Systèmes d'Exploitation - ENSIN6U3

Gestion de processus

Leonardo Brenner [1] Jean-Luc Massat [2]

[1] Leonardo.Brenner@univ-amu.fr

[2] Jean-Luc.Massat@univ-amu.fr

Aix-Marseille Université

Faculté des Sciences

Table de matière

1. Définition d'un processus
   - Représentation d'un processus
   - État d'un processus
2. Création de processus lourd
   - Cas de Unix
   - Cas de Windows
3. Création de processus léger

Table de matière

1. Définition d'un processus
   - Représentation d'un processus
   - État d'un processus
2. Création de processus lourd
   - Cas de Unix
   - Cas de Windows
3. Création de processus léger

Processus : définition, identification, filiation

Définition : processus

Un processus est un programme en cours d'exécution.

Composition d'un processus

Un processus comporte :

- un espace d'adressage ;
- le bloc du contrôle du processus, décomposé en :
  - entrée dans la table des processus ;
  - une zone u, allouée dynamiquement à la création du processus.

Représentation d'un processus

Données sur un processus

Pour chaque processus le systèmes maintient les données suivantes :

- un identifiant (PID),
- des informations diverses (priorités, filiations, propriétaires, ...),
- un état opérationnel,
- un contexte d'exécution,
- des statistiques (temps de CPU, # d'E/S, # de défauts de pages, ...).

Ces informations sont rangées dans un PCB (*Process Control Block*).

Identifiants des processus

Identification

- A chaque processus est associé un identifiant :
  ↪ Cet identifiant est appelé PID (pour Process Identifier)
- Le PID est attribué par le système au moment de l'exécution
- Tous les processus ont un PID unique
- Le premier processus : init possède le PID 1

Filiation

- Tout processus est créé à partir d'un autre processus :
  ↪ Sous Unix, le premier processus init
- Tous les processus sont créés à partir de init
- Un processus A créé par un processus B est appelé fils de B et B est le père de A
- Le PPID d'un processus est le PID de son père

Propriétés et accès aux fichiers

La priorité

- Utilisée lors de l'exécution du processus
- Un processus sera exécuté avant (ou plus souvent) que les autres processus moins prioritaire
- Généralement, la priorité évolue dans le temps

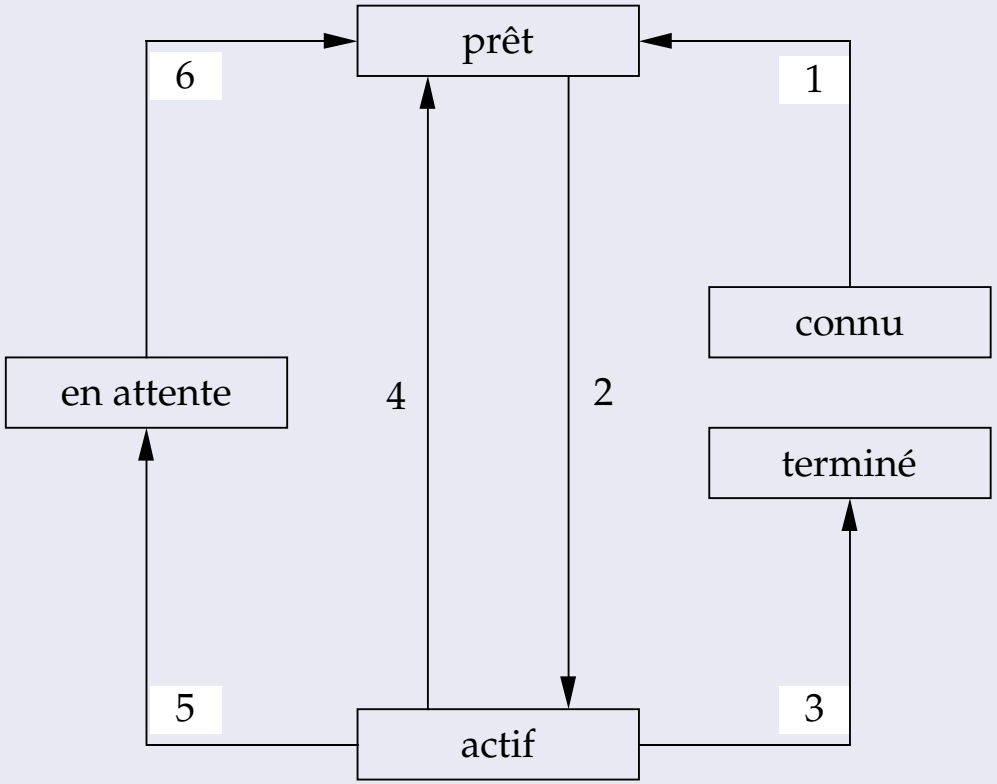
Accès aux fichiers

- A chaque processus est attribué des identifiants pour les accès aux fichiers
- En fonction des identifiants, le processus pourra ou non accéder à certains fichiers
- Plusieurs identifiants :
  - GID : l'identifiant de groupe
  - UID : l'identifiant de l'utilisateur

