Université de Jijel
Faculté des Sciences exactes et Informatique
Département d'informatique

SYSTÈME D'EXPLOITATION I

Pour 2ème Année Licence - informatique

PLAN DU COURS

- **Chapitre1**: Introduction aux système d'exploitation
- **Chapitre2**: Mécanismes de base d'exécution des programmes
- **Chapitre3**: Gestion des processus
- **Chapitre4**: Gestion de la mémoire
- **Chapitre5**: gestion des entrées/ sorties

Chapitre 1

INTRODUCTION AUX SYSTÈME D'EXPLOITATION

Introduction

Le système d'exploitation d'un ordinateur ou d'une installation informatique est un ensemble de programmes qui remplissent deux grandes fonctions :

- gérer les ressources de l'installation matérielle en assurant leurs partages entre un ensemble plus ou moins grand d'utilisateurs
- assurer un ensemble de services en présentant aux utilisateurs une interface mieux adaptée à leurs besoins que celle de la machine physique

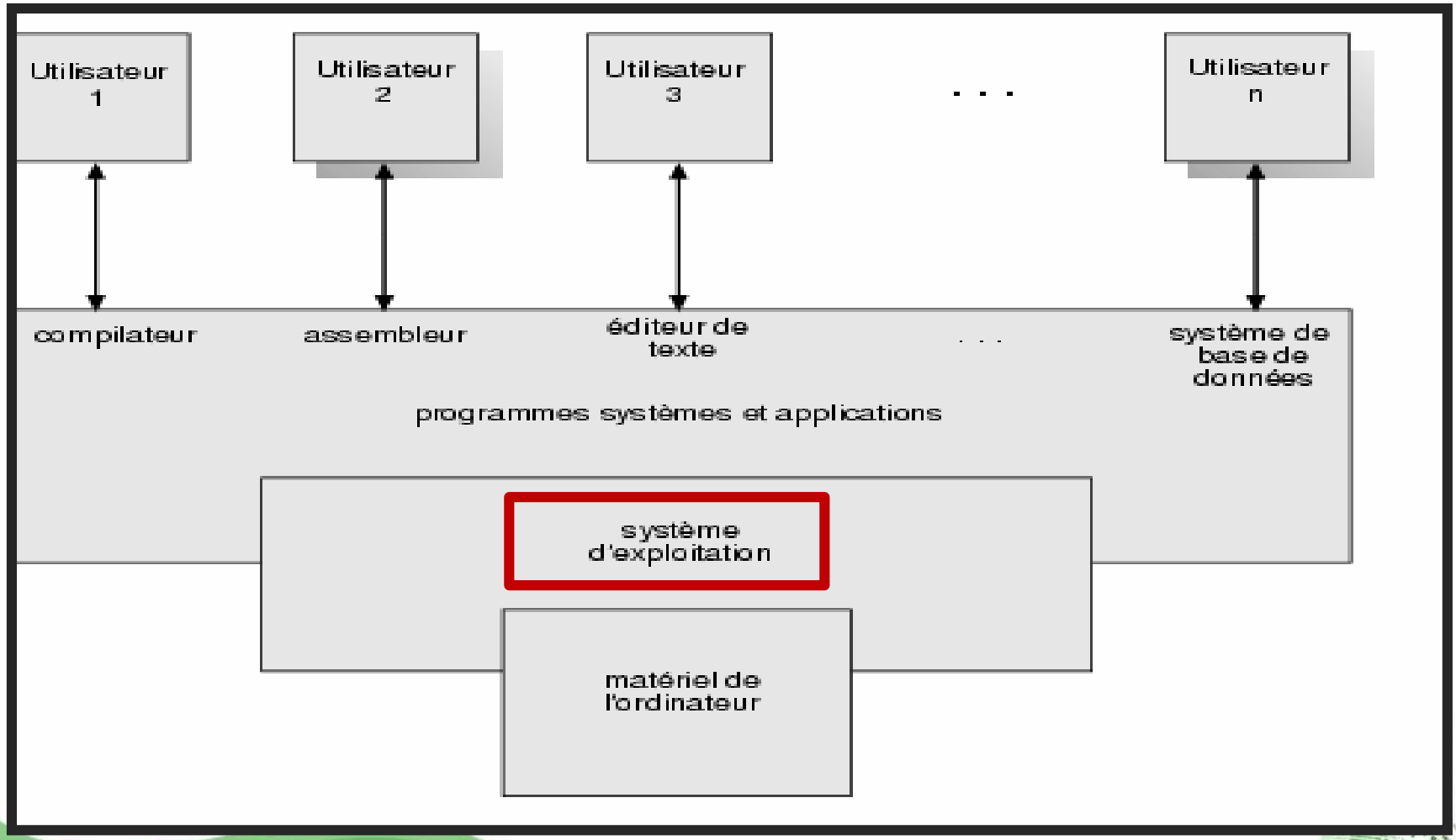
Définition

Le programme « système d'exploitation » est le programme fondamental des programmes systèmes. Il contrôle les ressources de l'ordinateur et fournit la base sur la quelle seront construits les programmes d'application.

Les rôles d'un système d'exploitation

Le système d'exploitation (Operating System, O.S.) est l'intermédiaire entre un ordinateur (ou en général un appareil muni d'un processeur) et les applications qui utilisent cet ordinateur ou cet appareil. Son rôle peut être vu sous deux aspects complémentaires

Vue abstraite d'un SE



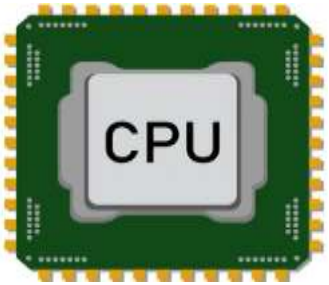
Les rôles d'un système d'exploitation

- Un système d'exploitation permet de répondre à deux besoins qui **ne sont pas forcément liés** :
  1. le système d'exploitation en tant que **machine étendue** (ou « machine virtuelle »),
  2. le système d'exploitation en tant que **gestionnaire de ressources**.

En tant que machine étendue

- Le système d'exploitation correspond à « l'interface » entre les applications et le matériel.
- De ce point de vue le système d'exploitation peut être assimilé à une machine étendue ou virtuelle plus facile à programmer ou à utiliser que le matériel :
  - Un programmeur va utiliser le système d'exploitation par l'intermédiaire " **d'appels système** ".

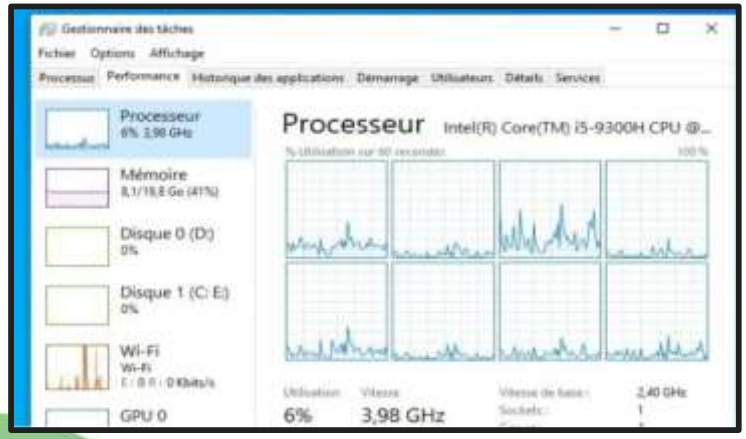
En tant que machine étendue



Processeur

RAM

Disque



En tant que gestionnaire de ressources

- Les différents composants d'un ordinateur doivent coopérer et partager des ressources.
- Dans cette optique, le travail du système d'exploitation consiste à :
  - ordonnancer,
  - contrôler l'allocation des ressources :
    - processeurs,
    - mémoires,
    - périphériques d'E/S,
    - ...

  entre les différents programmes qui y font **appel.**

Historique des systèmes d'exploitation

Pour comprendre ce que sont les SE, nous devons tout d'abord comprendre comment ils se sont développés. Cela permettra de voir comment les composants des SE ont évolué comme des solutions naturelles aux problèmes des premiers SE.

Porte ouverte ou exploitation self service (1945-1955)

Afin d'utiliser la machine, la procédure consistait:

- à allouer des tranches de temps directement aux Utilisateurs (programmeurs), qui se réservent toutes les ressources de la machine à tour de rôle pendant leur durée de temps.
- Les périphériques d'entrée/sortie en ce temps étaient respectivement le lecteur de cartes perforées et l'imprimante. Un pupitre de commande était utilisé pour manipuler la machine et ses périphériques.

