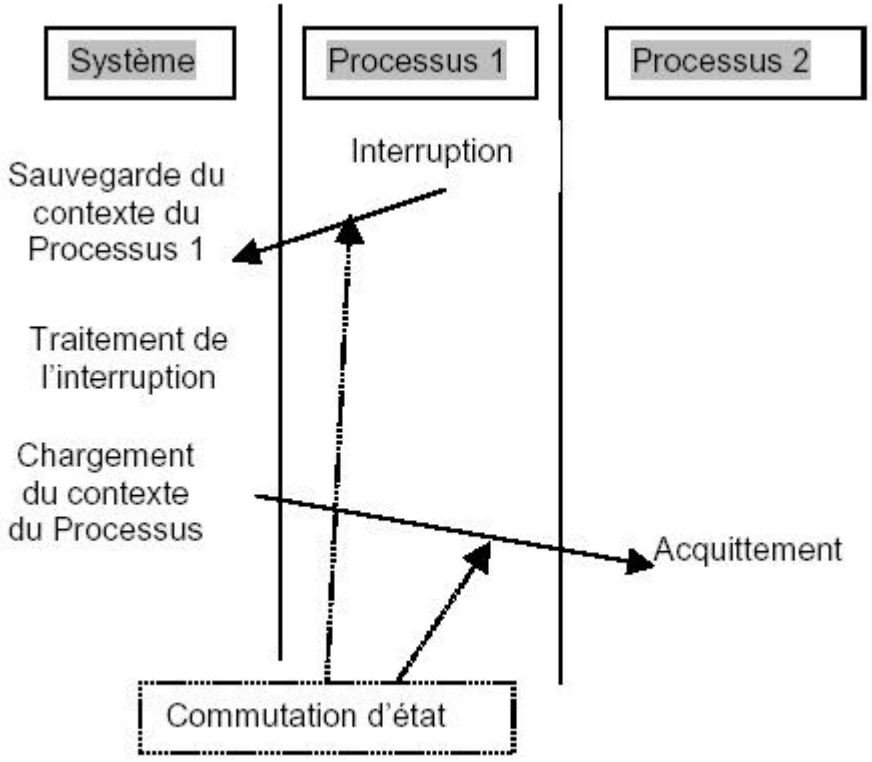
•Trois grands types :

- Externes, indépendante du processus
- Déroutements erreur interne du processeur
- Appels systèmes (E/S)

Niveau d'interruption	Nature de l'interruption	Fonction de traitement
0	Horloge	clockintr
1	Disques	diskintr
2	Console	ttyintr
3	Autres périphériques	devintr
	…	…

Les interruptions

Le traitement de l'interruption :



L’ordonnancement des processus

Partie 1c

Ordonnanceur

- Objectifs :
  - Maximiser l'utilisation du processeur
  - Equitable entre les différents processus
  - Présenter un temps de réponse acceptable
  - Assurer certaines priorités
- Pour cela un bon algorithme d'ordonnancement doit :
  - Que chaque processus reçoit sa part de quantum
  - Utiliser le temps processeur à 100 %
  - Minimiser le temps de réponse pour les utilisateurs en mode intéractif
  - Minimiser l'attente des utilisateurs qu travaillent en batch

Ordonnanceur

- **Ordonnancement avec réquisition**
  - Utilisation d'un compteur de temps (timer) qui génère une interruption périodiquement.
  - A chaque interruption de l'horloge : Poursuite de l'exécution ?
    - *Le processus est suspendu et le processeur est alloué à un autre processus*
    - *Un processus peut être suspendu à n'importe quel moment sans avoir été prévenu cela peut conduire à des conflits d'accès.*
- *Méthode jusqu'à achèvement*