État d'un processus

États opérationnels

Chaque processus est dans l'un des **états opérationnels** suivants :



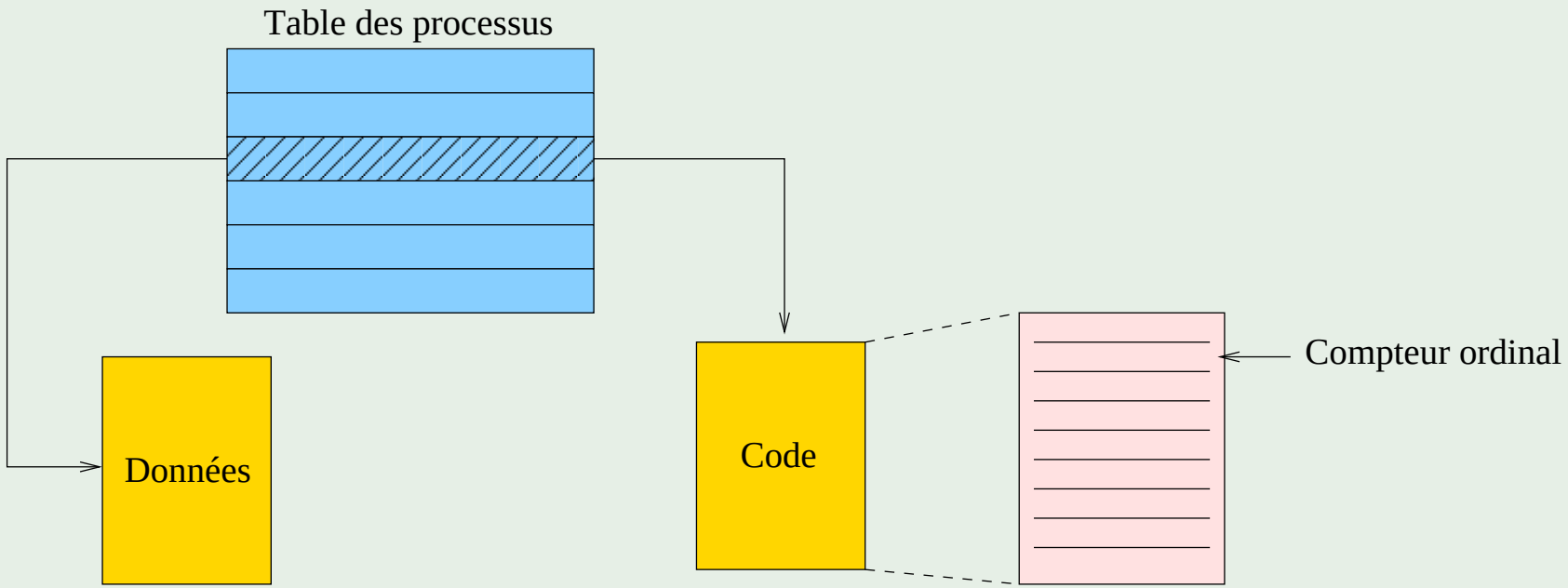
1. initialisation
2. exécution
3. achèvement
4. préemption
5. attente
6. signal

La table des processus

Description

- L'ensemble des processus sont gérés dans une table :
  ↪ Une entrée par processus
- Données sur les processus : identifiant du processus, priorité, compteur ordinal, pointeurs vers les segments mémoire, . . .