Porte ouverte ou exploitation self service (1945-1955)

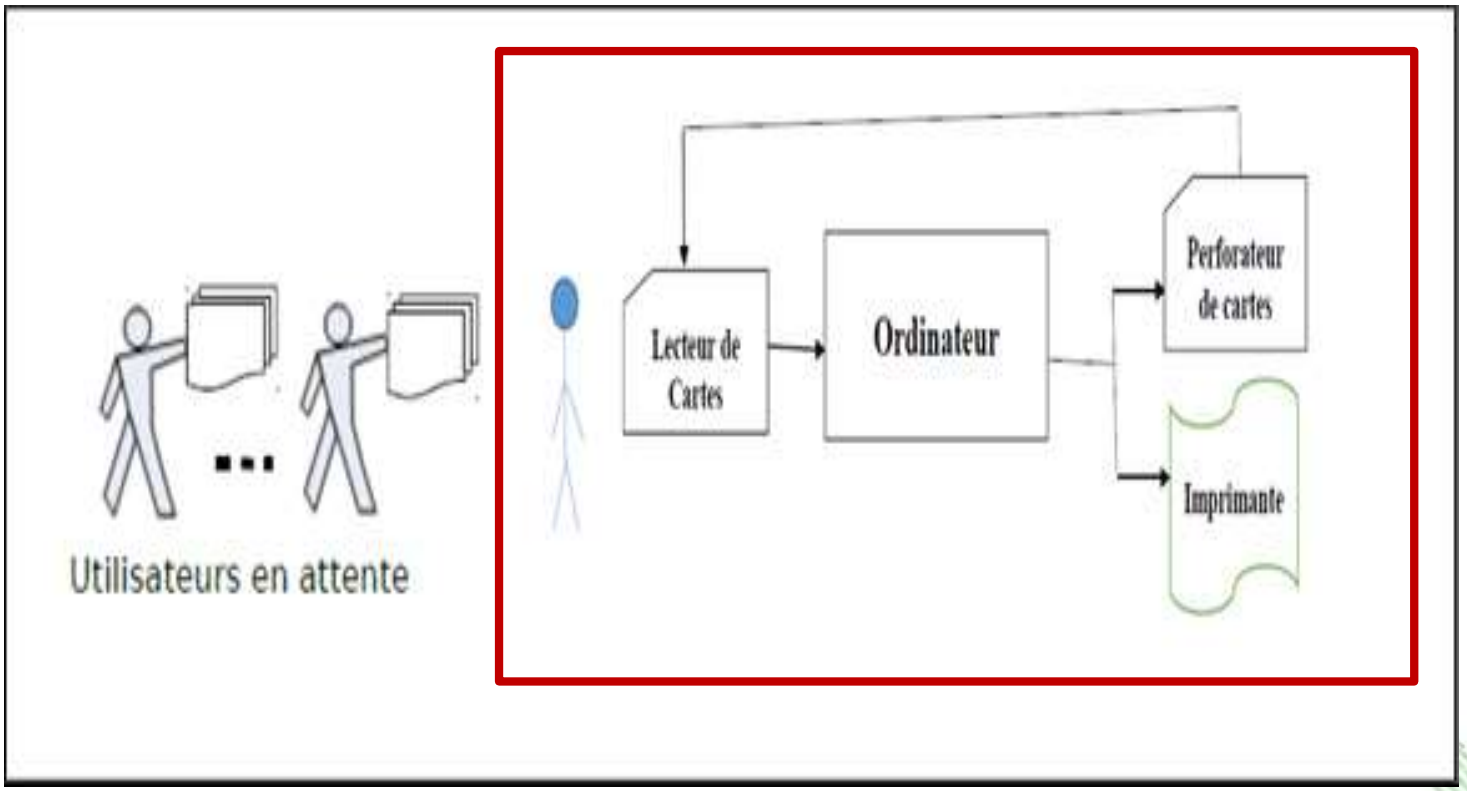
Chaque utilisateur, assurant le rôle d'**opérateur, devait lancer un ensemble d'opérations qui sont :**

- Placer les cartes du programme dans le lecteur de cartes.

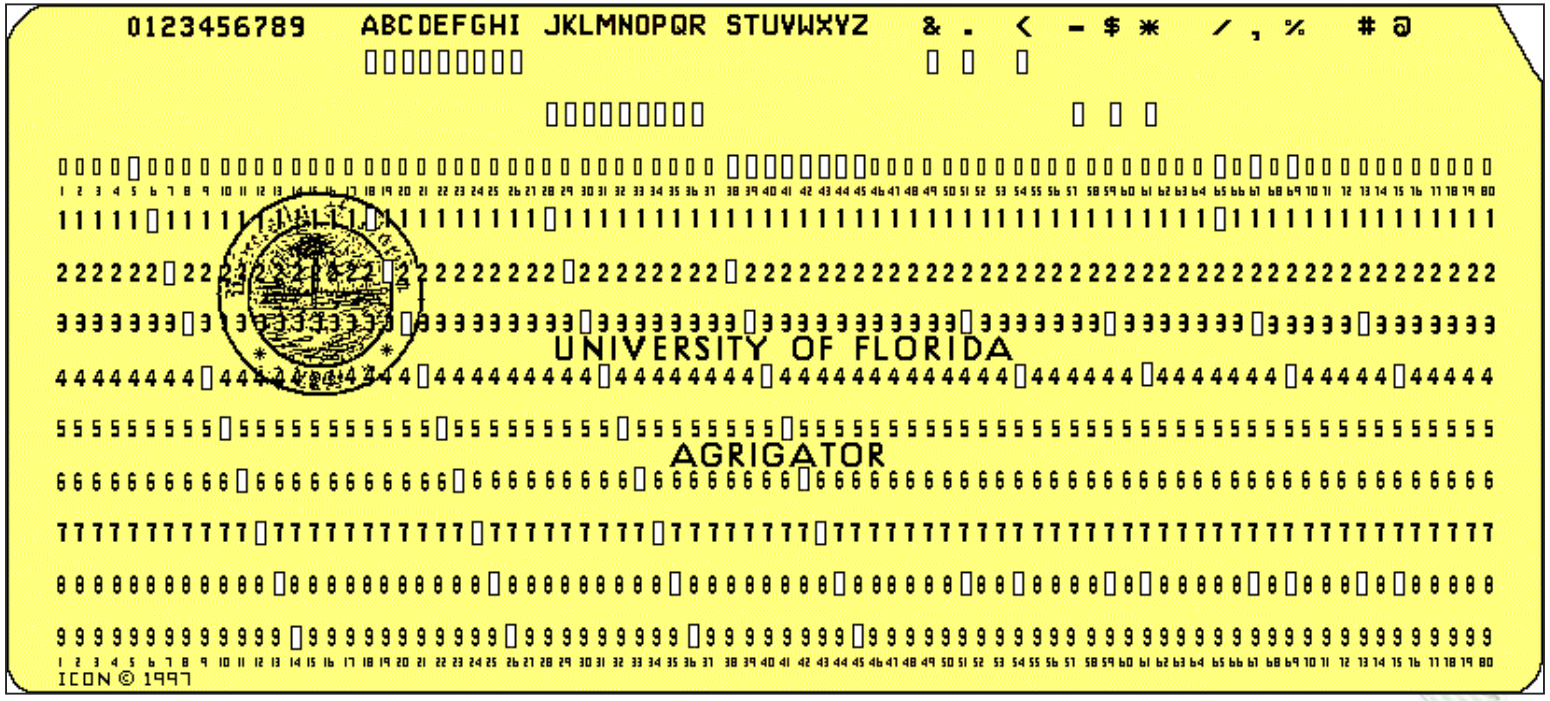
Initialiser un programme de lecteur des cartes.

- Lancer la compilation du programme utilisateur.
- Placer les cartes données s'il y en a, dans le lecteur de cartes.
- Initialiser l'exécution du programme compilé.
- Détecter les erreurs au pupitre et imprimer les résultats.

Porte ouverte ou exploitation self service (1945-1955)



Cartes perforées



Opérateur lisant un paquet de cartes perforées

Traitement par lots (Batch Processing) (1955-1965)

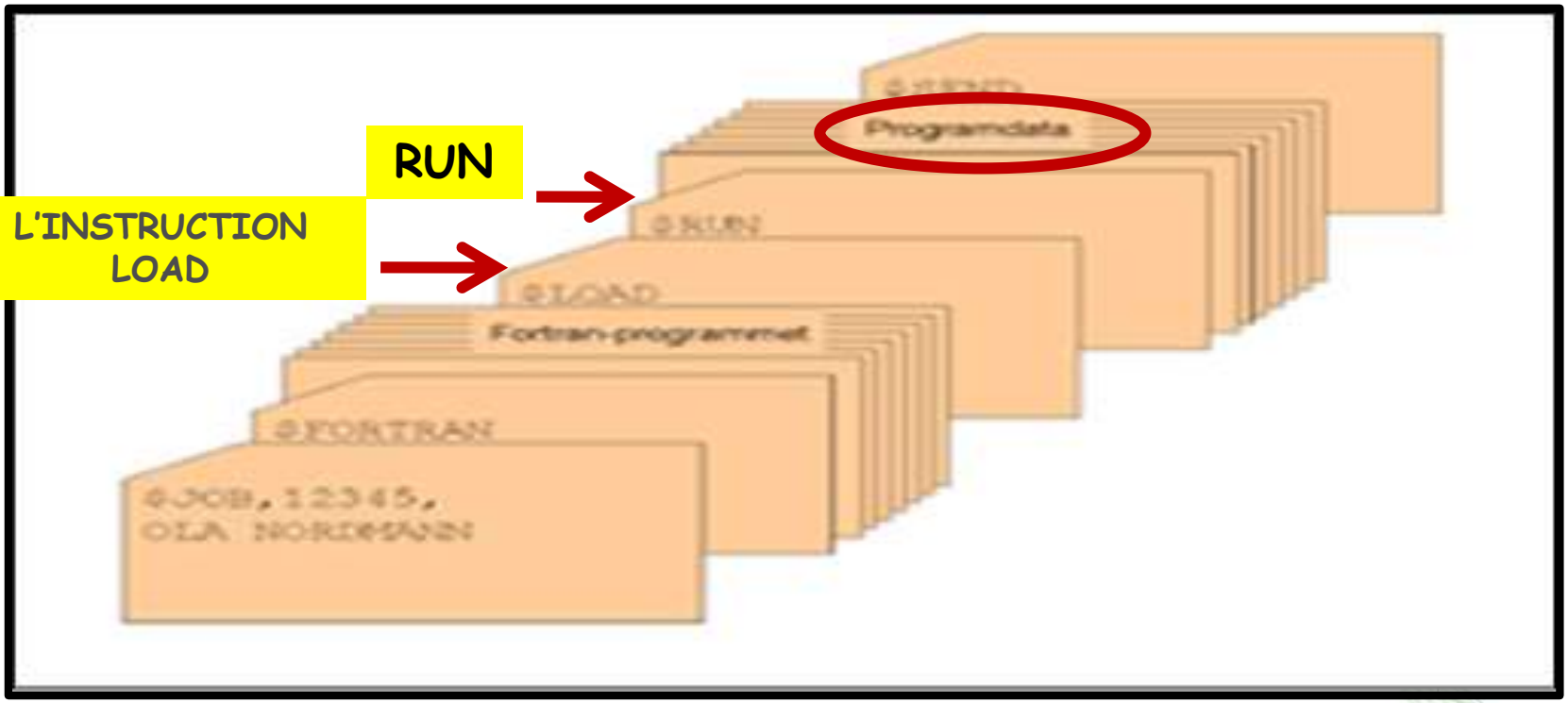
Ce sont des systèmes réalisant le séquencement des jobs ou travaux selon l'ordre des cartes de contrôle à l'aide d'un moniteur d'enchaînement. L'objectif était de réduire les pertes de temps .

