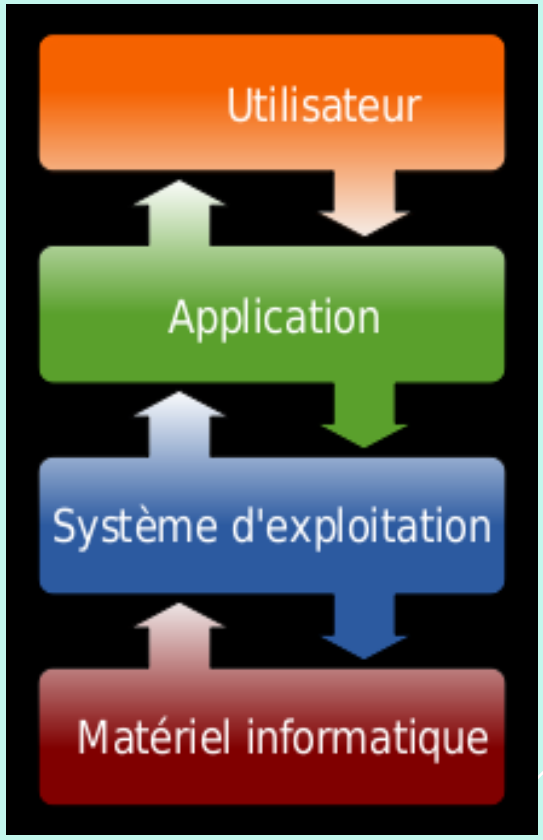
Chapitre 1 : Généralités

Plan

- Rappels sur la notion de SE.
- Notions de programme, processus, thread et ressource partagée.

1-Rappels sur la notion de SE

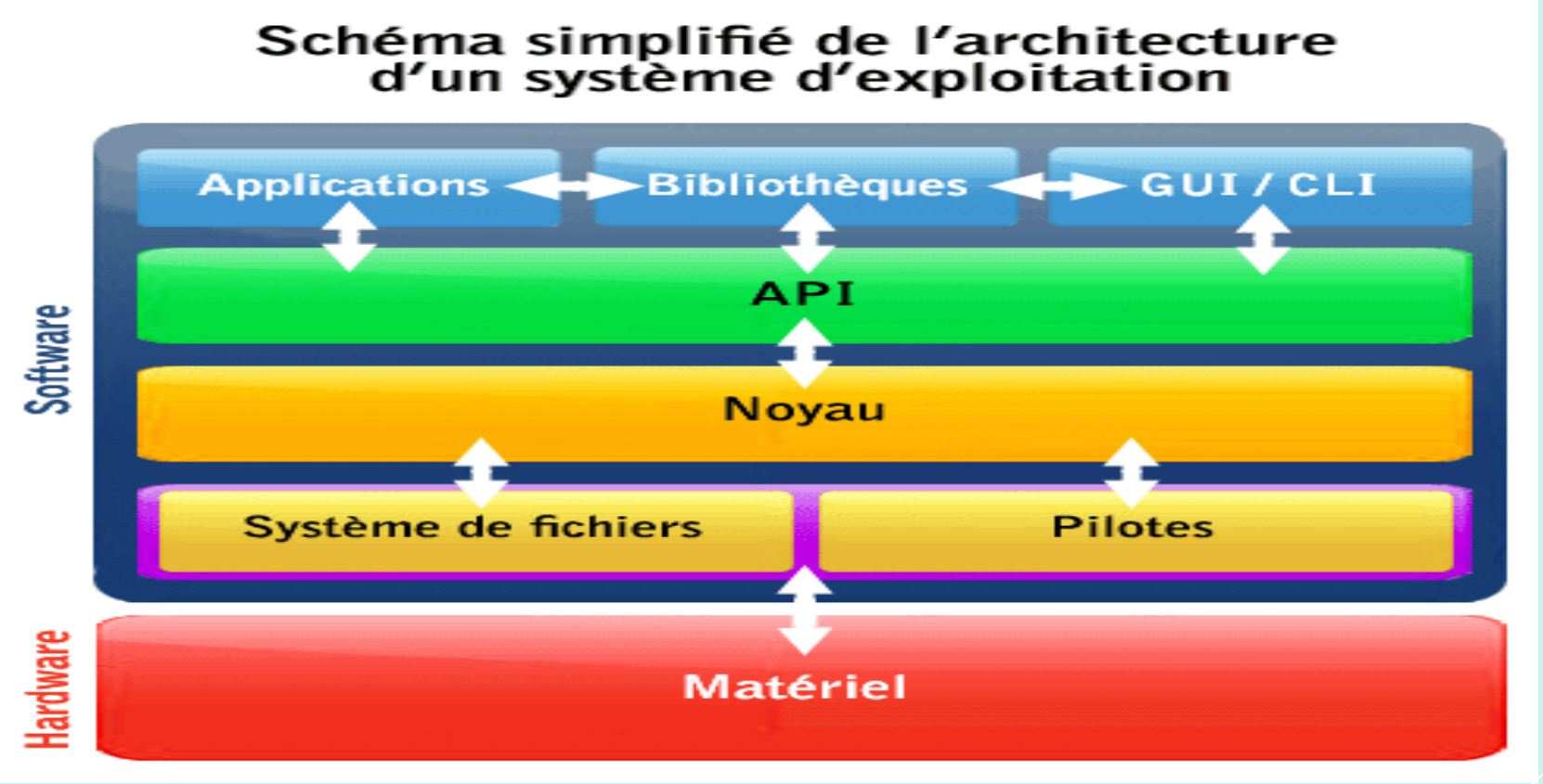
Définition : Un Système d'Exploitation peut être défini comme un logiciel qui , dans un appareil électronique, pilote les dispositifs matériels et reçoit des instructions de l'utilisateur ou d'autres logiciels (ou applications).