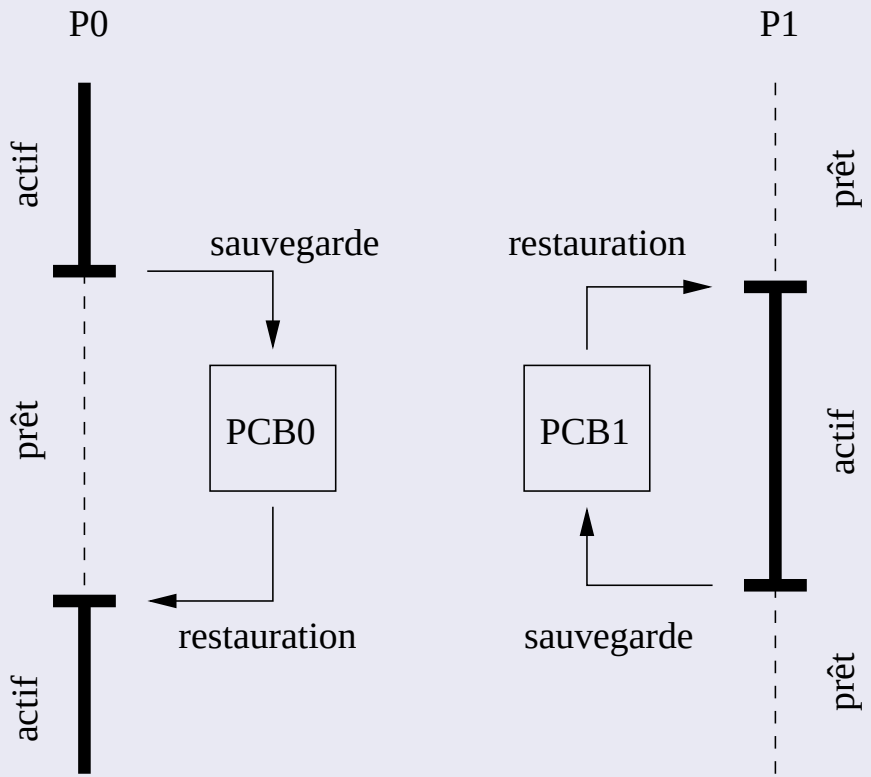
Représentation mémoire



Rôle de PCB

Commutation de contexte

Le rôle du PCB dans la commutation de contexte entre deux processus :



Contexte d'un processus

Quelques données nécessaires à la commutation de contexte

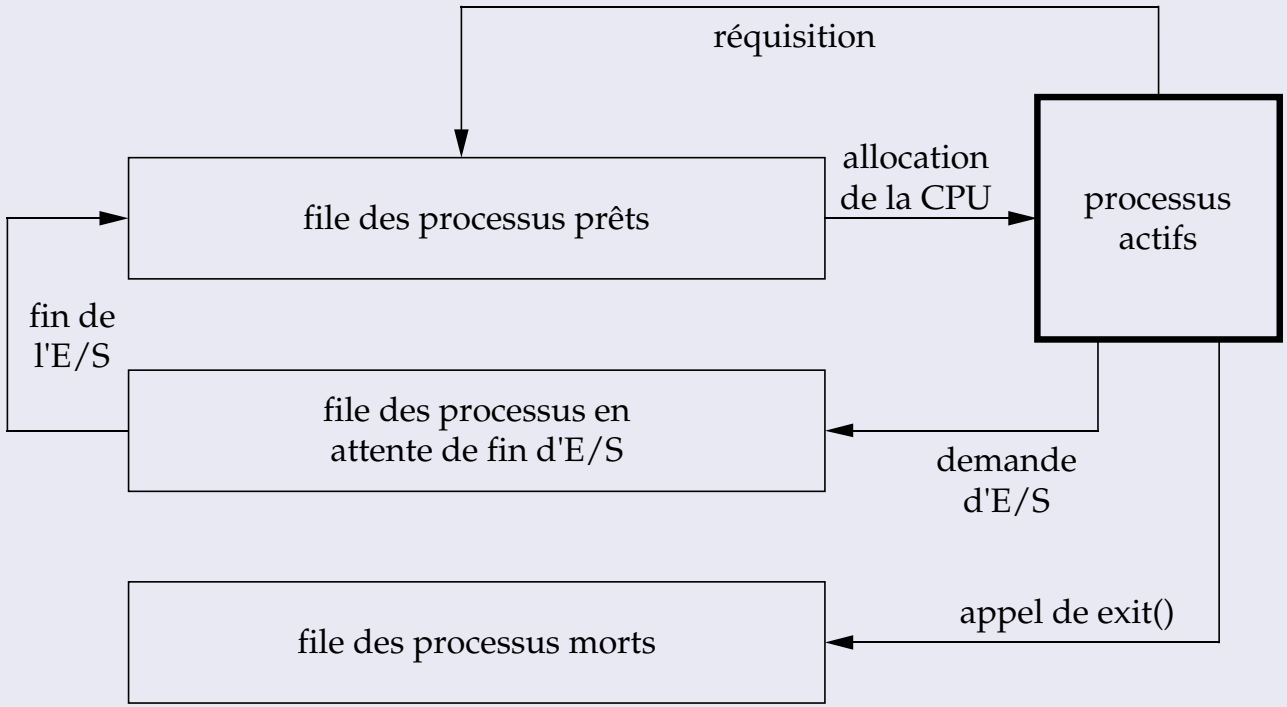
- mot d'état, contexte de l'unité centrale :
  - accumulateur,
  - registre d'instruction et compteur d'instruction,
  - registres d'état du processeur,
  - registres d'états du processus.
- état du processus,
- variables globales statiques dynamiques,
- entrée dans la table du processus,
- zone u,
- piles utilisateur et système,
- zones de codes et de données.

