Cours 04: La Gestion des Processus

Plan du cours

Notion de Processus

Implémentation de processus

Contexte & États des processus

Les opérations sur les processus

Les thread et le parallélisme

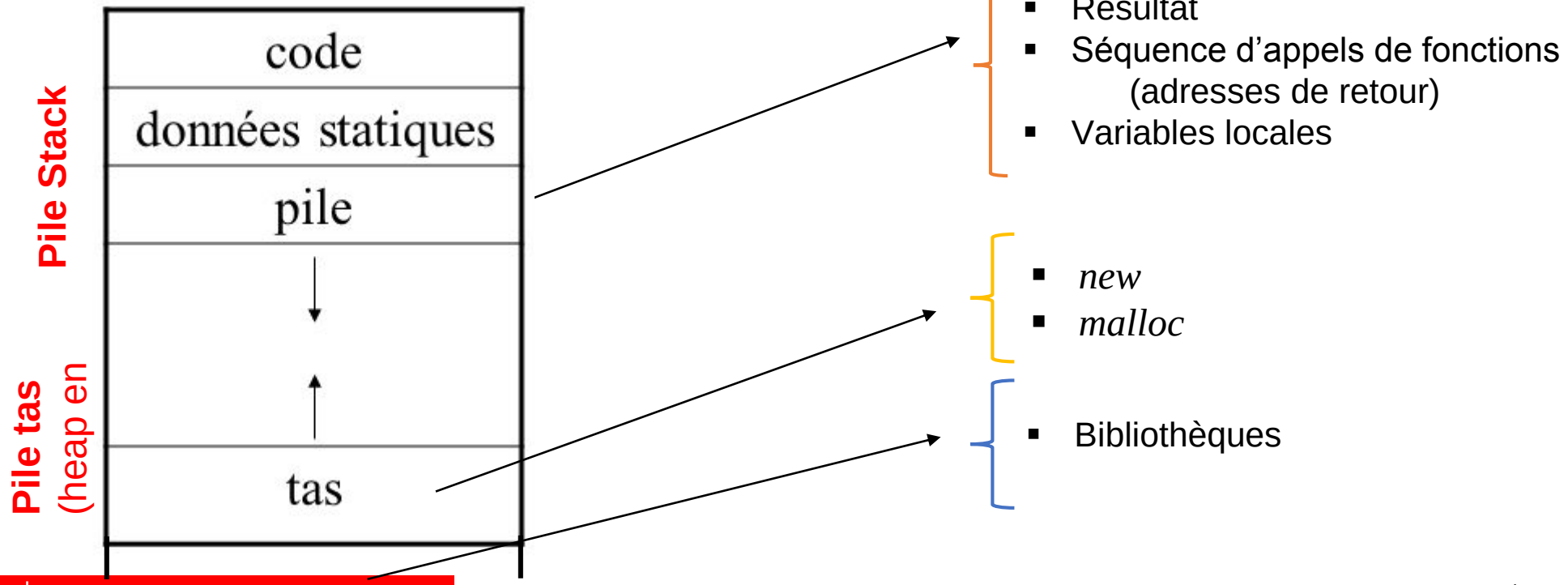
Section 1 : Notion de Processus

Programme vs processus

Programme ≠ Processus

description statique

Instance d'une tâche en cours d'éxecution



Qu'est ce qu'un Processus?

- Processus
  - Le terme a été introduit dans les années 60 pour généraliser "job concept".
  - Instance de programme binaire
  - Différents processus peuvent exécuter différentes instances du même programme.
- Au minimum, l'exécution du processus nécessite les ressources suivantes:
  - **Mémoire** pour contenir le code de programme et les données.
  - Un **ensemble de registres CPU** pour supporter l'exécution.
    - ⇨ Nécessité de la gestion
    - ⇨ Unité de base d'éxecution dans le SE (thread)

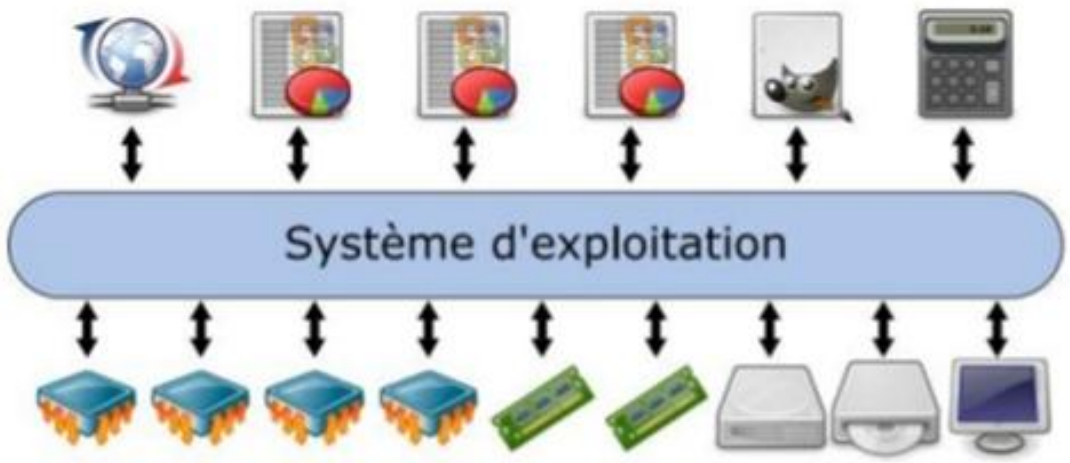
Processus et ressources

- SE: Interface entre le programme et le matériel.