Utilisent des langages évolués (texte, chiffres, image, son, dessin, …)

Utilisent un langage binaire (0, 1)

- Le S.E est composé d'un ensemble de logiciels permettant de gérer les interactions avec le matériel:



Les rôles du S.E

1.Gestion des processeurs. Gérer l'allocation des processeurs entre les différents programmes grâce à un algorithme d'ordonnancement.

2.Gestion des processus. Gérer l'exécution des processus en leur affectant les ressources nécessaires à leur bon fonctionnement.

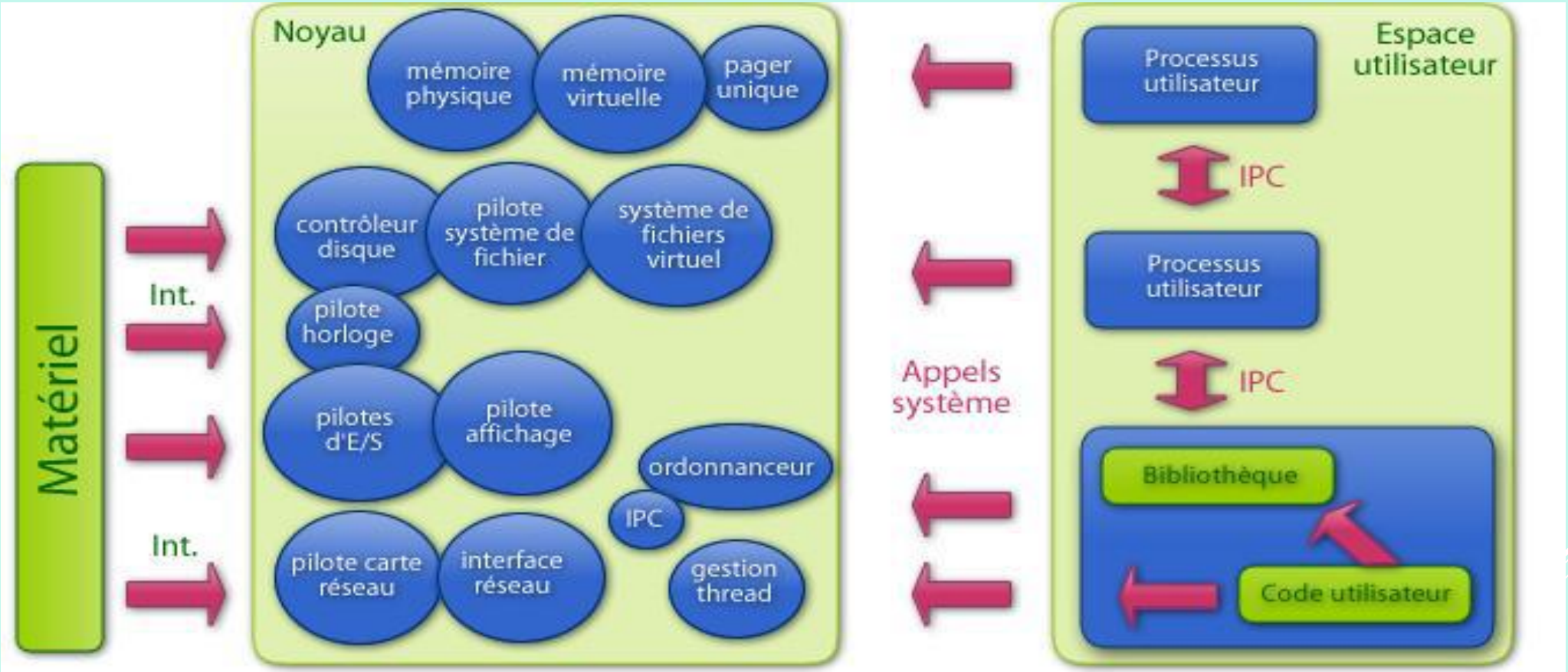
3.Gestion des mémoires. Gérer l'espace mémoire alloué à chaque processus.