Files et états des processus

Files des processus

Le système maintient un ensemble de files dans lesquelles il range les PCB. On trouve par exemple un file

- des processus prêts,
- des processus en attente (d'une ressource, de la fin d'une E/S, ...),
- etc.



Gestion des processus

Création de processus

- chaque processus peut créer des processus (ses fils)
- il est possible de *figer, tuer* des processus

Communication entre processus

Le système gère la communication entre les processus existants (signal, messages, ...)

Synchronisation de processus

La gestion de synchronisation entre les processus gère la dépendance entre les processus.

Gestion des processus

Types de processus

Le S.E. gère deux types de processus :

- Lourd : crées par `fork`, nouveau PID, nouvelle zone mémoire,
- Léger : crées par « threads », même PID, même zone mémoire (mémoire partagée)

Table de matière

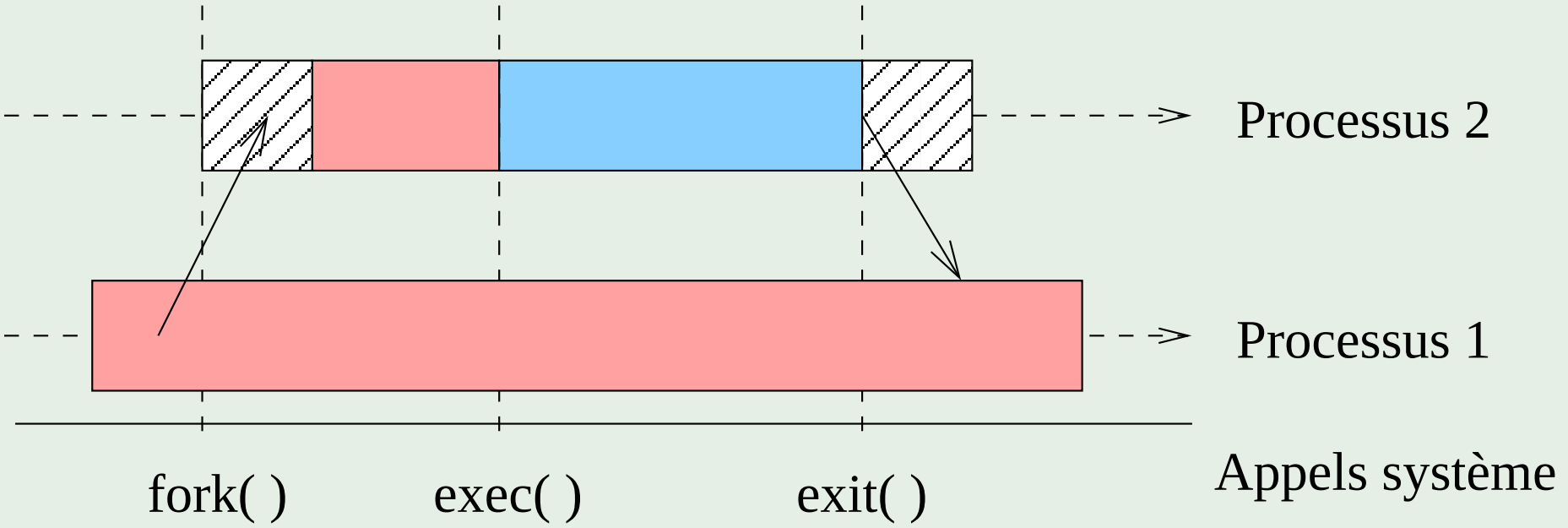
1. Définition d'un processus
   - Représentation d'un processus
   - État d'un processus
2. Création de processus lourd
   - Cas de Unix
   - Cas de Windows
3. Création de processus léger

Création d'un processus : cas de Unix (1/2)