Processus = Processeur virtuel

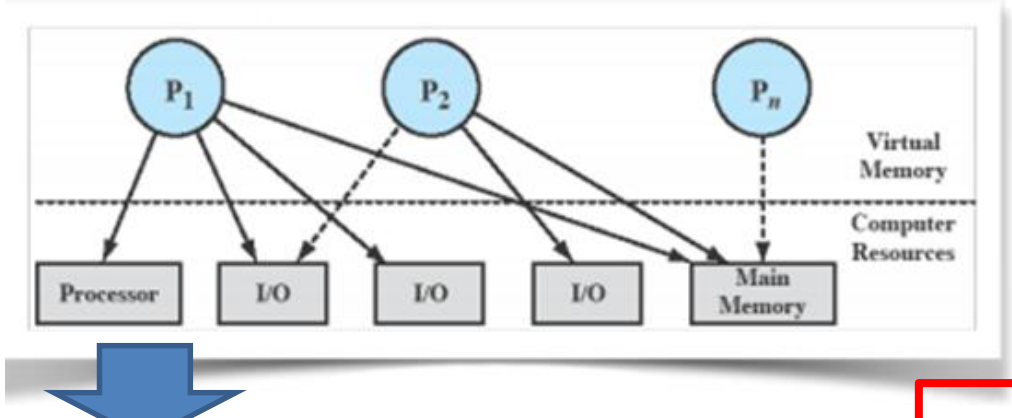
CPU, Mémoire

Ressources virtualisation



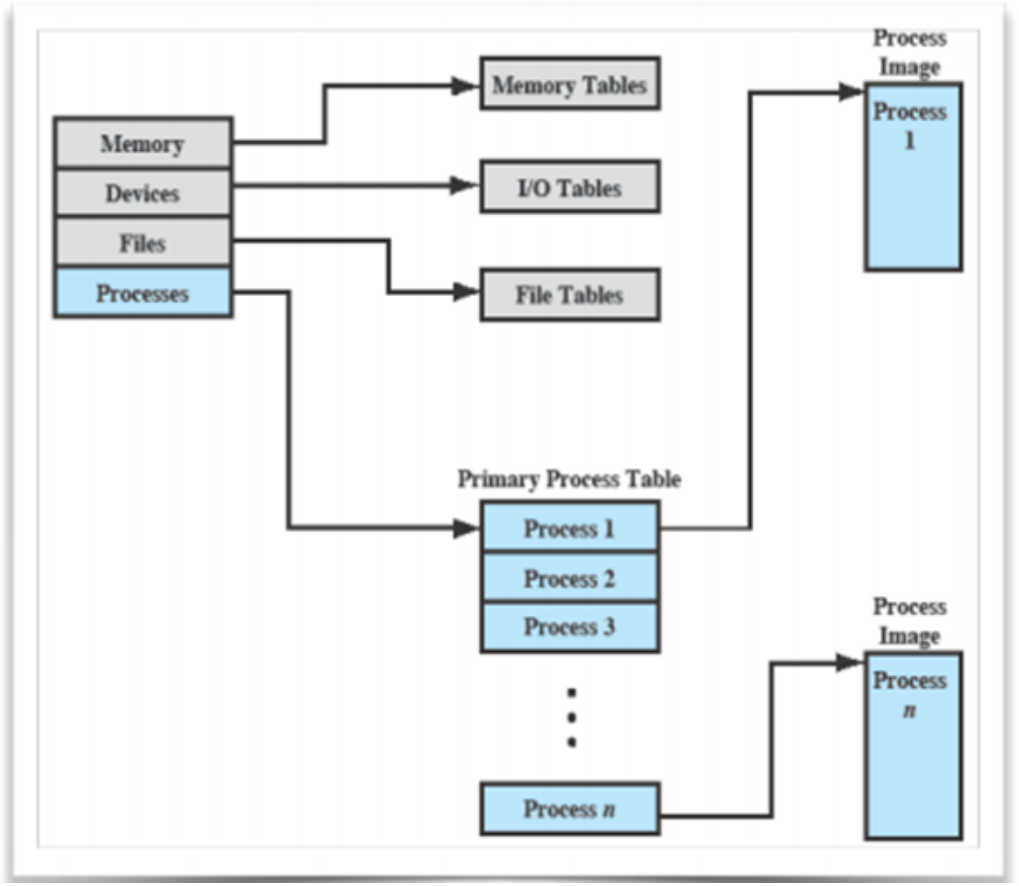
Processus et ressources

- **SE**: Gère l'utilisation des **resources de processus** (processor cycles, main memory, I/O devices)



Tables de Ressources

- **Tables de mémoire**
  - Garder une trace de la mémoire physique et de la mémoire virtuelle par processus
  - Allocation de l'espace de permutation aux processus
  - Attributs de protection
- **Tables d'E/S**
  - État de l'opération d'E/S en cours
  - Emplacement de mémoire utilisé comme source ou destination de l'opération
- **Tables de fichiers**
  - Les informations sur l'emplacement et les attributs des fichiers peuvent être gérées par le noyau
  - Toutes les tables de ressources du SE sont également sujettes à la gestion de la mémoire



Section 1: Notion de processus

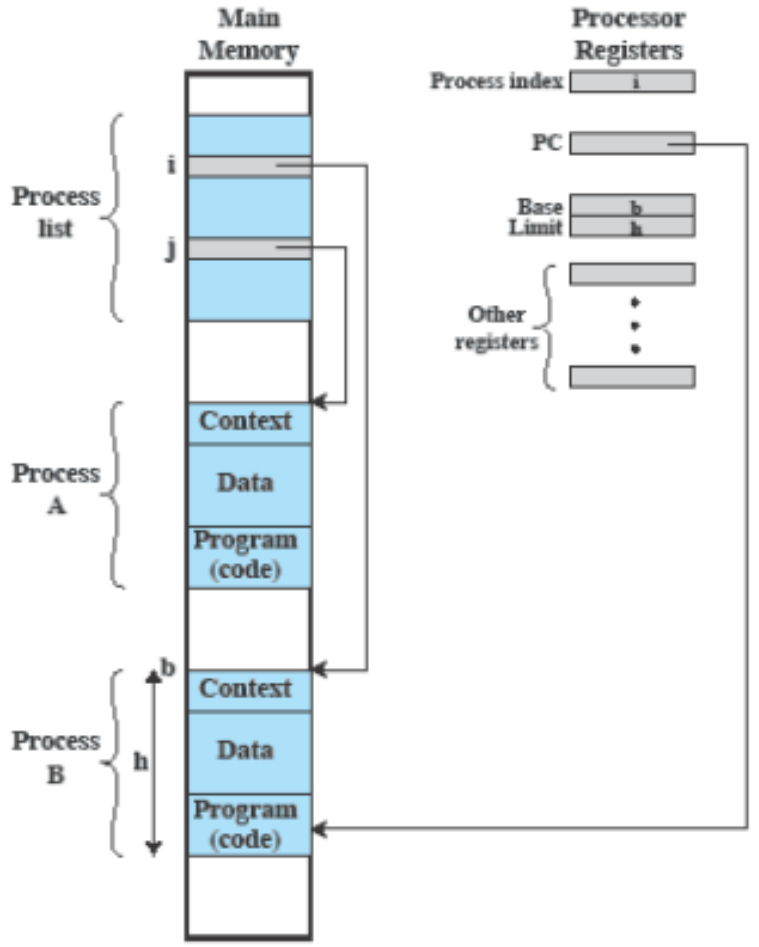
Section 2 : Contexte et états des processus

Contexte d'un processus (Process Image)

C'est l'ensemble des informations que les actions du processus peuvent consulter ou modifier.