Traitement par lots (Batch Processing) (1955-1965)

Le programmeur soumet une **job** à un opérateur

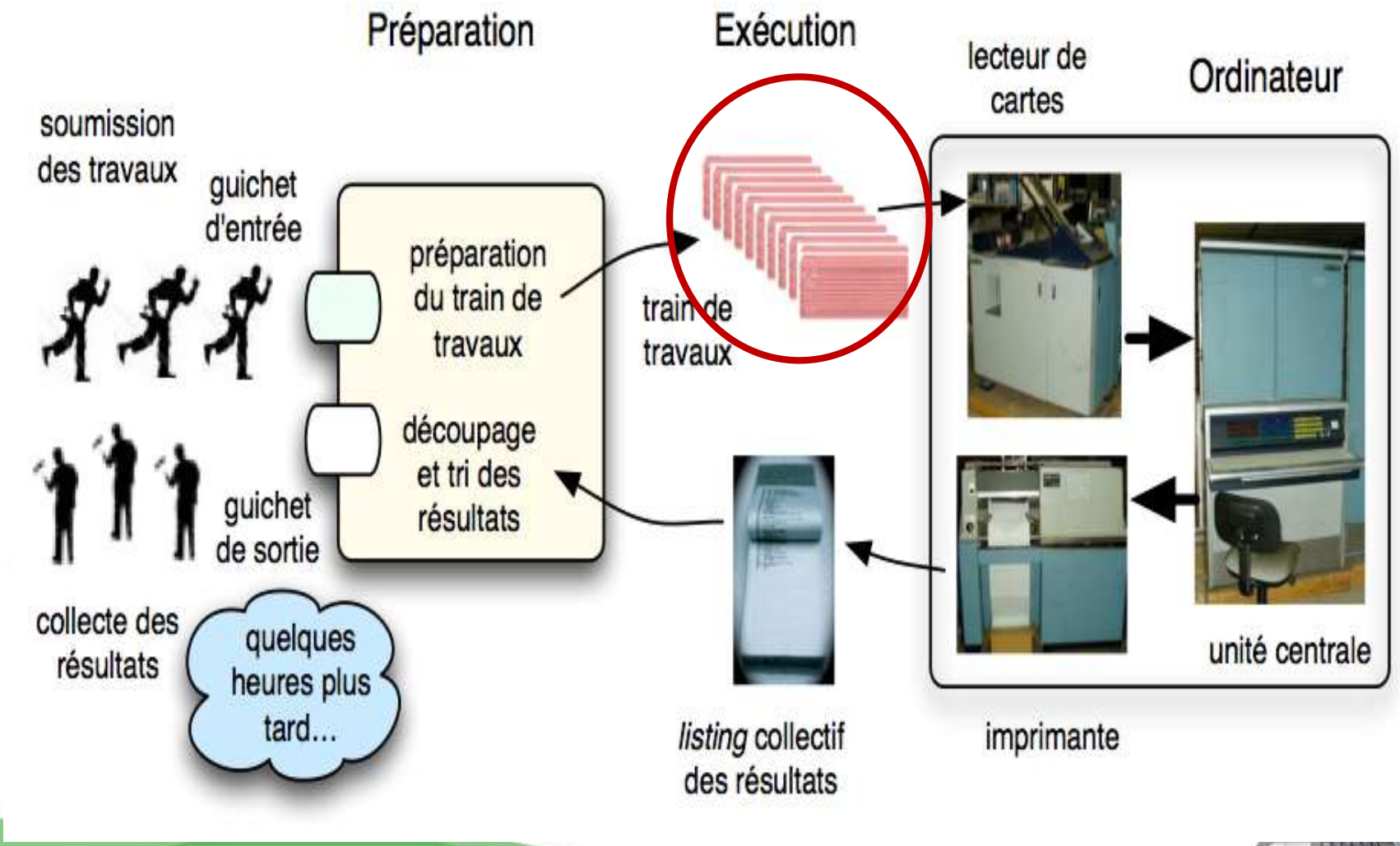
- Programme suivi par données
- L'opérateur place un lot de plusieurs jobs sur le dispositif de lecture
- Un programme, **le moniteur**, gère l'exécution de chaque programme du lot
- Le moniteur est toujours en mémoire et prêt à être exécuté
- Les utilitaires du moniteur sont chargés au besoin
- Un seul programme à la fois en mémoire, programmes sont exécutés en séquence

Traitement par lots (Batch Processing) (1955-1965)



Structure d'un travail FMS typique

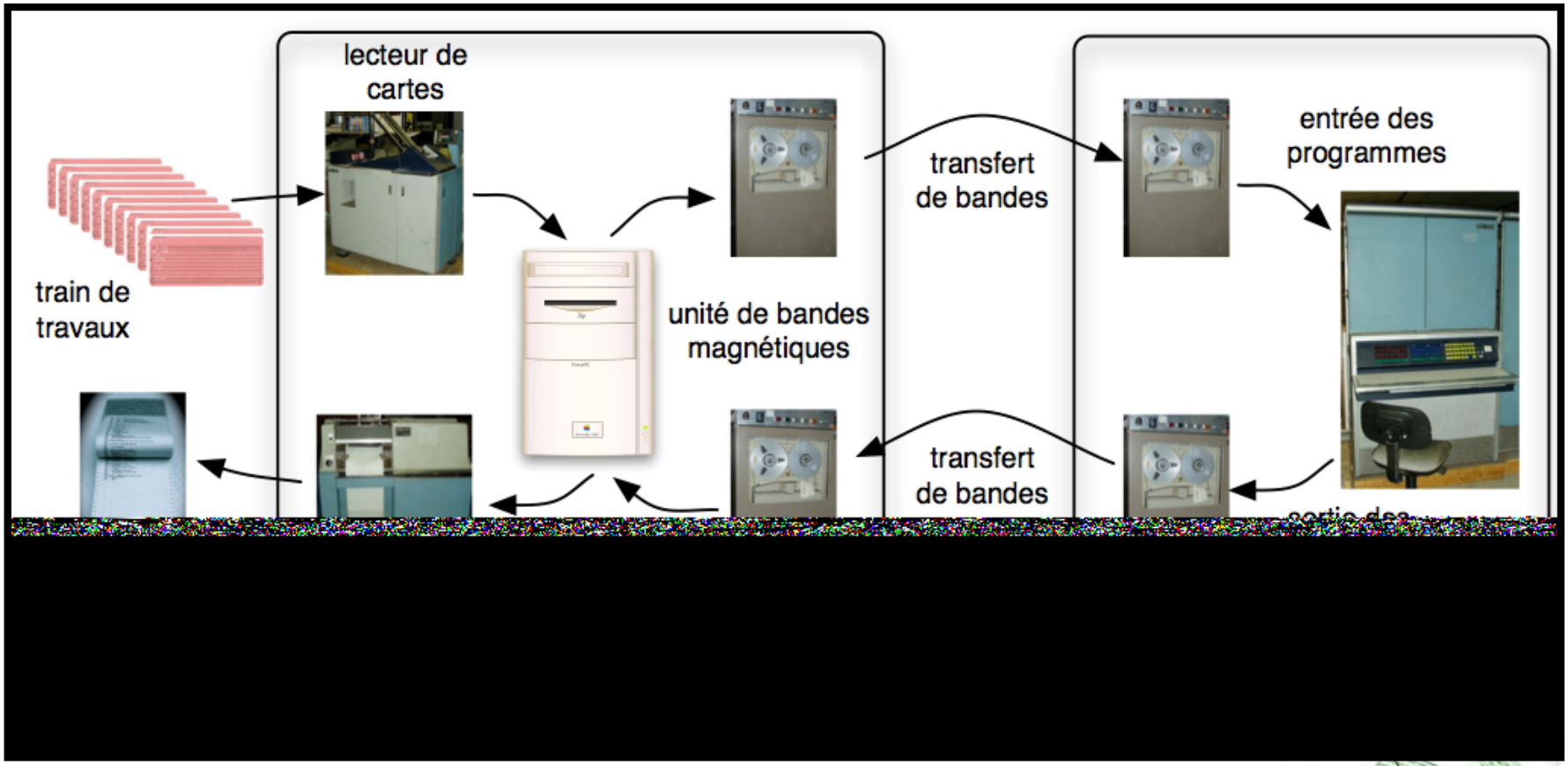
Traitement par lots (Batch Processing) (1955-1965)



Traitement par lots (Batch Processing)
(1955-1965)

La bande magnétique

Traitement par lots (Batch Processing) (1955-1965)



Traitement par lots avec ordinateur auxiliaire

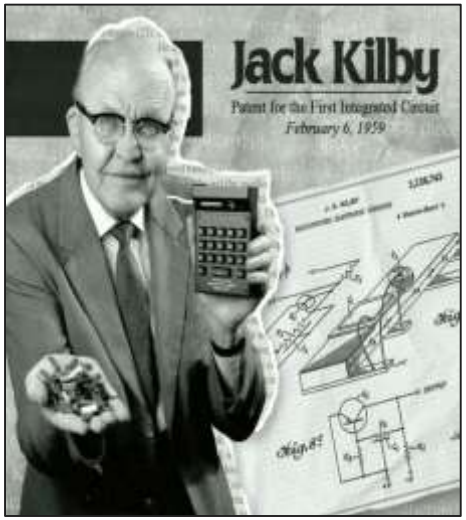
Traitement par lots (Batch Processing) (1955-1965)

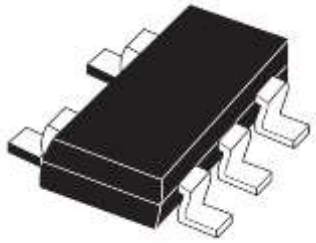
Multiprogrammation (Multiprogramming) (1965-1970)

L'introduction des circuits intégrés dans la construction des machines a permis d'offrir un meilleur rapport coût/performance. L'introduction de la technologie des disques a permis au système d'exploitation de conserver tous les travaux sur un disque.