Sous Unix

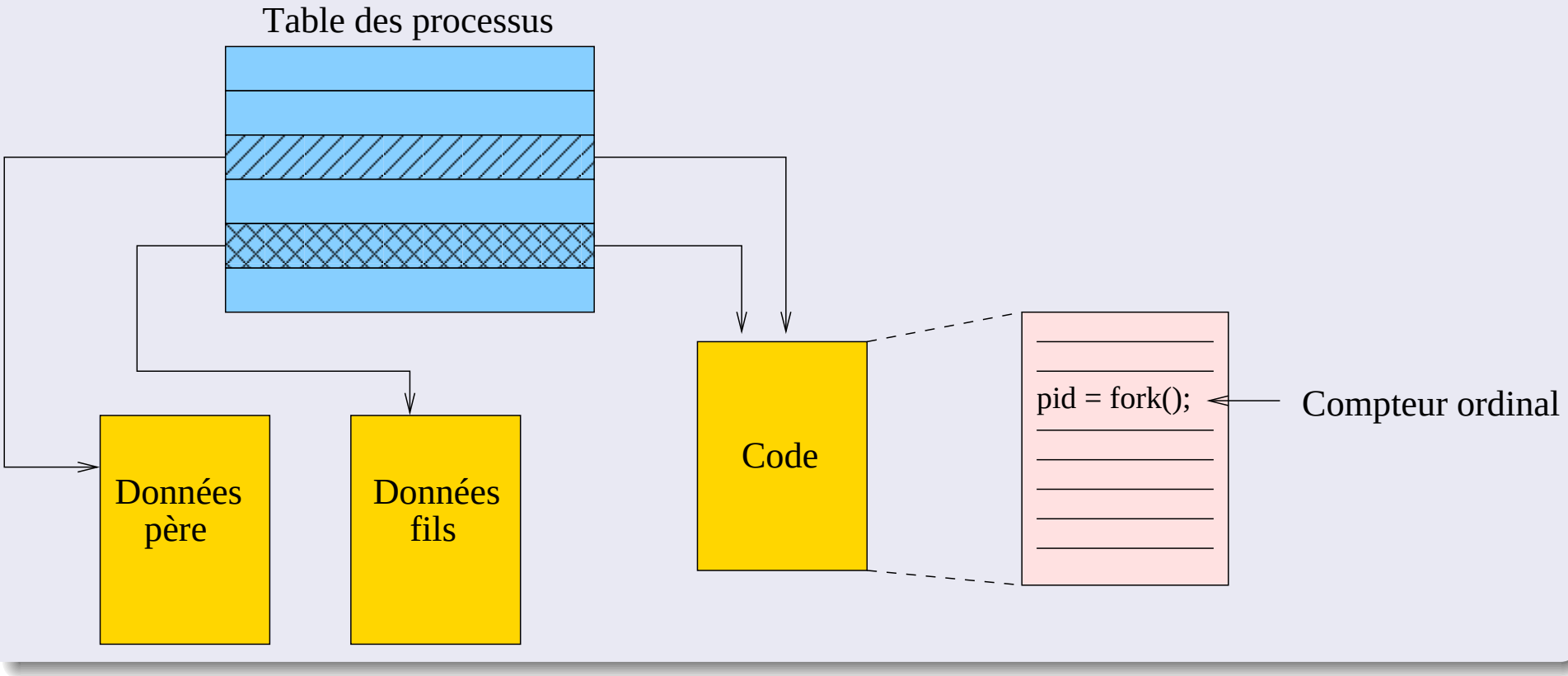
- Appel système `fork()`
- Le processus appelant (le père) est dupliqué (même image mémoire)
- La copie est remplacé par le fils (via exec)

Illustration



Création d'un processus : cas de Unix (2/2)

Création d'un processus avec fork()



Création d'un processus : cas de Windows

Sous Windows

- Appel à `CreateProcess()` (Win32)
- Le processus fils est créé et remplacé directement par le processus