Ces informations sont :

1. **Contexte du processeur** (mot d'état et registres généraux) ;
2. **Contexte du mémoire** : c'est l'espace de travail qui comprend :
   - Les segments procédures
   - Les segments données
   - Les piles d'exécution
3. **Ensembles d'attributs du processus** :
   - **Nom** : nom interne désignant le PCB du processus, qui sera discuté plus tard
   - **Priorité** : en fonction du *scheduling*
   - **Droits** : opérations permises par le processus

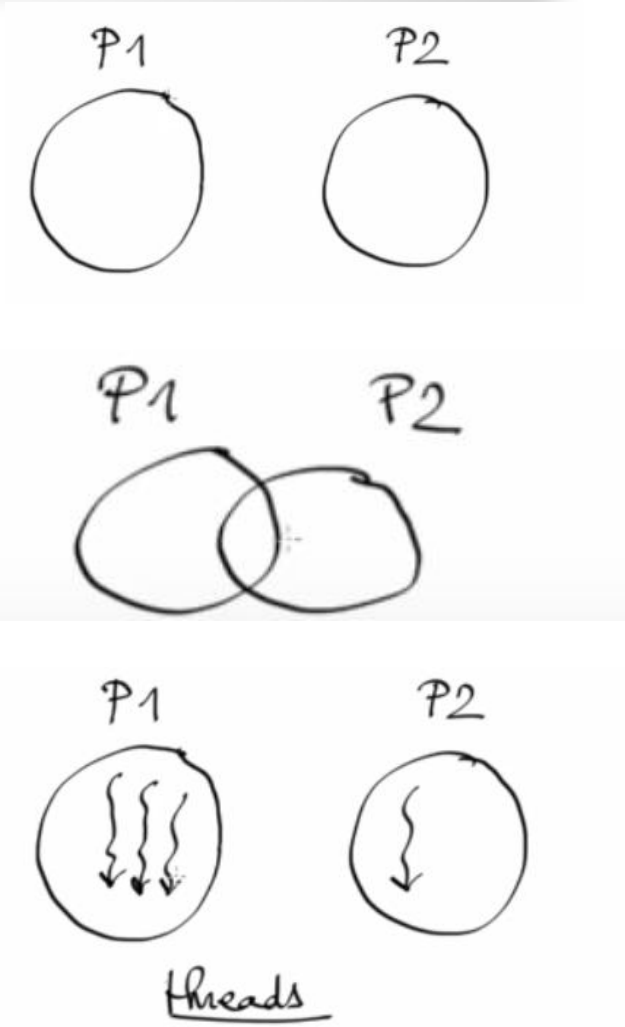


Relations entre processus

- Chaque processus dispose de son propre espace mémoire

- Ne pas perturber les autres (sécurité)

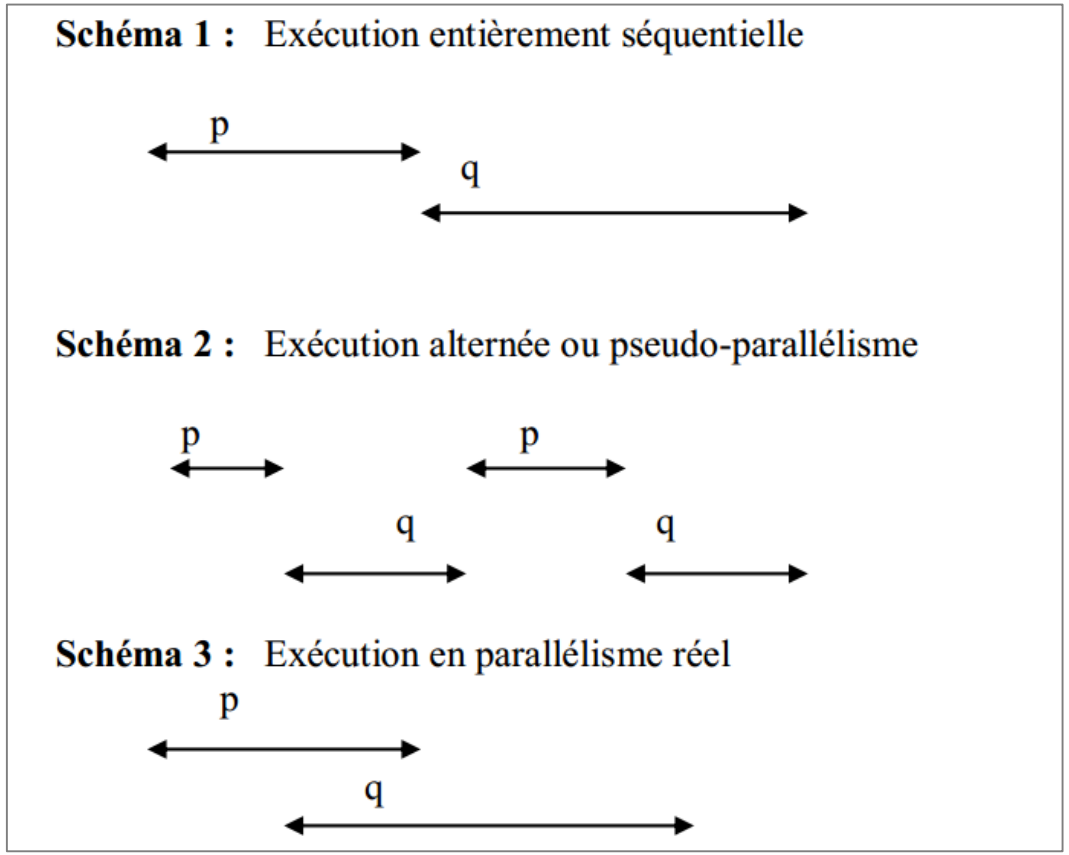
- Différents flots d'exécution partagent les même ressources des processus