Multiprogrammation (Multiprogramming) (1965-1970)

Transistor





Circuit intégré

Multiprogrammation (Multiprogramming) (1965-1970)

Disque dur
(IBM,1954,1962)

RAM

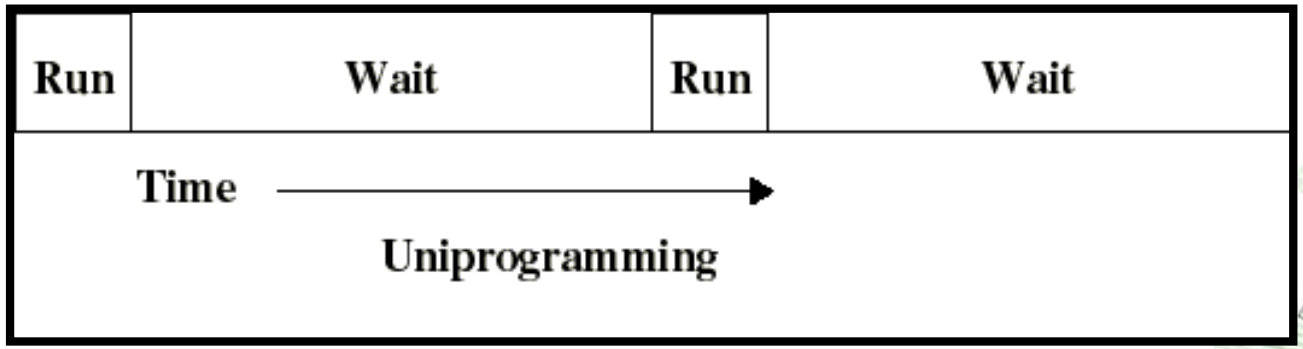
Exécution

Multiprogrammation (Multiprogramming) (1965-1970)

- Amélioration des coûts et des performances (circuits intégrés).
- Une famille d'ordinateurs compatibles entre eux.
- Une seule architecture et un même jeu d'instructions.
- Des ordinateurs uniques pour les calculs scientifiques et commerciaux.
- Apparition du spoule (spool, Simultaneous Peripheral Operation On Line) pour le transfert des travaux vers le disque.

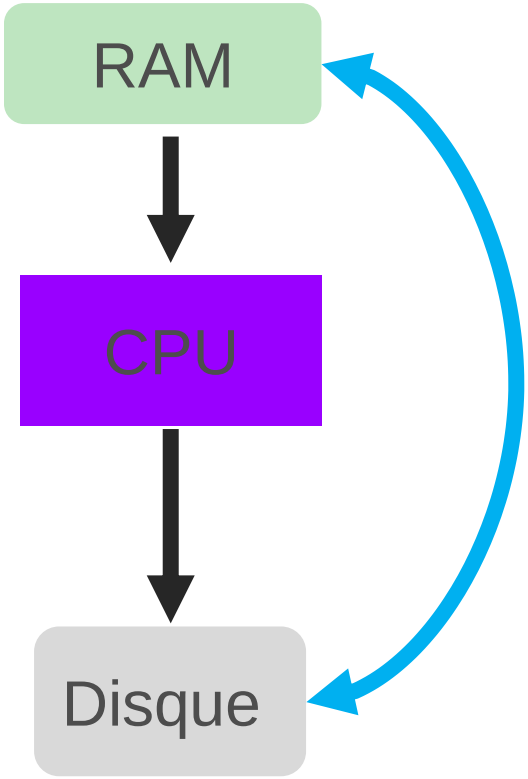
Multiprogrammation (Multiprogramming) (1965-1970)

- Les opérations E/S sont extrêmement lentes (comparé aux autres instructions)
- Même avec peu d'E/S, un programme passe la majorité de son temps à attendre
- Donc: pauvre utilisation de l'UCT lorsqu'un seul programme se trouve en mémoire



Multiprogrammation (Multiprogramming) (1965-1970)

Programme

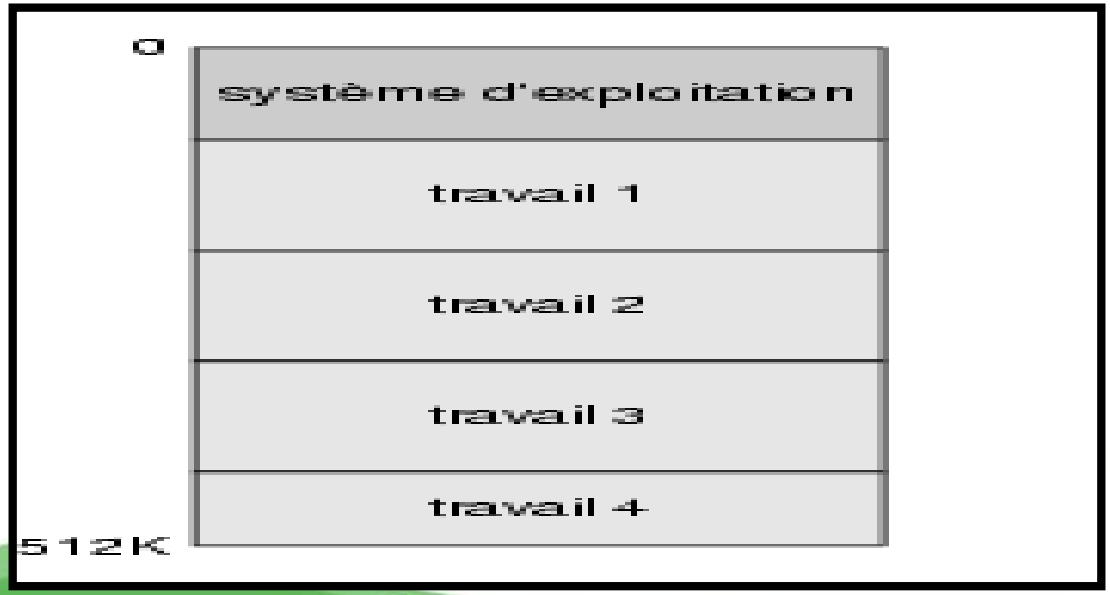


DMA

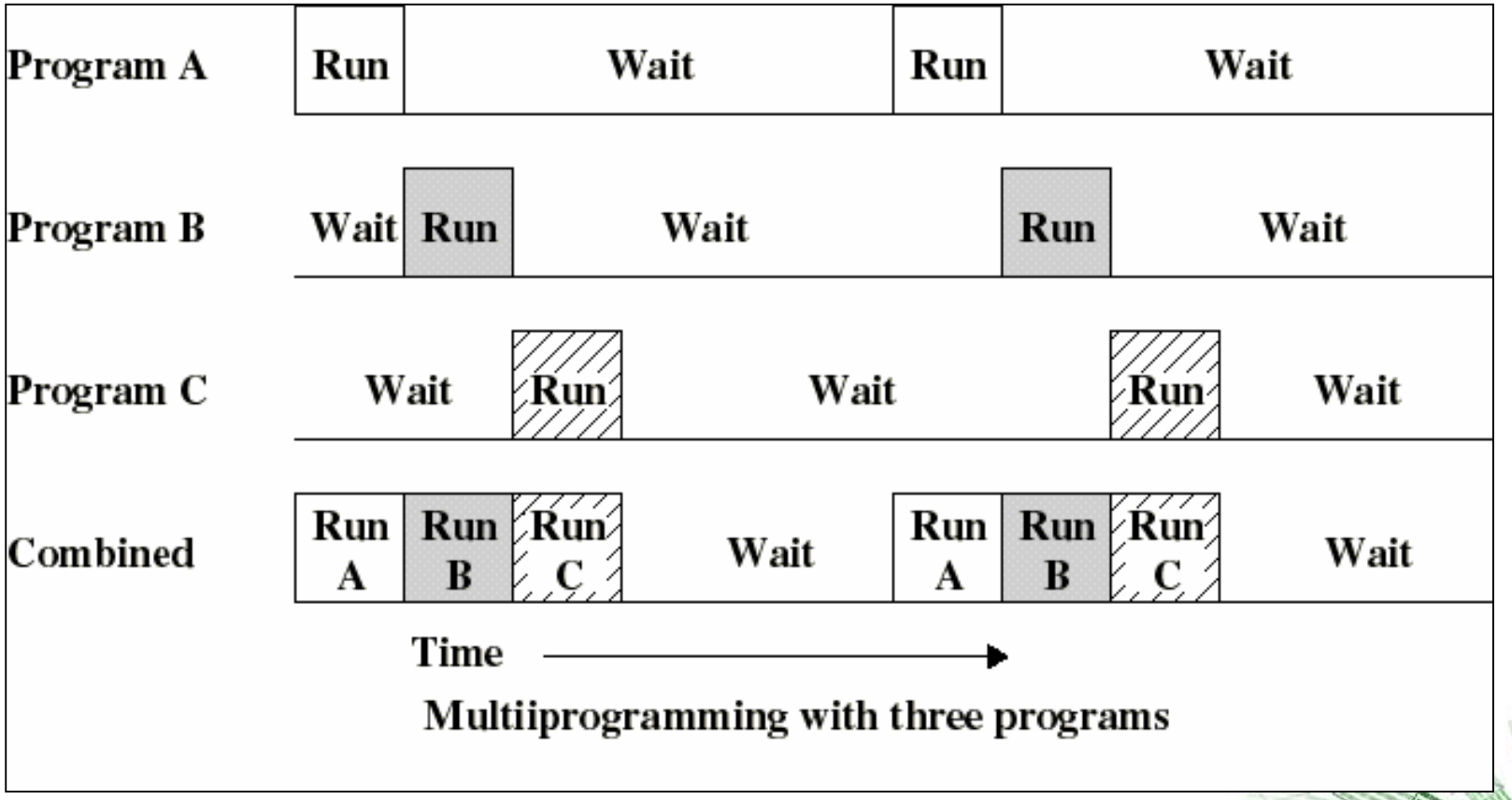
L'accès direct à la mémoire *(Direct Memory Access)*