Ordonnanceur

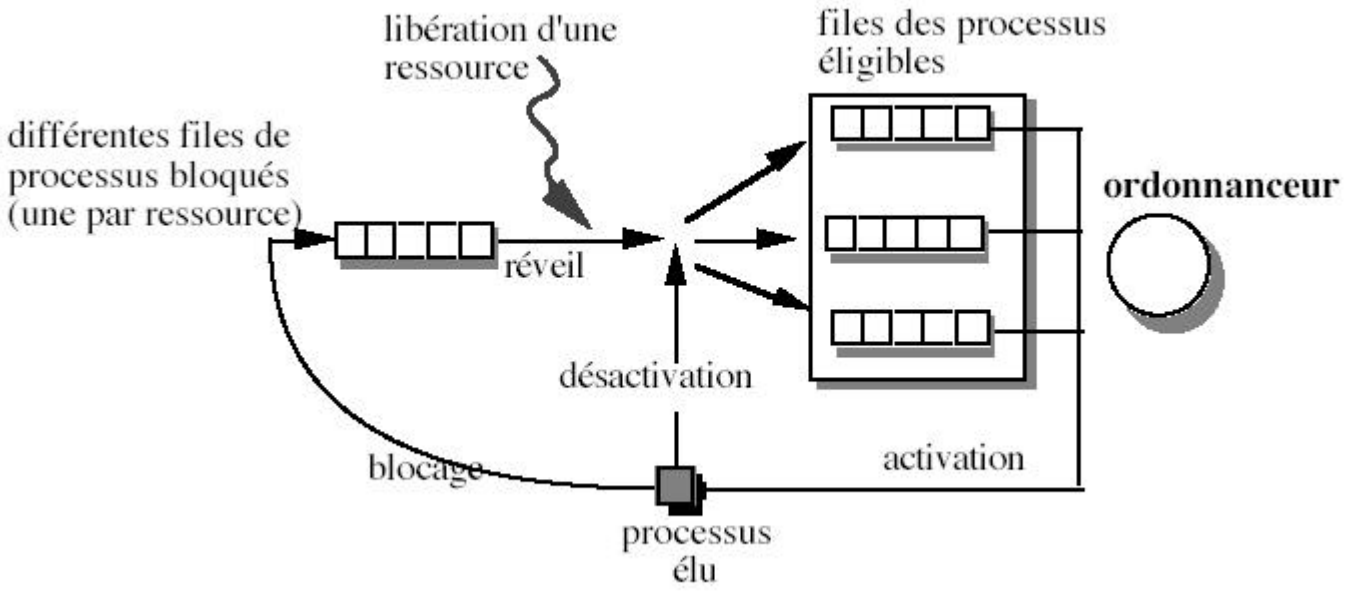
- **Tourniquet**
  - Chaque processus prêt dispose d'un quantum de temps pendant lequel il s'exécute
  - Lorsqu'il a épuisé ce temps, ou qu'il se bloque : le processus suivant est élu et le remplace
  - Le processus suspendu est mis en queue du tourniquet
  - Paramètre à régler : Gestion du quantum
  - Avantage : Utilisation équitable du processeur pour les processus présents en mémoire

Ordonnanceur

- **Les priorités**
  - Nécessité de priorisé certains processus
  - L’algorithme de priorité choisit le processus à priorité le plus élevé
  - Ces priorités peuvent être statiques ou dynamiques
    - *priorités statiques fortes non modifiables*
    - *Les processus utilisateurs verront leur priorité modifié au cours de leur exécution par l’ordonnanceur*
  - Avantage : les processus système sont exécuter en premier lieu

Ordonnanceur

- **Tourniquet avec priorités**



Ordonnanceur

- **Autres algorithmes d'ordonnancement**
  - PAPS (Premier Arrivé Premier Servi)
    - *L'ordre d'exécution est identique à l'ordre d'arrivée dans la file des processus éligibles*
  - PCA (Plus Court d'Abord)
    - *On exécute les processus qui ont le temps d'exécution le plus court : prédiction de ce temps + durée maximale*