Illustration

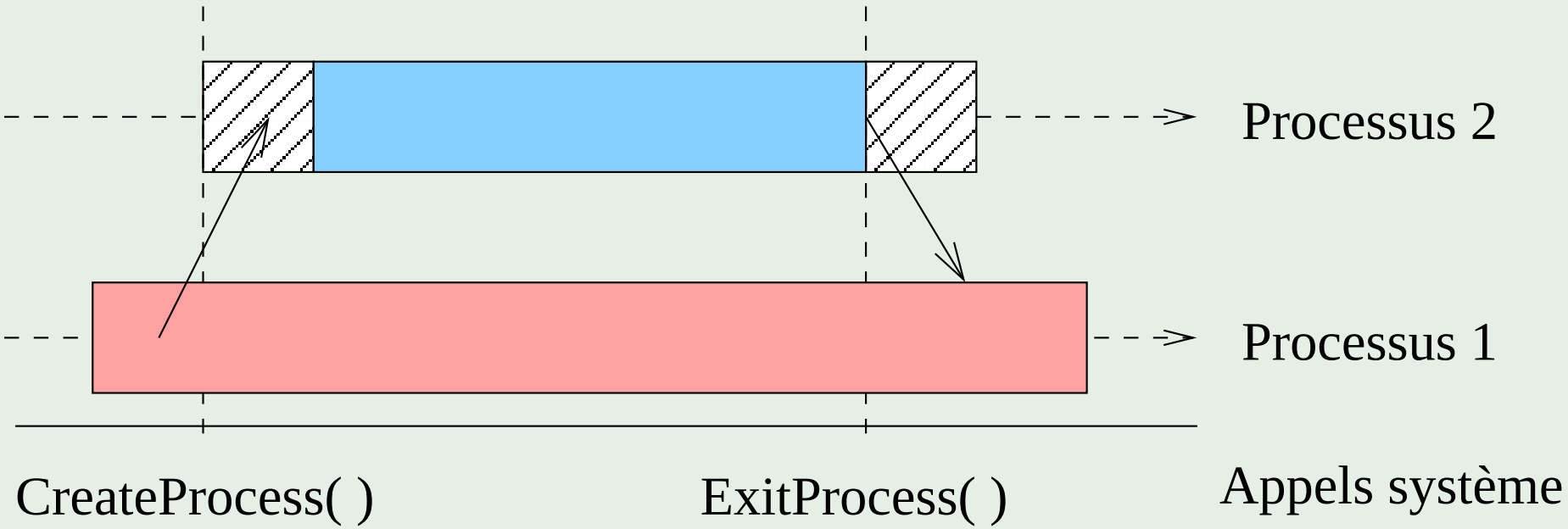


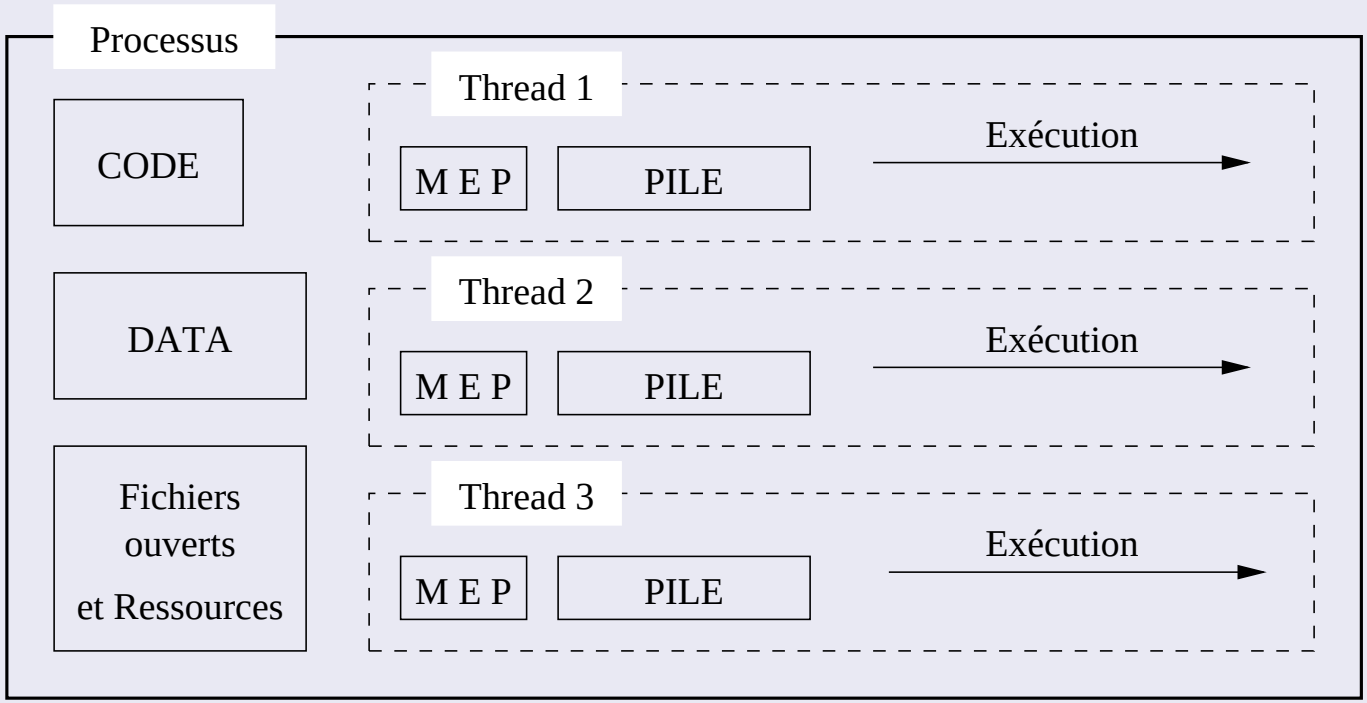
Table de matière

1. Définition d'un processus
   - Représentation d'un processus
   - État d'un processus
2. Création de processus lourd
   - Cas de Unix
   - Cas de Windows
3. Création de processus léger

Les threads : processus de poids léger

Threads

Un **thread** (fil) est un programme en cours d'exécution qui partage son code et ses **données**.