Multiprogrammation (Multiprogramming) (1965-1970)

- Si la mémoire peut contenir plusieurs programmes, l'UCT peut exécuter un autre programme lorsqu'un programme attend une E/S



Multiprogrammation (Multiprogramming) (1965-1970)

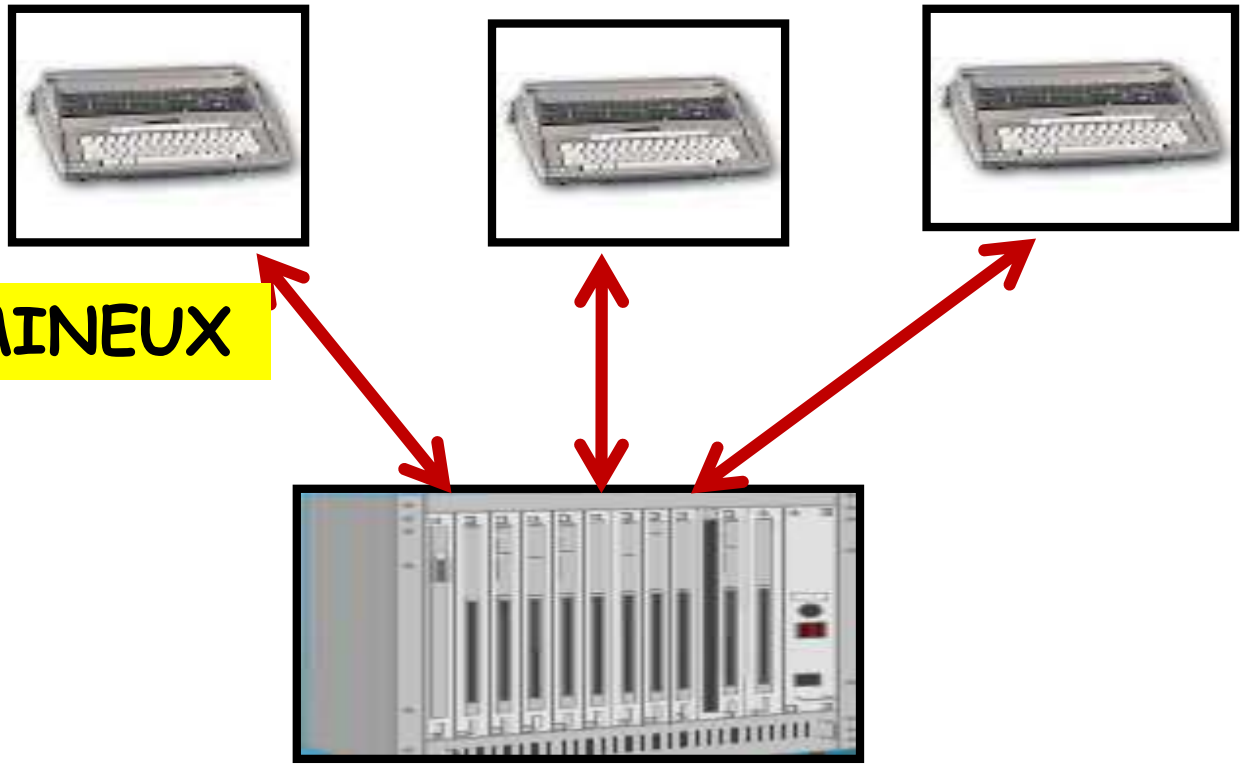


Multiiprogramming with three programs

Temps partagé (Time Sharing, 1970-)

C'est une variante du mode multiprogrammé où le temps CPU est distribué en petites tranches appelées quantum de temps

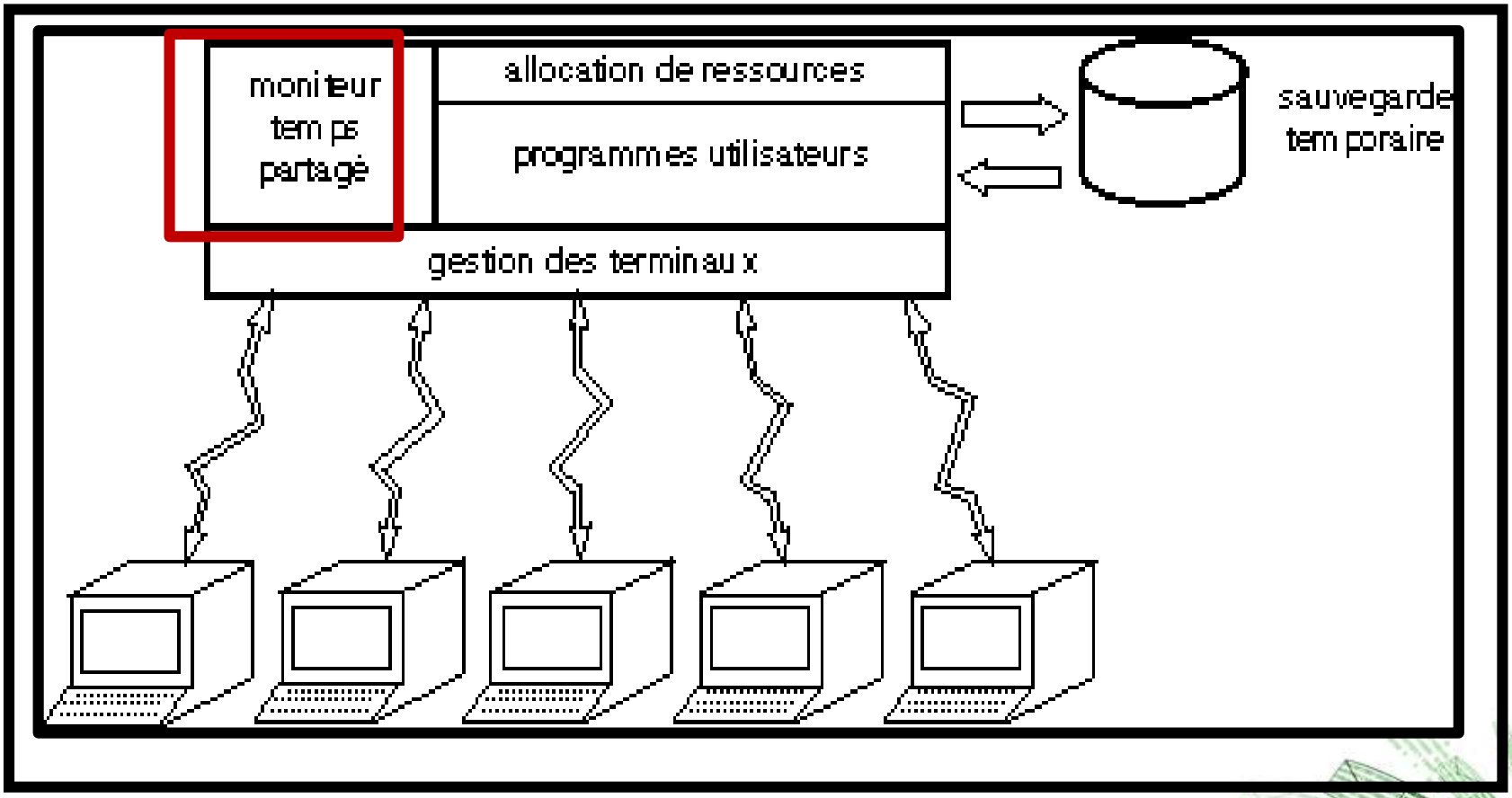
Systèmes à temps partagé



TERMINEUX

Ordinateur principal

Systèmes à temps partagé



Composants d'un système d'exploitation

plusieurs systèmes modernes possèdent les composants suivants :

- Gestion de processeurs.
- Gestion mémoire principale.
- Gestion mémoire auxiliaire.
- Gestion du système d'entrée/sortie.
- Gestion des fichiers.
- Système de protection.
- Gestion de réseaux.
- Système interpréteur de commande.