La création des processus
La destruction des processus

Partie 1d

Création de processus

- **Un processus peut en créer un autre**
  - Processus Père peut créer un processus fils qui lui-même..
  - Suite partiellement ordonnée : Graphe

  - Procédure de création :

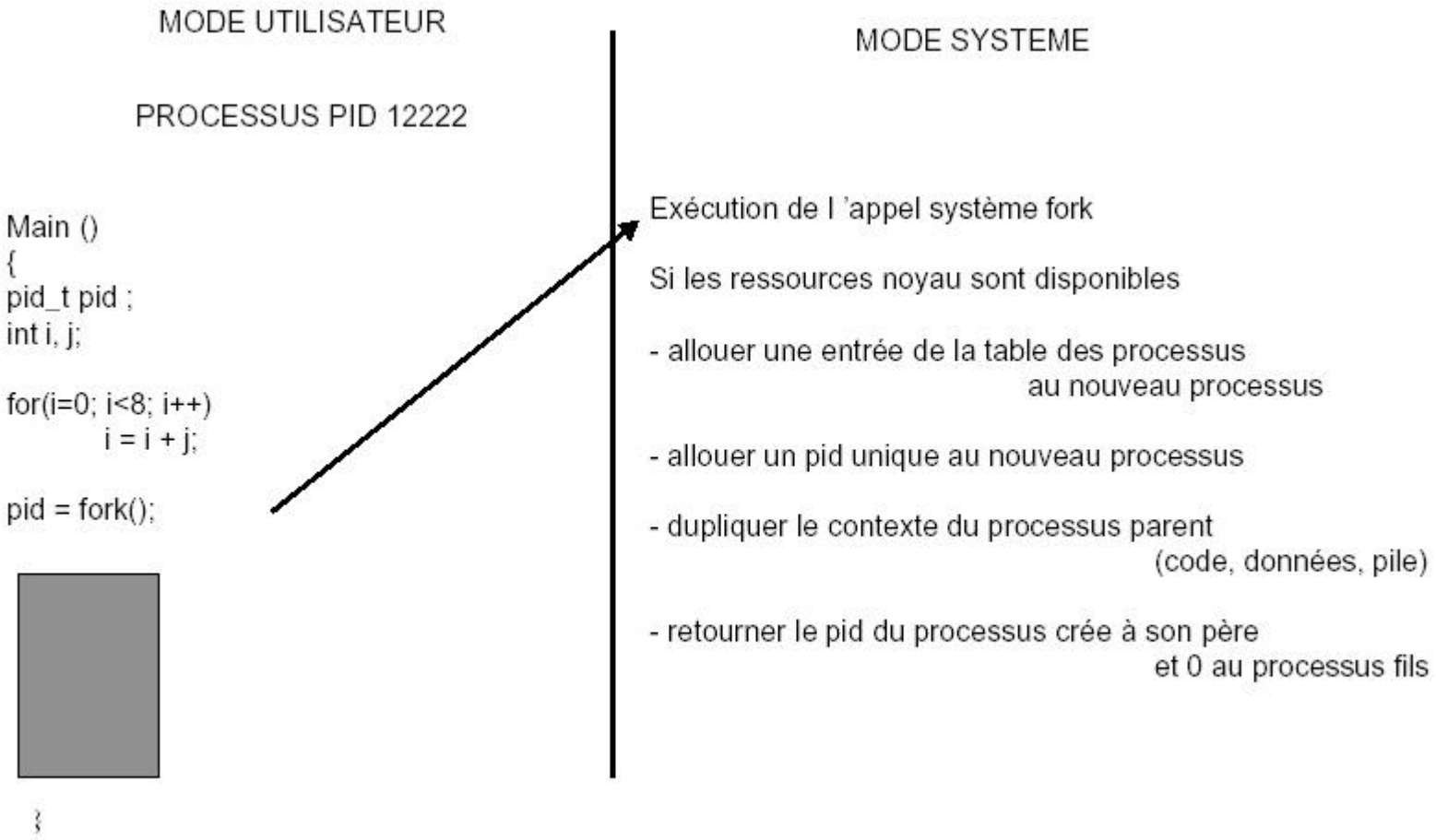
- *Obtenir une entrée de la table des processus (PCB) et un numéro de processus*
- *Initialiser le PCB et l'état du processus*
- *Obtenir un espace mémoire pour le processus*
- *Charger le code et les données du processus en mémoire*
- *Ouvrir les fichiers standards*