Modes d'exécution des processus

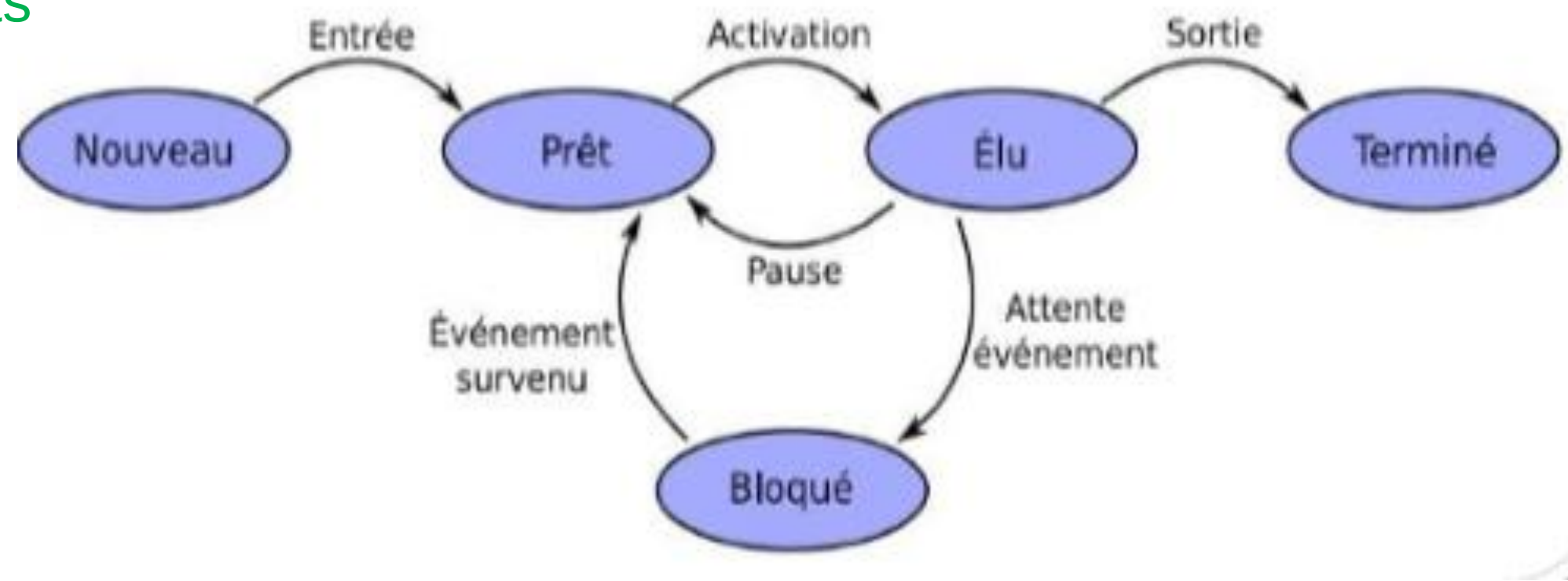
Afin d'exécuter des programmes, il faut mettre à leur disposition les ressources nécessaires à leur exécution :

- Processeur
- Espace mémoire
- Données nécessaires



Etats des processus
Cycle de vie d'un processus

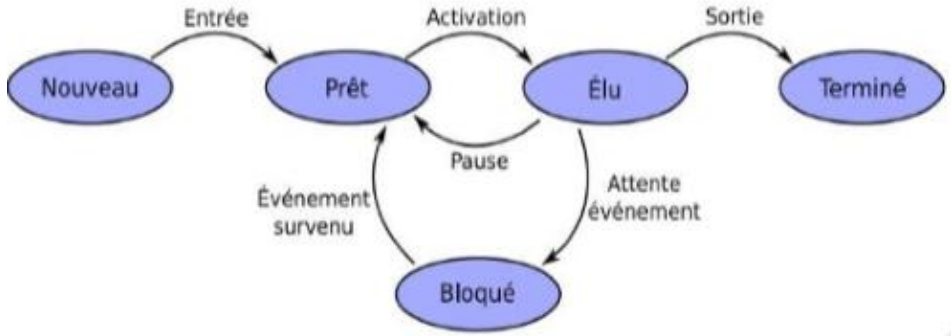
5 états



Unix SVR4: Process States

Same dispatch queue

User Running	Executing, in user mode
Kernel Running	Executing, in kernel mode
Ready to Run, in Memory	Ready to run as soon as being chosen
Asleep in Memory	Waiting for an event, unable to execute
Ready to Run, Swapped	Ready to run, but still not in memory
Sleeping, Swapped	Waiting for an event, unable to execute, swapped out
Preempted	Returning from kernel mode, but the kernel switched to another one
Created	Process is newly created, not ready to run so far
Zombie	Process is gone with his resource allocations, but record is left for parent process



Section 2: Contexte et états des processus

Les informations nécessaires pour implémenter un processus ?

- Une mémoire virtuelle
- un contexte d'exécution

Section 3 : Implémentation des processus

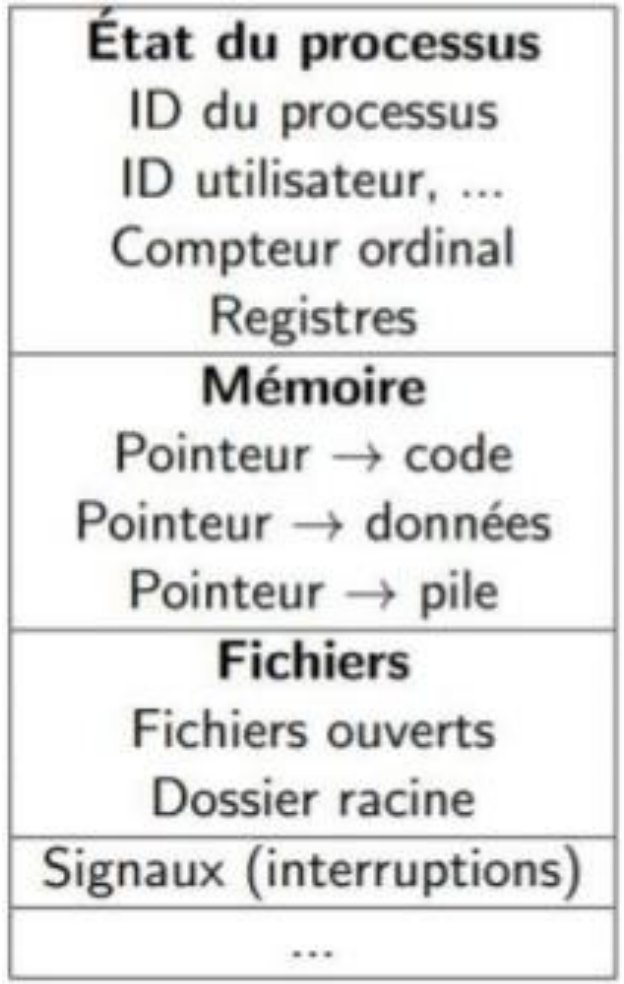
Bloc de contrôle de processus (PCB) **ou** descripteur de processus

Processus: implémentation

PCB

Le *Process Control Block* (PCB) est une **structure de de données** qui contient toutes les informations nécessaires à la bonne exécution du processus.