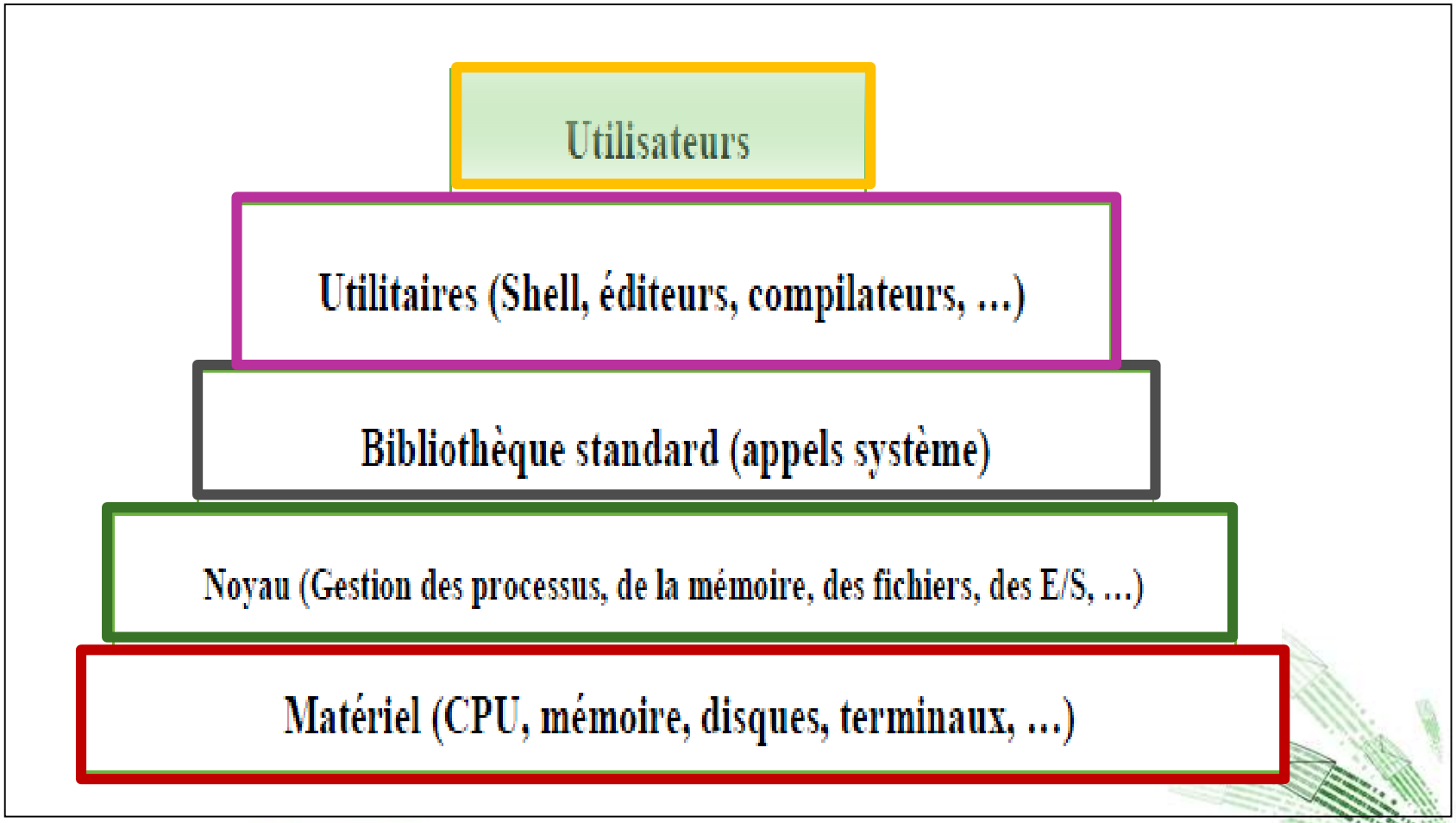
Classification des SE

- Système monotâche
- Multitâche
- Multi-utilisateurs
- Systèmes temps réel

Structuration des SE

La conception d'un SE est basée sur une structure à couches. Elle consiste à découper le SE en un certain nombre de couches (niveaux), chacune d'entre elles étant construite au-dessus des couches inférieures

Structuration des SE



Exemples de systèmes d'exploitation

- **MS-DOS: Microsoft Disk Operating system** (années 80, conçu par Bill Gates, associé aujourd'hui avec Windows.)
- **Le système Windows :** Il est passé par plusieurs versions comme pour PC : 1.0, 2.0, 3.10, 3.11, 9x, 2000, Me, XP, Vista, Seven, 10, aussi : Windows Server, NT pour l'architecture multi-utilisateur et client/serveur.

Exemples de systèmes d'exploitation

- **Le système UNIX** : Uniplexed Information and Computer Service (Une famille de systèmes en temps partagé proposée pour la plupart des architectures) C'est un système multitâche, multi-utilisateurs, multisessions, et multipostes.

Exemples de systèmes d'exploitation

Système	Codage	Mono-utilisateur	Multi-utilisateur	Mono-tâche	Multitâche
DOS	16 bits	X		X	
Windows3.1	16/32 bits	X			non préemptif
Windows95/98/Me	32 bits	X			coopératif
WindowsNT/2000	32 bits		X		préemptif
WindowsXP	32/64 bits		X		préemptif
Windows7	32/64 bits		X		préemptif
Unix / Linux	32/64 bits		X		préemptif
MAC/OS X	32 bits		X		préemptif
VMS	32 bits		X		préemptif

Chapitre 2

Mécanismes de base d'exécution des programmes

INTRODUCTION

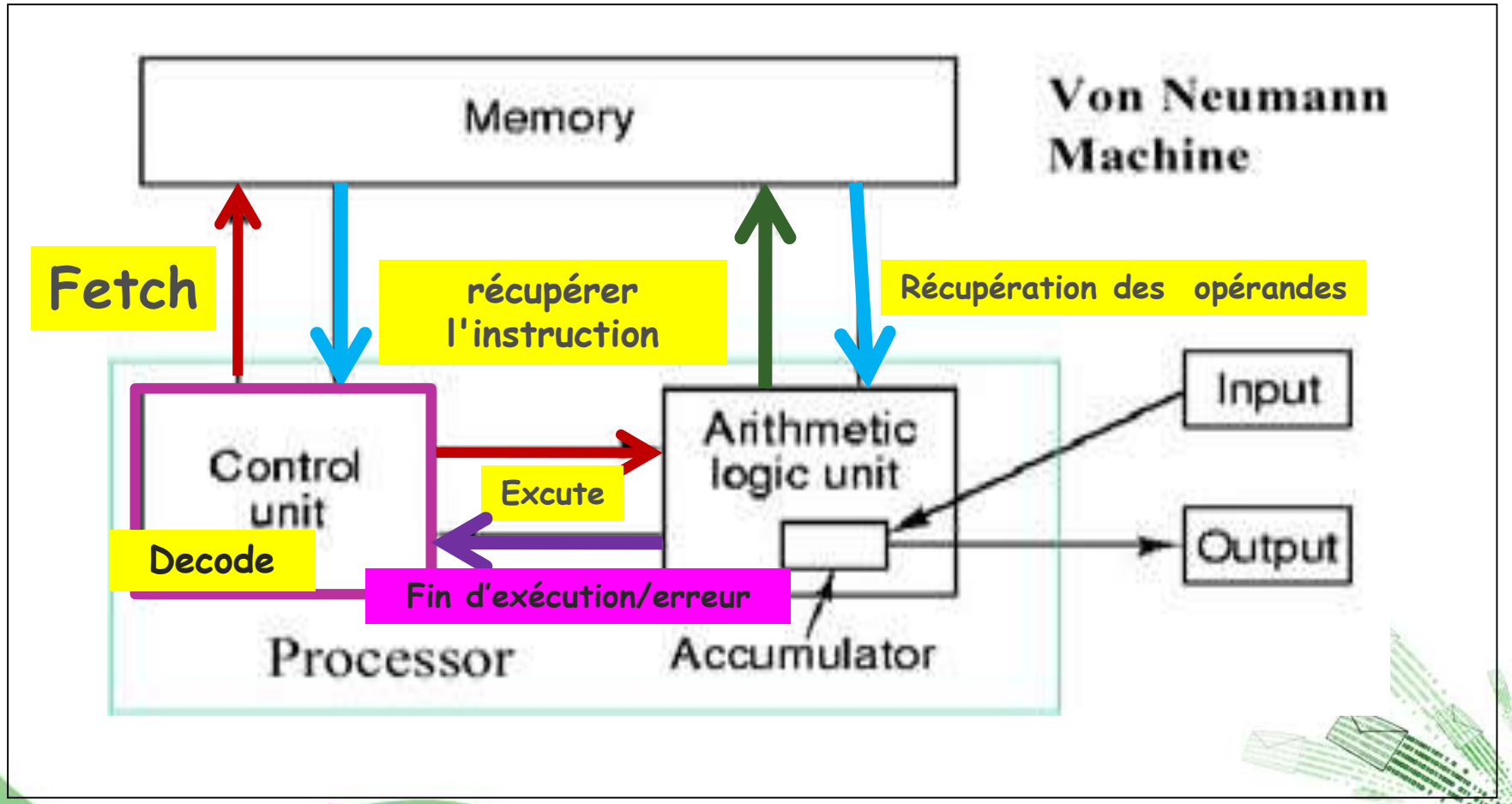
Dans cette partie du cours , nous étudierons les mécanismes de base utilisés par les systèmes d'exploitation pour réaliser leurs fonctions fondamentales .