4.Gestion des périphériques. Contrôler l'accès des processus aux ressources matérielles par l'intermédiaire des pilotes.

5.Gestion des fichiers. Gérer la lecture et l'écriture dans le système de fichiers et les droits d'accès aux fichiers par les utilisateurs et les applications.

6.Etc.

Architecture du S.E



Générations des S.E

1.Mono utilisateur/Mono tâche

- C'est le cas le plus simple : Un seul utilisateur à la fois et une seule tâche à la fois
- Les systèmes d'exploitation des premiers micro-ordinateurs ne dépassaient pas ce niveau de complexité.

2.Multitâches

- Partager le temps du processeur entre plusieurs programmes (tâches) → impression de réalisation simultanée.
- Le passage de l'exécution d'un programme à un autre peut être initié:
  - ❑ Par les programmes eux-mêmes (coopératif)
  - ❑ Par le S.E (préemptif)

3.Multiutilisateurs (temps partagé)

- Plusieurs utilisateurs peuvent utiliser simultanément une même machine pour des applications similaires ou différentes.
- Chaque utilisateur à l'impression d'être le seul à utiliser l'ordinateur.

4.Multiprocesseurs

- Un processeur central (maître) peut coordonner une série de tâches sur plusieurs autres processeurs (esclaves).
- Organisation à l'extérieur d'une série de tâches sur plusieurs processeurs (systèmes répartis)

5.Temps réels

- Servent pour le pilotage et le contrôle des déroulements externes (exemple centrale électrique)
- Doivent garantir des temps de réactions données pour des signaux extérieurs urgents.

6.Distribués

- Doivent permettre l'exécution d'un seul programme sur plusieurs machines.
- Distribuer les tâches et les remettre ensemble
- Pour gros calculs, exemple inversion de grandes matrices

Les S.E dédiés aux ordinateurs (bureau ou portable)

Système		Codage	Utilisateur	Tâche	Processeur
MS-DOS (Gates, 1981)		16 bits	Mono	Mono	Mono
Windows 95/98/ME/	Microsoft (depuis 1991)	32 bits		Multi (coopératif)	
Windows NT/2000			Multi	Multi (préemptif)	
Windows XP/Vista/7/8/ 10		32/64 bits			Multi
Unix (Ritchie & Thompson,1969)		32/64 bits			
Linux (Torvalds & Stallman, 1992)		32/64 bits			
MacOS (Jobs & Wozniak, 1984)		32 bits			

2-Notions de programme, processus, thread et ressource partagée

Processus et Programme

Processus

Définition : c'est un programme en cours d'exécution auquel on associe :

⇒ ***Un Contexte du processeur (CPU) : l'ensemble des registres (CO, SW, SP, Rx, …)***

⇒ ***Un Contexte de la Mémoire Centrale : segments de code, segments de données, …***

Un processus est dit **séquentiel** si l'exécution de l'ensemble de ses instructions s'établie dans un ordre strict et bien déterminé.