Les PCB sont stockés dans une **table des processus**

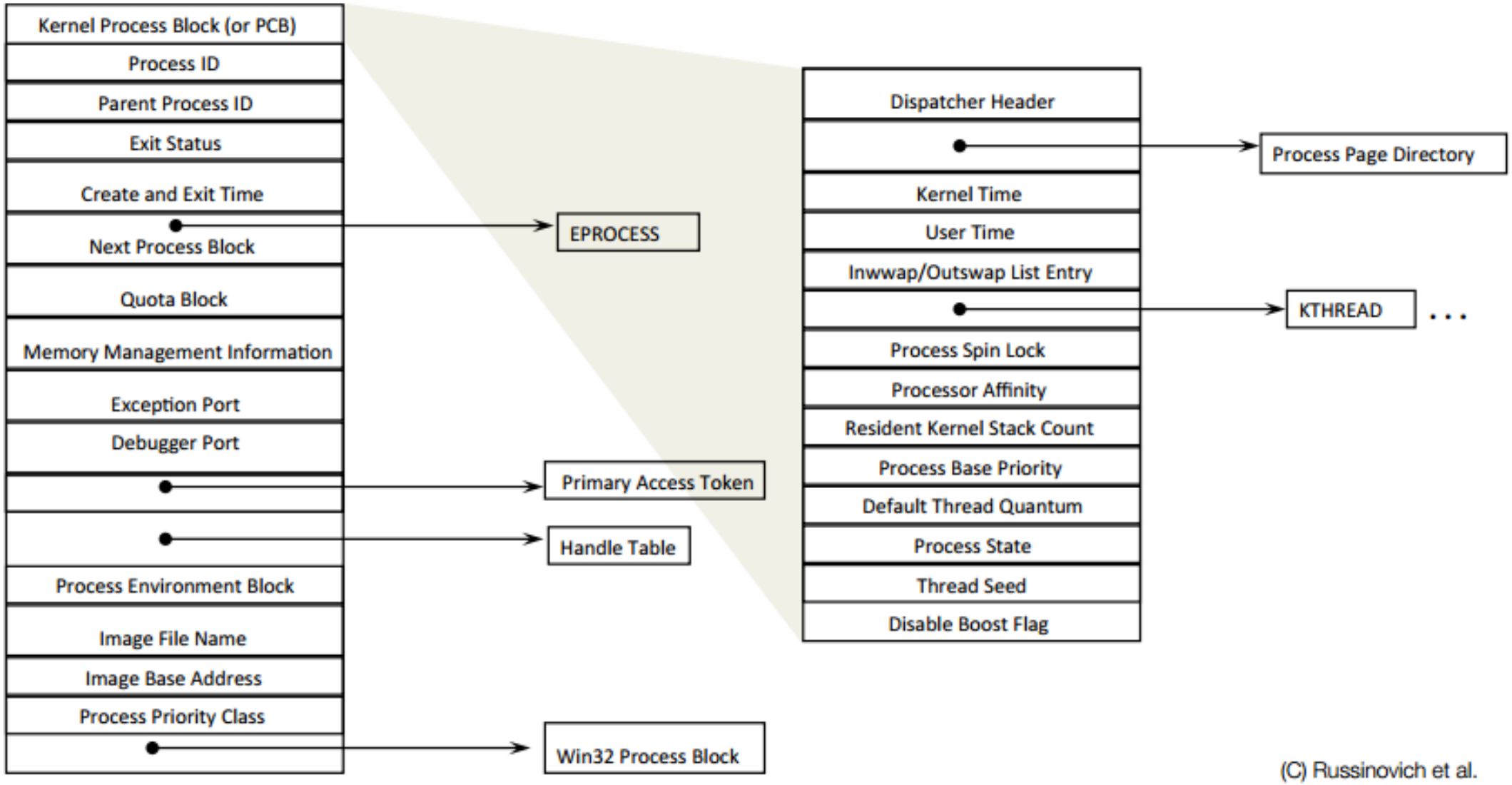


Process control block

- **Process state:** The state may be new, ready running, waiting, halted, and so on.
- **Program counter:** The counter indicates the address of the next instruction to be executed for this process.
- **CPU registers:** The registers vary in number and type, depending on the computer architecture. They include accumulators, index registers, stack pointers, and general-purpose registers, plus any condition-code information. Along with the program counter, this state information must be saved when an interrupt occurs, to allow the process to be continued correctly afterward
- **CPU-scheduling information:** This information includes a process priority, pointers to scheduling queues, and any other scheduling parameters.
- **Memory-management information:** This information may include such information as the value of the base and limit registers, the page tables, or the segment tables, depending on the memory system used by the operating system.
- **I/O status information:** This information includes the list of I/O devices allocated to the process, a list of open files, and so on.

In brief the PCB simply serves as the repository for any information that may vary from process to process.

Windows: Process Control Block



(C) Russinovich et al.

Section 3 : Implémentation de processus

Section 4 : Opérations sur les processus

Opérations sur les processus

- Un processus est vu comme un objet par le système d'exploitation.
  - un identificateur unique (PID)
    - peut appliquer les opérations suivantes :
      - Création
      - Destruction
      - Blocage
      - Réveil
      - Activation
      - Désactivation
      - Suspension …
  - Un noyau de système se compose de toutes ces opérations

Création de processus

Cette opération consiste à :

1. - Allouer un descripteur à un processus (PCB)
2. - Affecter un identificateur unique au processus (PID)
3. - Initialiser le descripteur (PCB) : programme à exécuter, pile d'exécution, mémoire centrale utilisé par le programme et les données du processus, état du processus, priorité du processus, autres ressources…etc