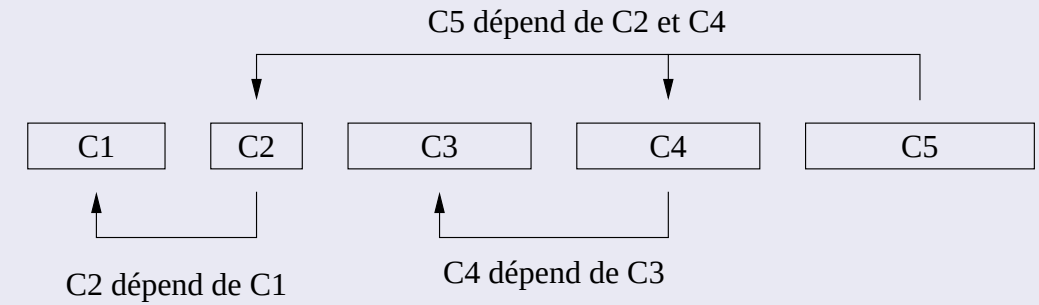


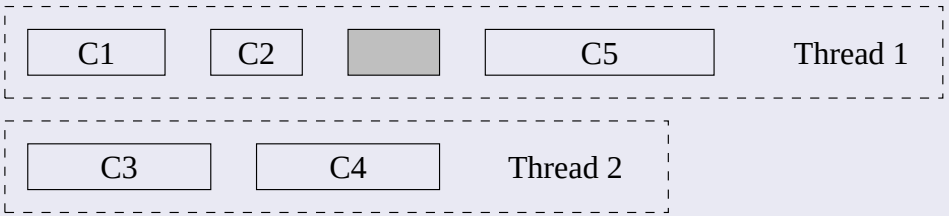
- Chaque thread a une pile d'exécution autonome.
- Un processus est composé de *threads*

Les threads : processus de poids léger

Utilité des threads

Étude des dépendances et découpage :





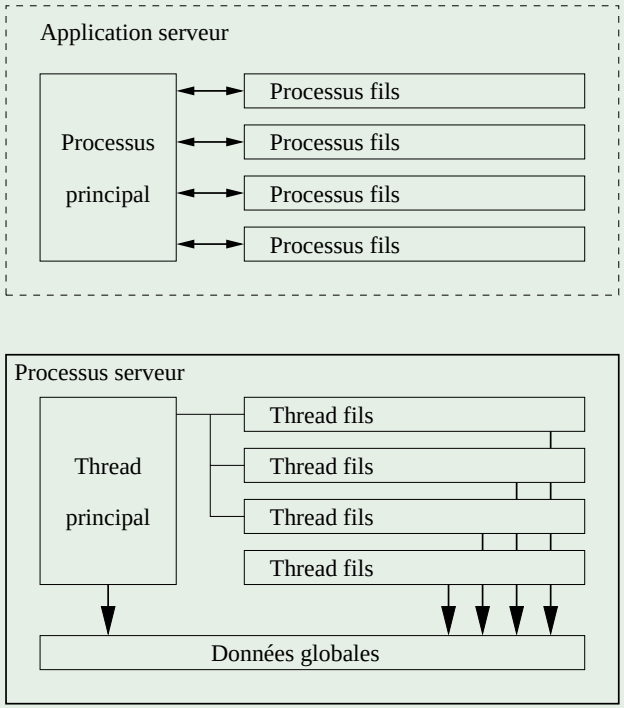
Avantages :

- récupération des temps d'E/S,
- exploitation des machines multi-processeurs,
- coopération entre threads.