- Programme : Entité statique
- Processus ; Entité dynamique

Les états d’un processus

Dans les systèmes, un programme ne quitte pas l’unité centrale avant de terminer son exécution. Pendant cette période, il dispose de toutes les ressources de la machine. Par contre, ce n’est pas le cas dans les systèmes multiprogrammés et temps partagé, un processus peut se trouver dans l’un des états suivants :

1- **Elu :** (en cours d’exécution) : si le processus est en cours d'exécution

2- **Bloqué** : attente qu’un événement se produit ou bien ressource pour pouvoir continuer

3- **Prêt :** si le processus dispose de toutes les ressources nécessaires à son exécution à l'exception du processeur

Sémantique des Transitions

(1) : Allocation du processeur au processus sélectionné

(2) : Réquisition du processeur après expiration de la tranche du temps par exemple soit parce qu'un processus de plus haute priorité réquisitionne le processeur.

(3) : Blocage du processus élu dans l'attente d'un événement (E/S ou autre)

(4) : Réveil du processus bloqué après disponibilité de l'événement bloquant (Fin E/S, etc…)

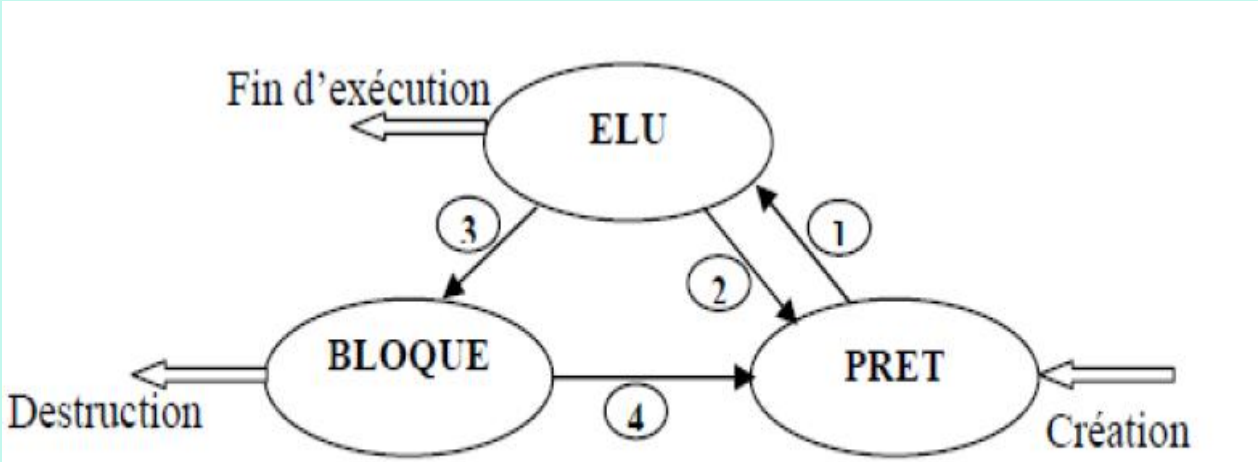


Figure : Diagramme de transition d'un processus