Création de processus

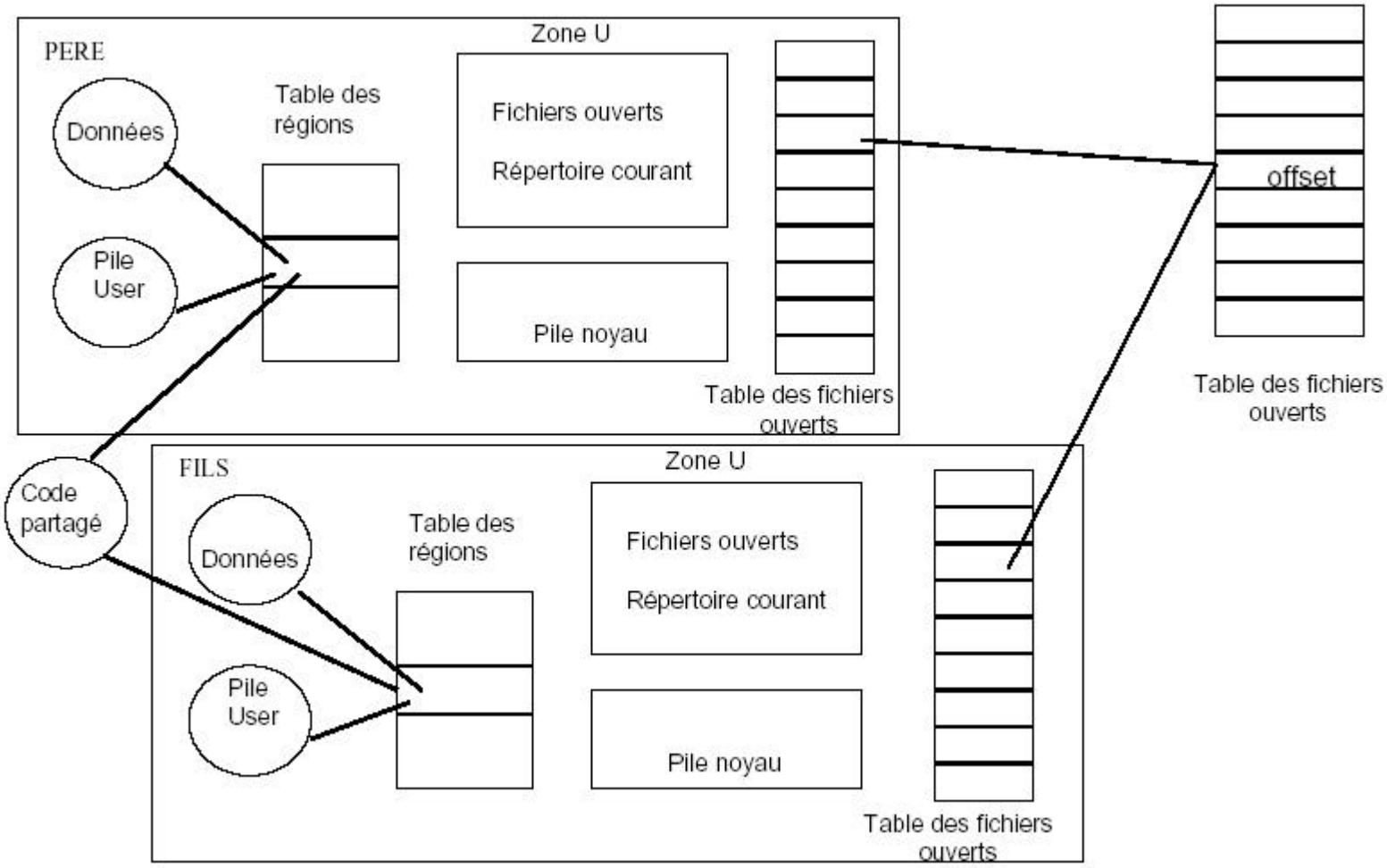
- **La primitive Fork()**
  - Permet la création dynamique d'un nouveau processus qui s'exécute de manière concurrente avec le processus qui l'a créé
  - Tout processus Unix hormis le processus 0 est crée à l 'aide de cette primitive.
  - Le processus créateur crée un processus fils qui est une copie exacte de lui-même (code et données)