Les threads : processus de poids léger

Utilité des threads : application serveurs

Organisation logicielle d'une application serveur :



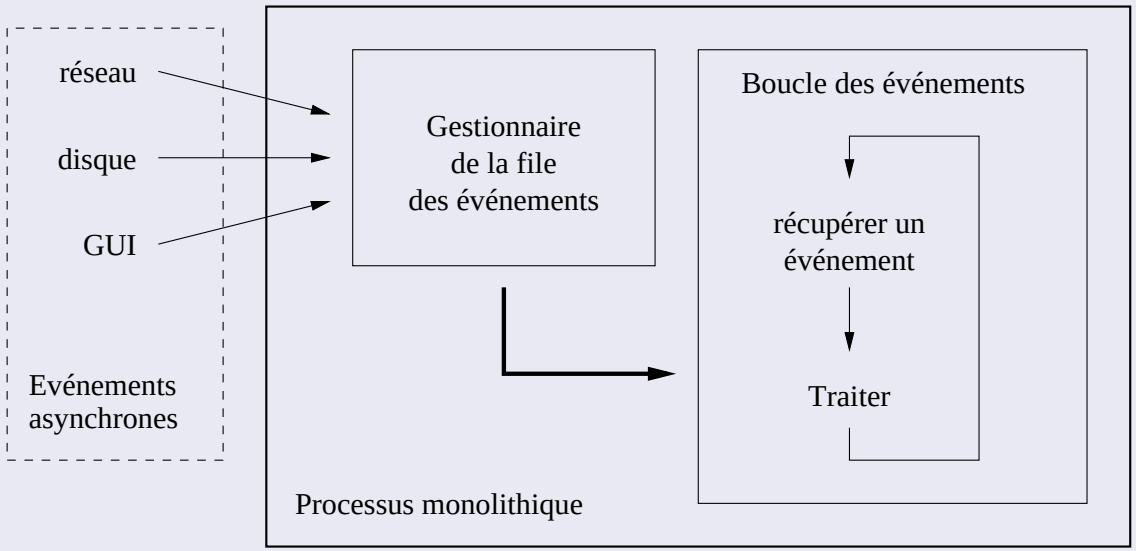
La **commutation** et la **communication** entre threads est une opération plus simple.

Les threads : processus de poids léger

Les processus monolithiques

Il existe un conflit entre :

- entrées / sorties synchrones (bloquantes),
- interface homme/machines (IHM).



Solution d'attente active :

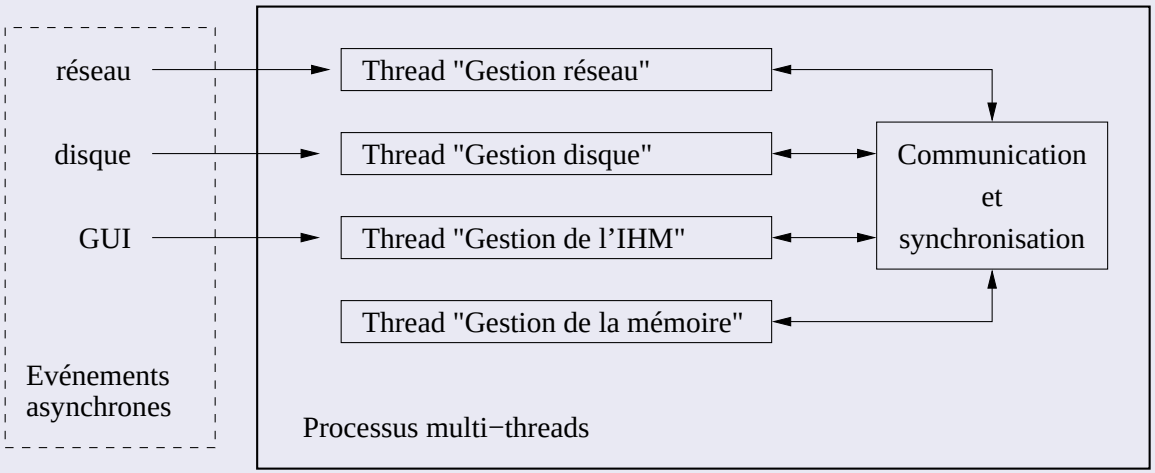
- E/S asynchrones (non-bloquantes),
- structure avec boucle d'événements.

Les threads : processus de poids léger

Les processus multi-threads

Il existe une autre solution basée sur :

- le découpage en plusieurs threads,
- une utilisation des E/S synchrones,
- un module de communication.



Avantages :

- plus grande simplicité du code,
- indépendance entre les modules.

Les threads : processus de poids léger

Implantation des « threads »

Implantation au niveau du S.E. :

+ le S.E. connaît et ordonnance les threads,
+ la répartition de la CPU est bonne,
- les structures du S.E. sont alourdies.

Implantation au niveau utilisateur (java) :

+ une librairie se charge de la gestion des threads (création, destruction, etc.),
+ la commutation entre threads d'un même processus est plus rapide,
- la répartition de la CPU n'est pas équitable,
- la mise « en attente »d'un processus entraîne le blocage de tous ses threads.