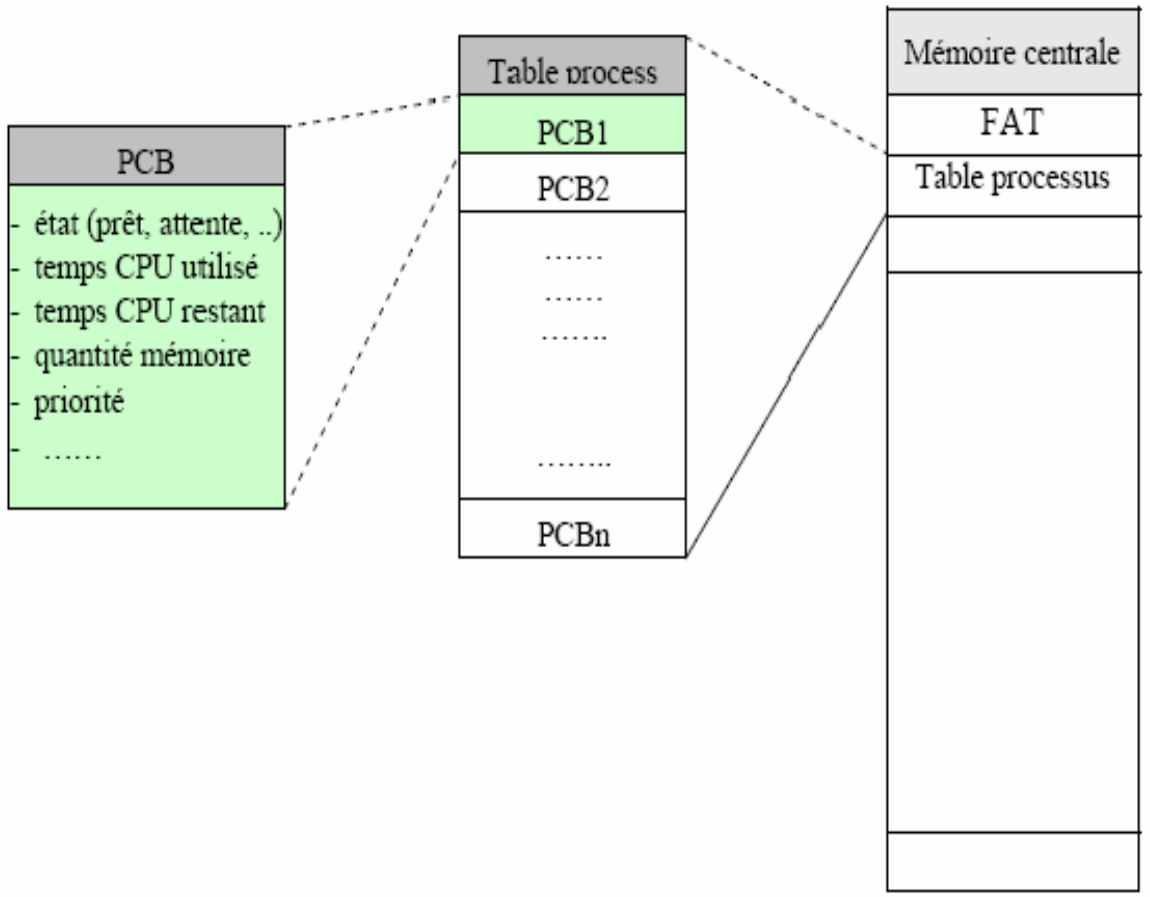
Représentation interne des processus

→Au moment du chargement d'un programme exécutable, le SE lui crée une structure représentant le processus associé. Cette structure devrait contenir toutes les informations nécessaires permettant l'évolution dynamique du processus au sein du SE, entre autre son contexte, son état, …cette structure est appelée le Bloc de Contrôle du Processus (PCB).

→Pour manipuler tous les processus, le SE détient d'une **table de processus** dont chaque entrée contient un pointeur vers un PCB d'un processus.

PCB
ID process, IDpropriétaire
état du process
Contexte processeur
Contexte mémoire
Informations ressources (fichiers ouverts, E/S,..
Infos : Pages, Segments,
infos sur les fils
…

Les BCP sont rangés dans une table (table des processus) qui se trouve dans l'espace mémoire du système



Bloc de contrôle de processus (PCB)

Notion de tâche (thread)

Lorsqu'un processus présente un code un petit peu long ou compliqué, il peut être décortiqué en un ensemble d'unités de traitements élémentaires ayant une cohérence logique dont la fonction est bien déterminée. Cette unité est appelée une **tâche.**