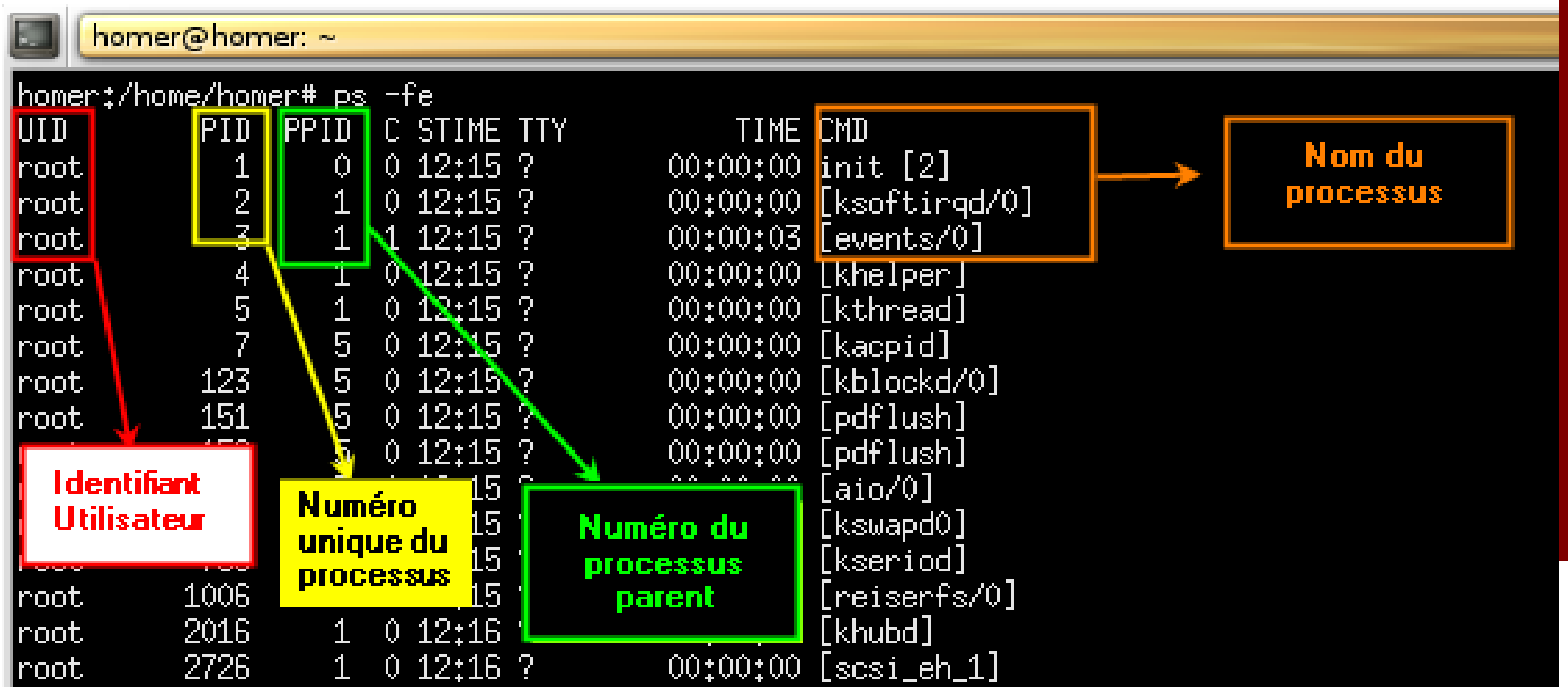
Création de processus

- Un processus peut créer plusieurs nouveaux processus en exécutant

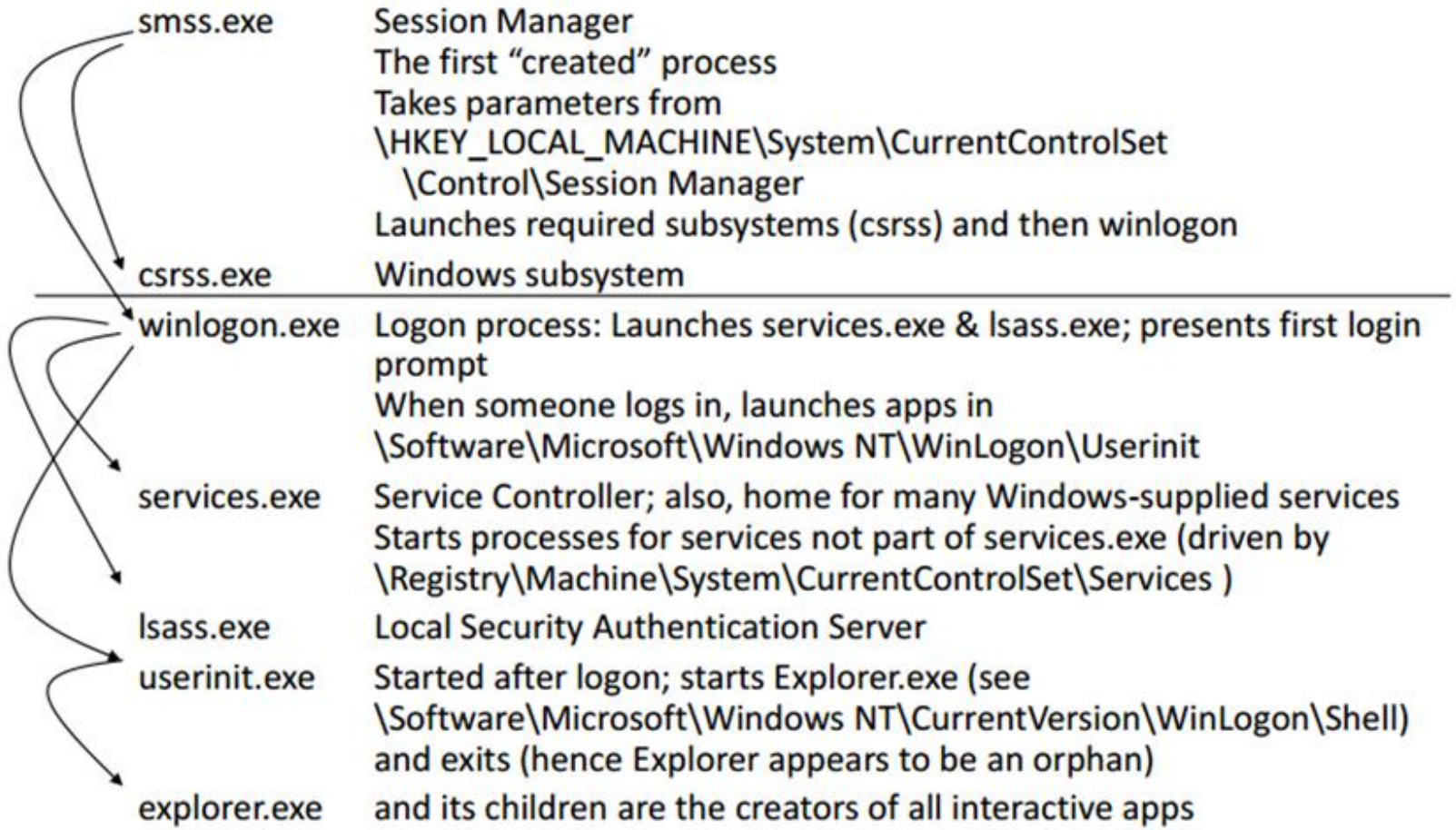
des appels système de création de processus.

- Le processus effectuant la création est dit processus père, tandis que les nouveaux processus sont ses fils.
- Chacun des nouveaux processus peut encore créer d'autres processus (fils) créant ainsi un arbre de processus. Un processus père doit toujours connaître l'identité de ses fils.