Création de processus

- **La primitive Fork()**



Création de processus



Synchronisation Père Fils

- **La primitive Exit()**
  - Permet la terminaison du processus effectuant l'appel avec un code retour valeur
  - Un processus qui se termine passe par un état zombi tant que le père n'a pas pris en compte sa terminaison

Synchronisation Père Fils

PROCESSUS
PID 12222

```
Main ()
{
pid_t pid ;
int i, j;

for(i=0; i<8; i++)
        i = i + j;

pid= fork();

if (pid == 0)
{
      printf(" je suis le fils ");
      exit(); }

else
{
    printf ("je suis le père");
    printf ("pid de mon fils, %d" , pid)
    wait();
 }

}
```

PROCESSUS
PID 12224

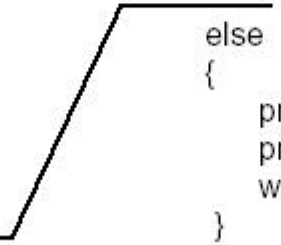
```
Main ()
{
pid_t pid ;
int i, j;

for(i=0; i<8; i++)
        i = i + j;

pid= fork();

if (pid == 0)
{
      printf(" je suis le fils ");
      exit; }
else
{
    printf ("je suis le père");
    printf ("pid de mon fils, %d" , pid);
    wait();
 }

}
```



Terminaison processus

- Lorsqu'un processus se termine (exit), le système démantèle tout son contexte, sauf l'entrée de la table des processus le concernant.

- Le processus père, par un wait(), "récupère" la mort de son fils, cumule les statistiques de celui-ci avec les siennes et détruit l'entrée de la table des processus concernant son fils défunt.

- Le processus fils disparaît complètement.

- La communication entre le fils zombi et le père s'effectue par le biais d'un signal transmis du fils vers le père (signal SIGHLD ou mort du fils)

ATTENTION : mauvaise synchronisation = saturation table des processus = blocage du système

Génétique des processus

- Décès et adoption

- Un processus fils défunt reste zombi jusqu'à ce que son père ait pris connaissance de sa mort

- Un processus fils orphelin, suite au décès de son père (le processus père s'est terminé avant son fils) est toujours adopté par le processus 1 (init).

Destruction de processus

- **Un processus peut terminer de différentes façon**

  - Soit normalement par terminaison de sa dernière instruction
  - Soit par exécution d'un instruction autodestructrice
  - Soit détruit par un autre processus

  - La destruction du processus
    - libèrent les ressources qu'ils lui ont été affectées
    - disparaît de la table des processus

La synchronisation des processus

Partie 1e

La synchronisation

- Problème d’accès concurrent vis-à-vis des ressources partagées
- Problème de la synchronisation des actions sur ces ressources en cas de modification
- La partie de programme dans laquelle se font des accès à une ressource partagée s'appelle **une *section critique*.** (Ex Base de Donnée)
- L'accès à cette section critique devra se faire en *exclusion mutuelle* (ce qui implique de rendre indivisible la séquence d'instructions composant la section critique).