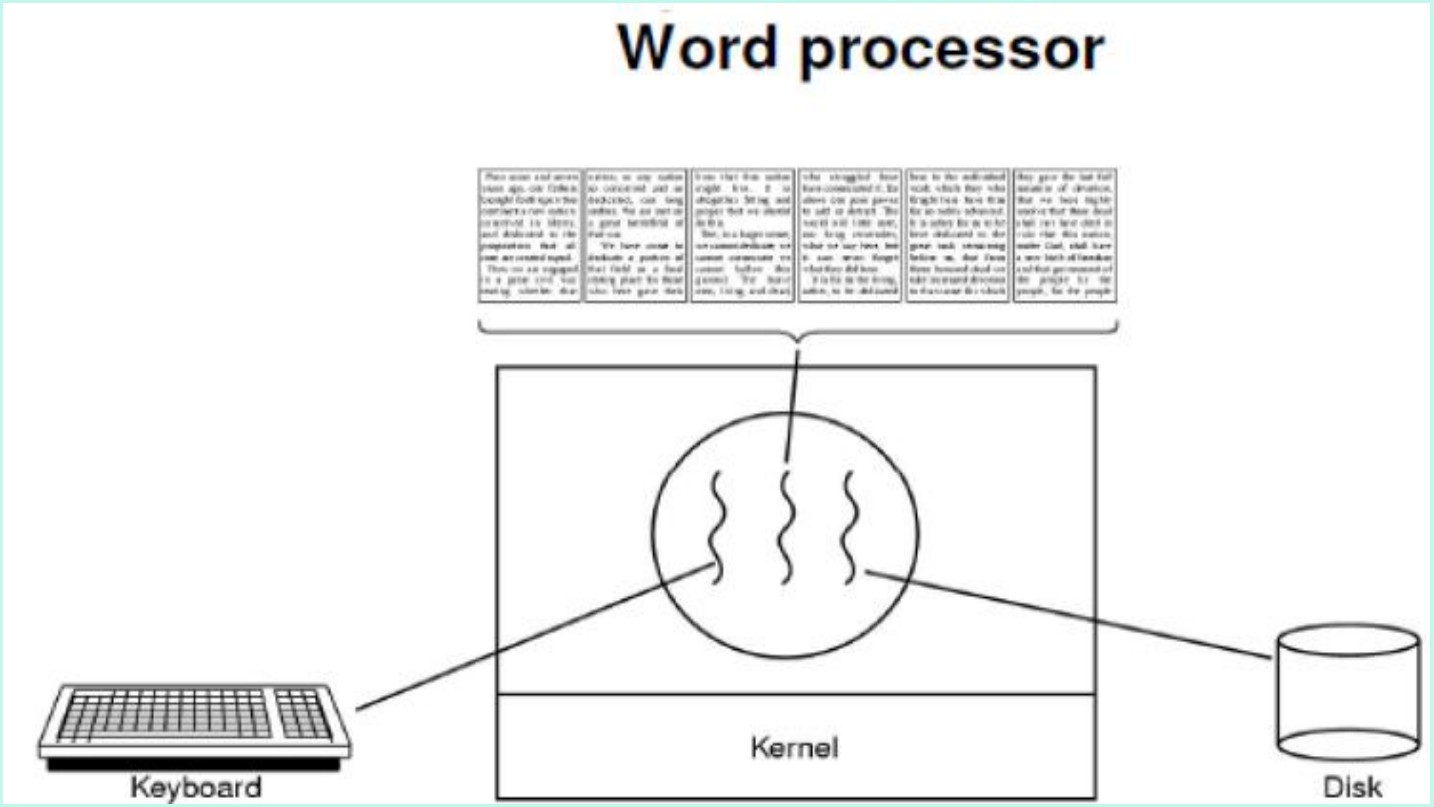
- Un thread est une subdivision d`un processus
- Les différents threads d'un processus partagent l'espace adressable et les ressources d'un processus
- Les threads sont des processus légers exécutés «à l'intérieur» d'un processus
- L'exécution des threads est concurrente
- il existe toujours au moins un thread : le thread principal
- La durée de vie d'un thread ne peut pas dépasser celle du processus qui l'a crée

Multi-threading

Une application multi-thread est un logiciel qui dès sa conception a été partagé en différentes sous-applications appelées threads.