***ps* (i.e., *process* status)**



Windows: Processes during System Start



smss.exe	Session Manager The first "created" process Takes parameters from \HKEY_LOCAL_MACHINE\System\CurrentControlSet \Control\Session Manager Launches required subsystems (csrss) and then winlogon
csrss.exe	Windows subsystem
winlogon.exe	Logon process: Launches services.exe & lsass.exe; presents first login prompt When someone logs in, launches apps in \Software\Microsoft\Windows NT\WinLogon\Userinit
services.exe	Service Controller; also, home for many Windows-supplied services Starts processes for services not part of services.exe (driven by \Registry\Machine\System\CurrentControlSet\Services)
lsass.exe	Local Security Authentication Server
userinit.exe	Started after logon; starts Explorer.exe (see \Software\Microsoft\Windows NT\CurrentVersion\WinLogon\Shell) and exits (hence Explorer appears to be an orphan)
explorer.exe	and its children are the creators of all interactive apps

fichiers batch: *Net stop* nom_du_servic et *Net start* nom_du_servic

Destruction de processus

La destruction d'un processus ne peut être effectuée que par le **processus père** ou un **processus système**.

Cette opération consiste à :

- Libérer les ressources occupées par le processus
- Détruire ou non la descendance du processus (Unix ne détruit pas les processus fils)
- Libérer son descripteur (PCB), qui pourra être réutilisé
- Le PID est supprimé mais ne peut plus être réutilisé

Destruction de processus

- Arrêt volontaire
- Erreur
- Erreur fatale

Processus suspendu

- Le processeur est plus rapide que I/O, alors tous les processus puissent attendre le périphérique d'E / S.
  - Échangez ces processus sur disque (Swap) pour libérer plus de mémoire et utiliser le processeur sur plus de processus.
  - L'état bloqué devient l'état de suspension quand il est échangé sur le disque
- **Deux nouveaux états**
  - Blocked/Suspend
  - Ready/Suspend

Les raisons:

- **Swapping**:
  - L'OS doit libérer de l'espace Pour exécuter le processus (état prêt).
- **Causes SE:**
  - Le système d'exploitation suspendue le processus défectueux ou à cause d'un problème.

TP: Commandes Linux de base: La gestion des processus

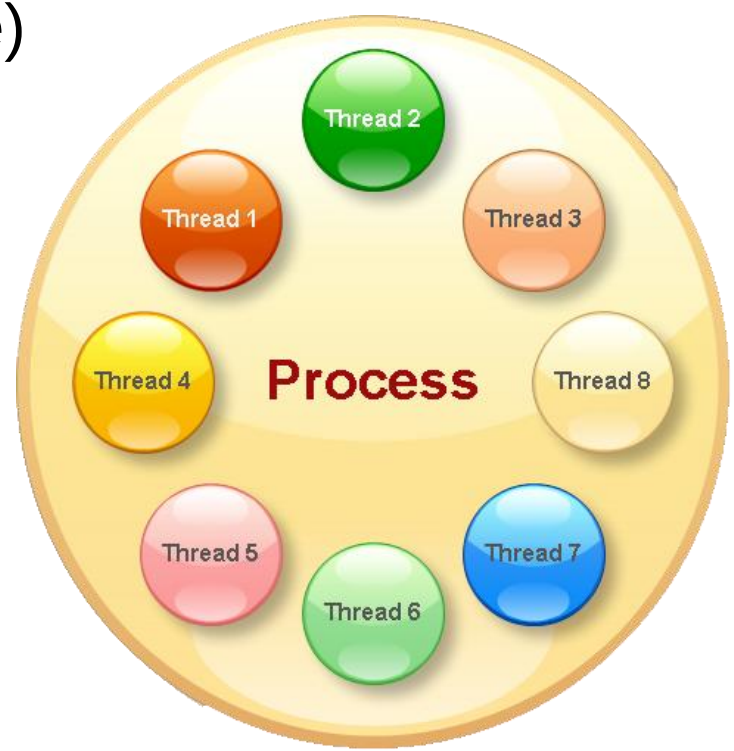
Gestion des processus	
ps auxr	Liste des process en cours d'exécution
ps aux|more	Liste de tous les process
top	Suivi de l'activité de la machine
&	Mise en arrière plan d'un processus
prog &	Lancement de prog en arrière plan
fg	Mise en avant plan d'un processus stoppé
jobs	Liste des jobs en arrière plan
kill %1	Tue le job d'arrière plan [1]
kill 1492	Tue le processus de PID 1492
free	Espace mémoire disponible

Section 4 : Opérations sur les processus

Section 5 : Les thread et le parallélisme

Les thread et le parallélisme

- Un thread (appelée aussi processus léger ou activité) est un fil d'instructions (un chemin d'exécution) à l'intérieur d'un processus



Un thread est donc une portion de code capable de s'exécuter en parallèle à d'autres traitements.

Il existe deux approches d'implémentation :

- bibliothèques utilisateur (API),
- supportés au-dessus du noyau (appels système).

Que faire avec fork/join ?

Les mécanismes pour effectuer plusieurs tâches de façon concurrente, appelé « **fork** ») et « **join** ».

Plusieurs librairies implémentent cette idée:

(Intel Thread Building Blocks, Microsoft Task Parallel Library, Java 7 ForkJoin).

Nous pouvons maintenant poser des questions de nature plus algorithmique :

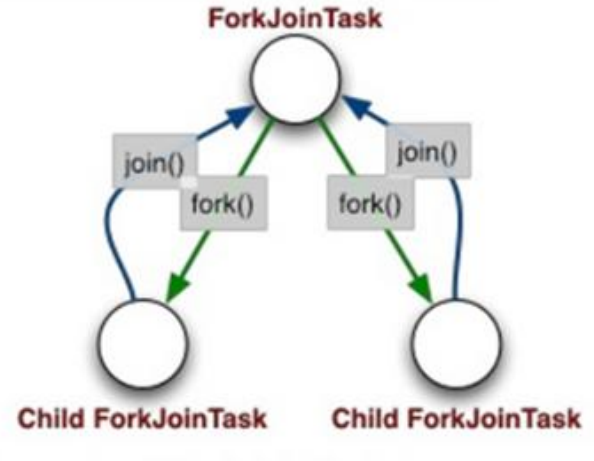
- Comment utiliser ces mécanismes pour obtenir un gain de performance ?
- Comment prédire le gain de performance que l'on peut espérer ?