Architecture matérielle d'une machine Von Neumann

John Von Neumann est à l'origine (1946) d'un modèle de machine universelle (non spécialisée) qui caractérise les machines possédant les éléments suivants :

- une mémoire contenant programme (instructions) et données,
- une unité arithmétique et logique (UAL ou ALU),
- une unité permettant l'échange d'information avec les périphériques : l'unité d'entrée/sortie (E/S ou I/O),
- une unité de commande (UC).

Architecture matérielle d'une machine Von Neumann



Les registres du processeur central

- Le Compteur Ordinal (CO)
- Le Registre d'Instruction (RI)
- Les Registres Généraux : Registres Accumulateurs , Registres d'Index , Registre de Base , Registres Banalisés
- Le Registre Pile (Stack Pointer, SP)
- Registre Mot d'Etat (PSW, Program Status Word)

Registre Mot d'Etat (PSW)

Il contient plusieurs types d'informations ; à savoir :

- **Les valeurs courantes des codes conditions (Flags)** qui sont des bits utilisés dans les opérations arithmétiques et comparaisons des opérandes.
- **Le mode d'exécution:** Pour des raisons de protection, l'exécution des instructions et l'accès aux ressources se fait suivant des modes d'exécution. deux modes d'exécution existent généralement
- **masque d'interruptions**

Registre Mot d'Etat (PSW)

- **Mode privilégié** ou **maître** (ou **superviseur**). Il permet : L'exécution de tout type d'instruction. Les instructions réservées au mode maître sont dites **privilégiées** (Ex. instructions d'E/S, protection, …etc.).L'accès illimité aux données.
- **Mode non privilégié ou esclave (ou usager).** Il permet : Exécution d'un répertoire limité des instructions.

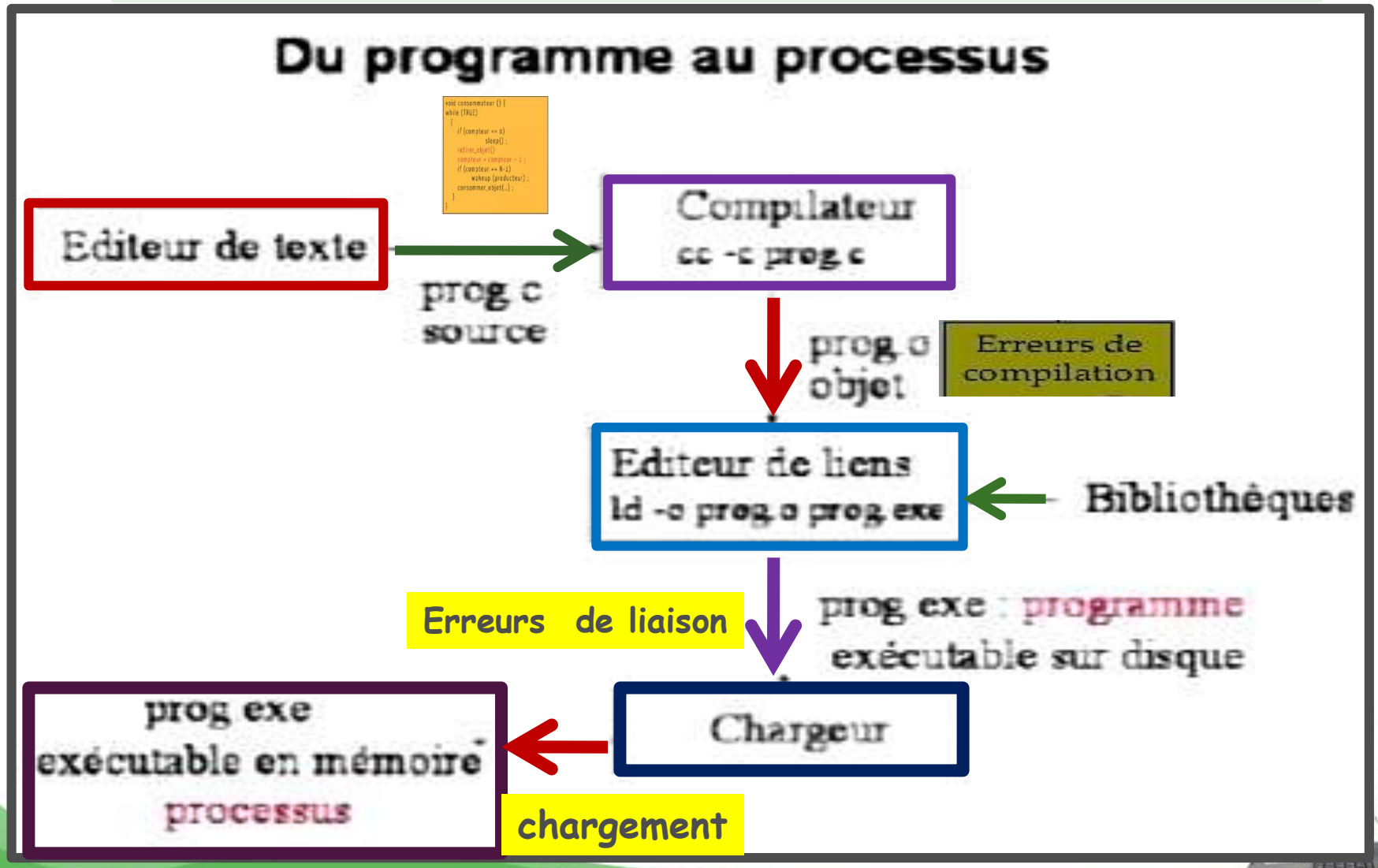
Etat du processeur (Processor State Information)

- Il est décrit par le **contenu** des **registres** du processeur. Le contenu des registres concerne le processus en cours d'exécution. Quand un processus est interrompu, l'information contenue dans les registres du processeur doit être sauvegardée afin qu'elle puisse être restaurée quand le processus interrompu reprend son exécution.
- **Remarque** :L'état d'un processeur n'est observable qu'entre deux cycles du processeur.

cheminement d'un programme dans un système

La chaine de production de programme transforme un programme écrit dans un langage de haut niveau en un programme dit exécutable, écrit en langage machine. Cette transformation s'effectue à l'aide de plusieurs étapes

cheminement d'un programme dans un système



le processus

Les processus correspondent à l'exécution de tâches : les programmes des utilisateurs, les entrées-sorties, … par le système. Un système d'exploitation doit en général traiter plusieurs tâches en même temps.

le processus

Un processus est une entité dynamique correspondant à l'exécution d'une suite d'instructions :un programme qui s'exécute, ainsi que ses données, sa **pile**, son **compteur ordinal**, son **pointeur de pile** et les autres contenus de registres nécessaires à son exécution.

le processus

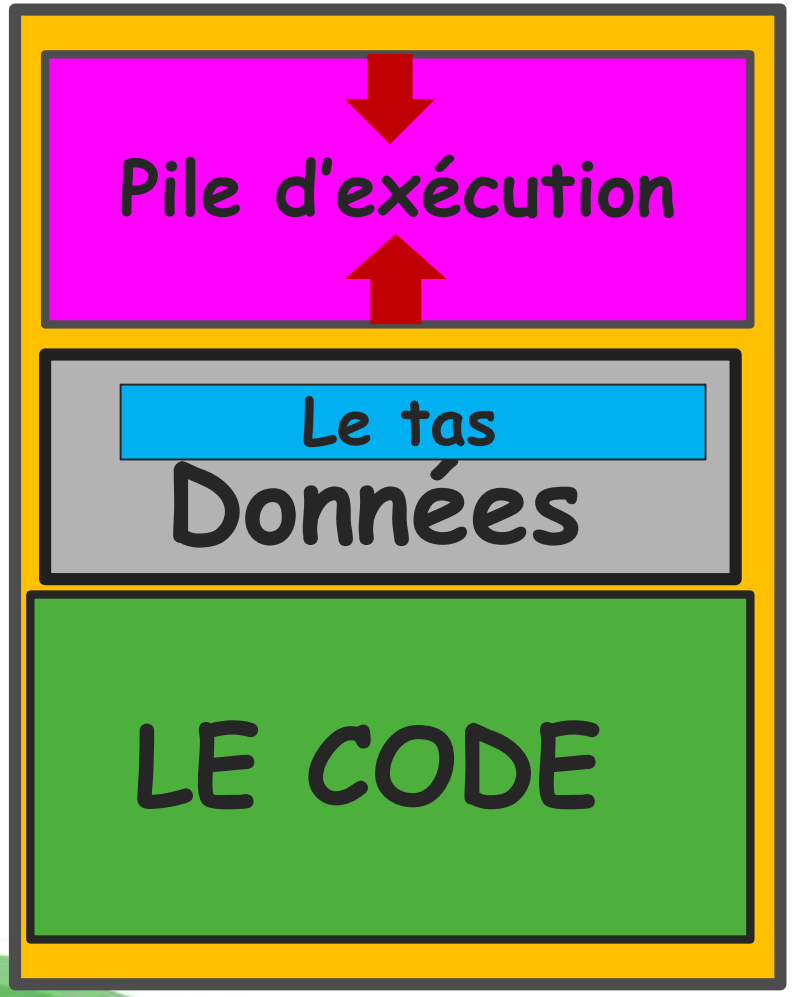
Le Rôle du SE pour les processus :

- **(Ré)Activer** un processus
- **Suspendre** un processus
- **Tuer** un processus
- **Contrôler l'exécution** d'un processus De manière optimale pour le CPU

le processus

Les segments d'un processus : Les processus sont composés d'un espace de travail en mémoire formé de 3 segments : **la pile**, **les données** et **le code** et d'un contexte.