A l'inverse de ce qui passe dans le cadre du multi-tâche où chaque processus dispose d'une partie distincte de la mémoire pour s'exécuter indépendamment des autres. Les threads issus de même application partagent un même espace mémoire.

Exemple :



- Un thread pour interagir avec l'utilisateur
- Un thread pour reformater en arrière plan
- Un thread pour sauvegarder périodiquement le document
- Un thread la vérification automatique de l'orthographe et de la grammaire

- Ainsi un processus séquentiel de N tâches P peut être écrit comme suit :

  P = T1 T2 T3 …. TN

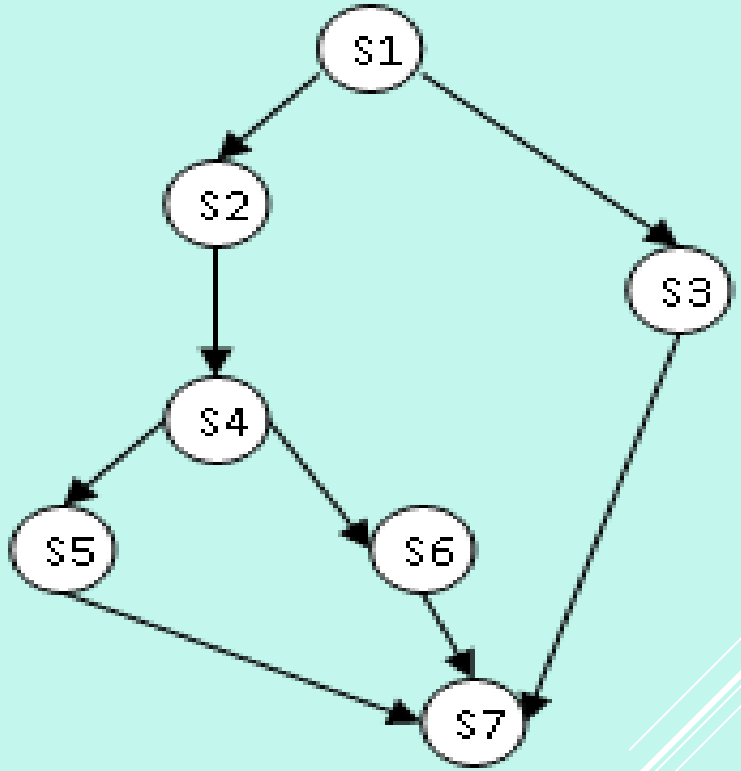
  → Certaines tâches doivent s'exécuter en **série** d'autres peuvent s'exécuter en **parallèle**.

  → On peut représenter l'ensemble de tâches constituant un processus par un **graphe de précédence.**

- Graphe de précédence : C'est un graphe dirigé acyclique dont les nœuds représentent les instructions et les arcs représentent les dépendances entre les nœuds incidents : un arc d'un nœud S_i vers un nœud S_j signifie que le nœud S_j ne peut s'exécuter avant la fin du nœud S_i.

S2 et S3 peuvent s'exécuter après la fin de S1. S4 peut s'exécuter après la fin de S2.

Comment peut-on détecter les instructions parallèles ?



- **Conditions de Bernstein**

Soit I une instruction d'un programme, on dénote par :

R(I) : l'ensemble des variables de l'instruction I qui ne changent pas par l'exécution de l'instruction I.

W(I) : l'ensemble des variables de l'instruction I qui sont mises à jour par l'exécution de l'instruction I.

Exemple :

Soit le programme suivant qui lit des informations de deux périphériques, puis écrit l'information dans un troisième périphérique :

```
Début
   Lire(a) ;
   Lire(b) ;
   C=a+b ;
   Ecrire(c) ;
Fin
```

Supposons que l'on désire exécuter ces instructions simultanément. Il est clair que l'instruction c=a+b ; ne peut s'exécuter avant la fin des deux instructions de lecture. De même, l'instruction d'écriture écrire(c) ; ne peut s'exécuter avant l'instruction c=a+b ;
On remarque qu'il y a des dépendances entre les instructions : contraintes de précédence.