Process Tree

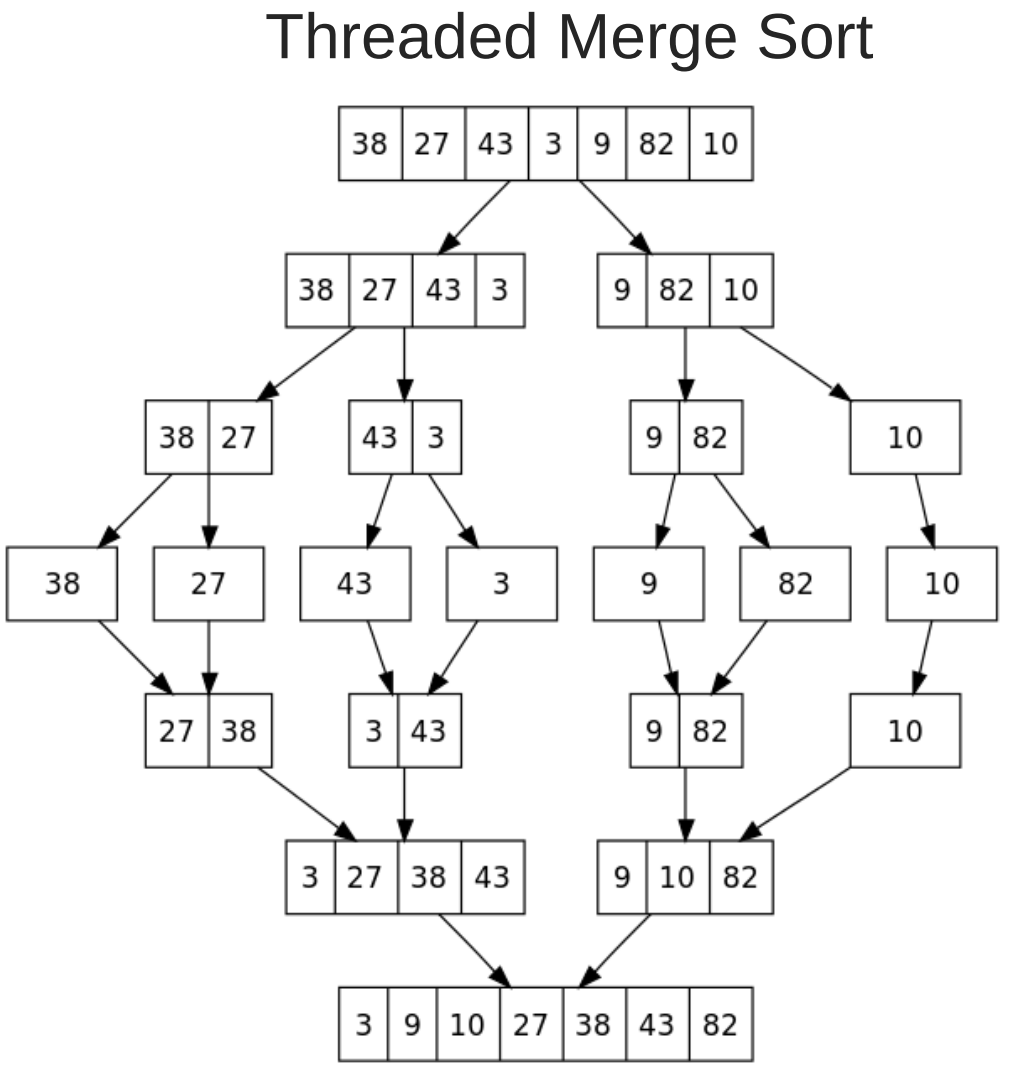
Fork and Join Framework

Tâche -> Sous Tâches

- diviser pour régner
- Comme map and reduce
- Obtenir un gain de performance
- Contraintes de dépendances entre tâches

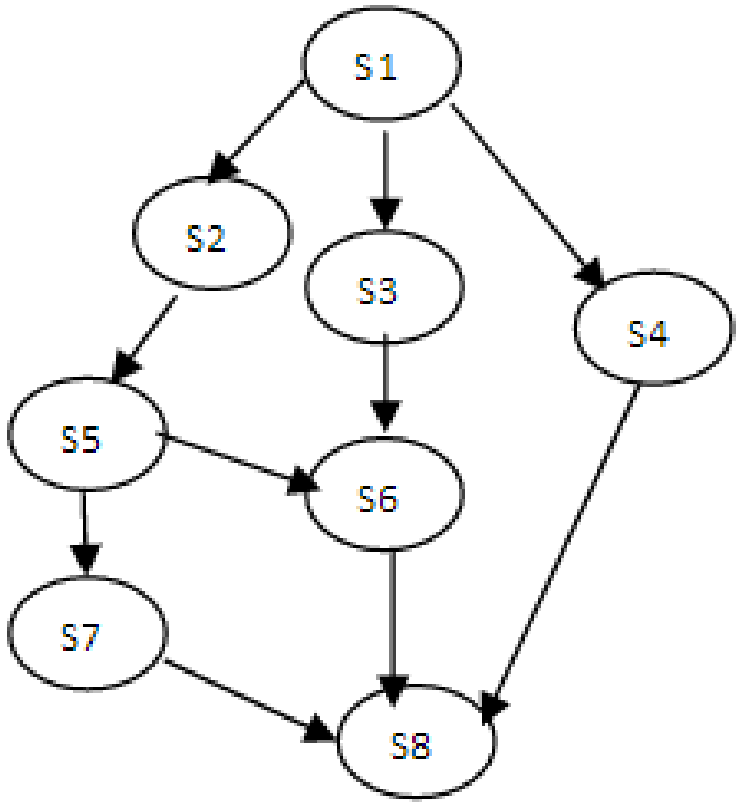


Exemple: Process Tree



Exemple
le programme parallèle (primitives fork/join

- Soit le graphe de précédence suivant :




```
N1=2, n2=3;
S1 ;
Fork L1;
S2 ;
S5;
Fork L3;
S7;
Goto L4;
```

```
L1: fork L 2;
S3;
L3: join n1;
S6;
Goto L4;
L2: S4;
L4: join n2;
S8;
```

Exercice 1 : Parallélisation maximale d'une fonction

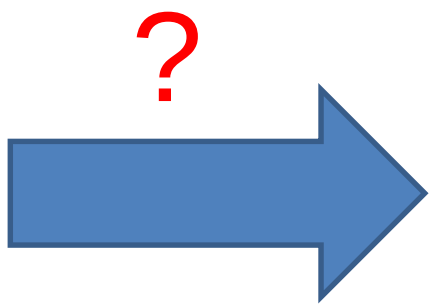
Programmer en langage C, à l'aide des fonctions C fork et pipe (ou bien Java 7 : fork / join), le graphe de parallélisation maximale G permettant d'évaluer la formule :

$$y = 2 * [(a + b) / (c-d) + (e * f)] + [(a + b) * (c-d)]$$

Exercice 1 : Parallélisation maximale d'une fonction

Donner le graphe de précédence

```
begin
        parbegin
                lire(a)
                lire(b)
        parend ;
        parbegin
                c := a*a
                begin
                        d := a*b ;
                        e := d*a ;
                end
        parend ;
        e := c + d + e
end
```



Section 5 : Les thread et le parallélisme