Nous appliquons les notations ci-dessus :

R(lire(a))={} **R(lire(b))={}**

W(lire(a))= {a} **W(lire(b))= {b}**

R(c=a+b)={a,b} **R(écrire(c))={c}**

W(c=a+b)= {c} **W(écrire(c))= {}**

On dira que deux instructions I1 e I2 peuvent s'exécuter en parallèle si les conditions suivantes sont satisfaites (conditions de Bernstein) :

a) $R(I1) \prod W(I2) = \emptyset$

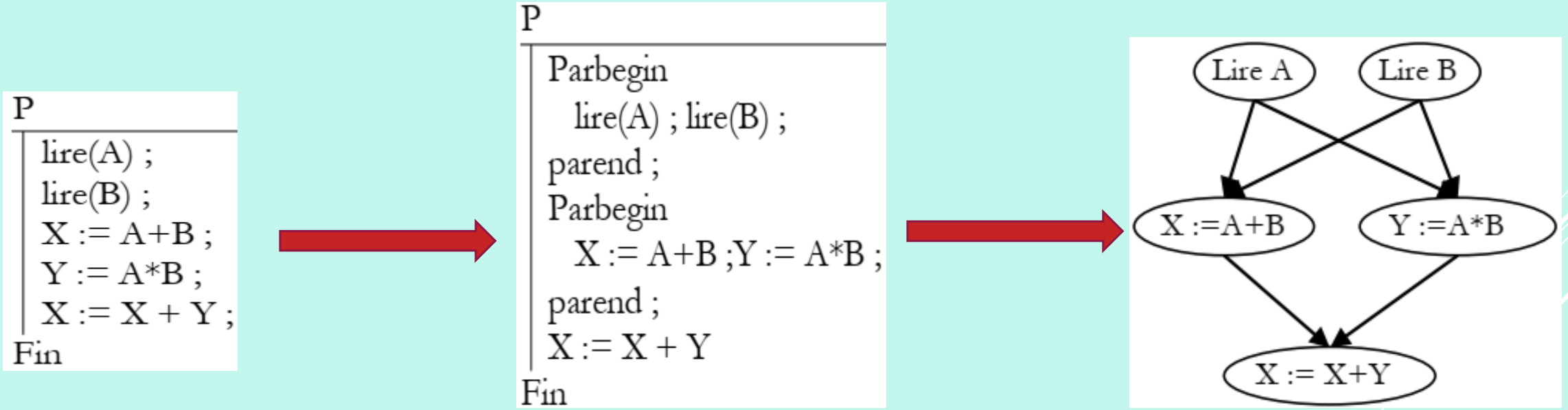
b) $W(I1) \prod R(I2) = \emptyset$

c) $W(I1) \prod W(I2) = \emptyset$

L'instruction lire(b) ne peut s'exécuter en parallèle avec c=a+b car :

$W(\text{lire}(b)) \prod R(c=a+b) = \{b\}$

- On résolvant un problème, on peut transformer un processus séquentiel (une série de tâches séquentielles) en un système équivalent où certaines tâches peuvent s'exécuter en parallèle. Ainsi on augmente le degré de multiprogrammation



Processus séquentiel

Les Ressources

Définitions

– Une ressource désigne toute entité dont a besoin un processus pour s'exécuter.

- Ressource matérielle (processeur, périphérique)
- Ressource logicielle (fichier, variable).

– Une ressource est caractérisée :

- par un état : libre / occupée
- par son nombre de points d'accès (nombre de processus pouvant l'utiliser en même temps)

– ressource critique à un seul point d'accès

Un processus utilise une ressource suivant l'algorithme suivant :

```
Demander( type_ressource, qté_demandée) ;
        <Utiliser(ressource) >;
Libérer(type_ressource, qté_allouée) ;
```

→Selon le degré de partageabilité d'une ressource, on distingue des ressources **N_ partageable** (mémoire, lecture disque, …) et d'autres **critiques** inpartageable (le processeur, imprimante, fichier en écriture, variable globale, ...)

→On distingue aussi des ressources **banalisée**, existant en plusieurs copies identiques pouvant être utilisées indifféremment les unes des autres (tampons (buffers) mémoires, dérouleurs de bande, imprimantes (si identiques)).