le processus



les appels de fonctions, avec leurs paramètres et leurs variables locales

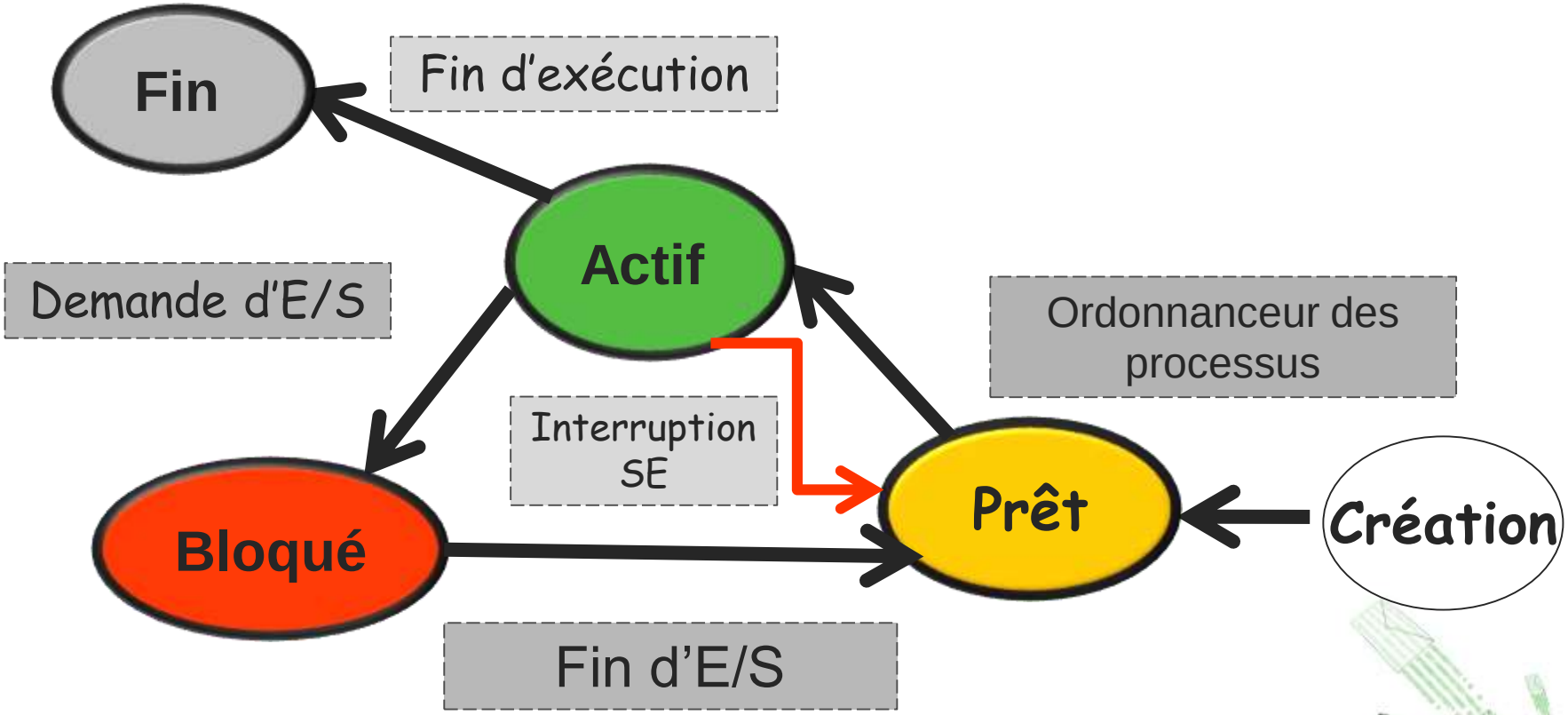
les variables globales ou statiques du programme

+

Les données dynamiques

Le code correspond aux **instructions**, en langage d'assemblage, du programme à exécuter.

Le Cycle de vie d'un processus



Le contexte d'un processus (PCB)

Le contexte est formé des données nécessaires à la gestion des processus. Une table contient la liste de tous les processus et chaque entrée conserve leur contexte. C'est L'ensemble des informations nécessaire et suffisante pour la reprise d'un processus au cas où il aurait été interrompu, Le contexte est utilisé :

- **En lecture lors de la restauration.**
- **En écriture lors de la sauvegarde**

Le contexte d'un processus (PCB)

- à tout processus, on associe un bloc de contrôle de processus (BCP).
- Il comprend généralement :
  - une copie du PSW au moment de la dernière interruption du processus
  - l'état du processus : prêt à être exécuté, en attente, suspendu, ...
  - des informations sur les ressources utilisées(mémoire principale

  ,temps d'exécution, périphériques d'E/S en attente)
  - files d'attente dans les quelles le processus est inclus, etc...
  - toutes les informations nécessaires pour assurer la reprise du processus en cas d'interruption

Le contexte d'un processus

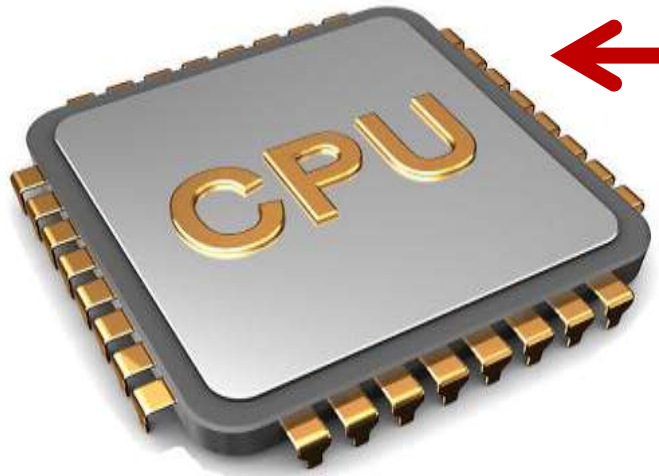
PSW: le mot d'état de processeur

PCB (processus actif)

CO	MODE	Mas IT	.....	.....

copier

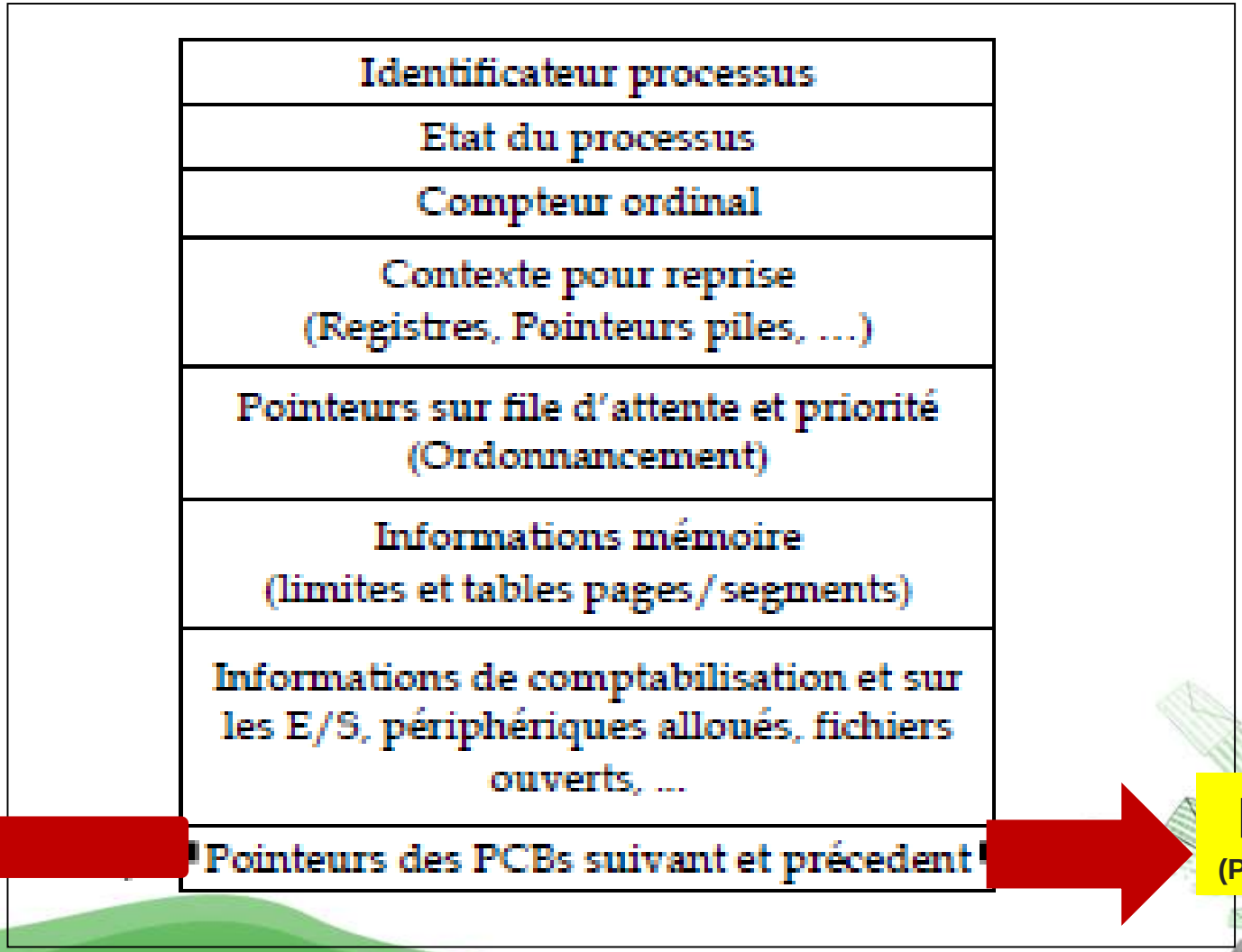
Interruption



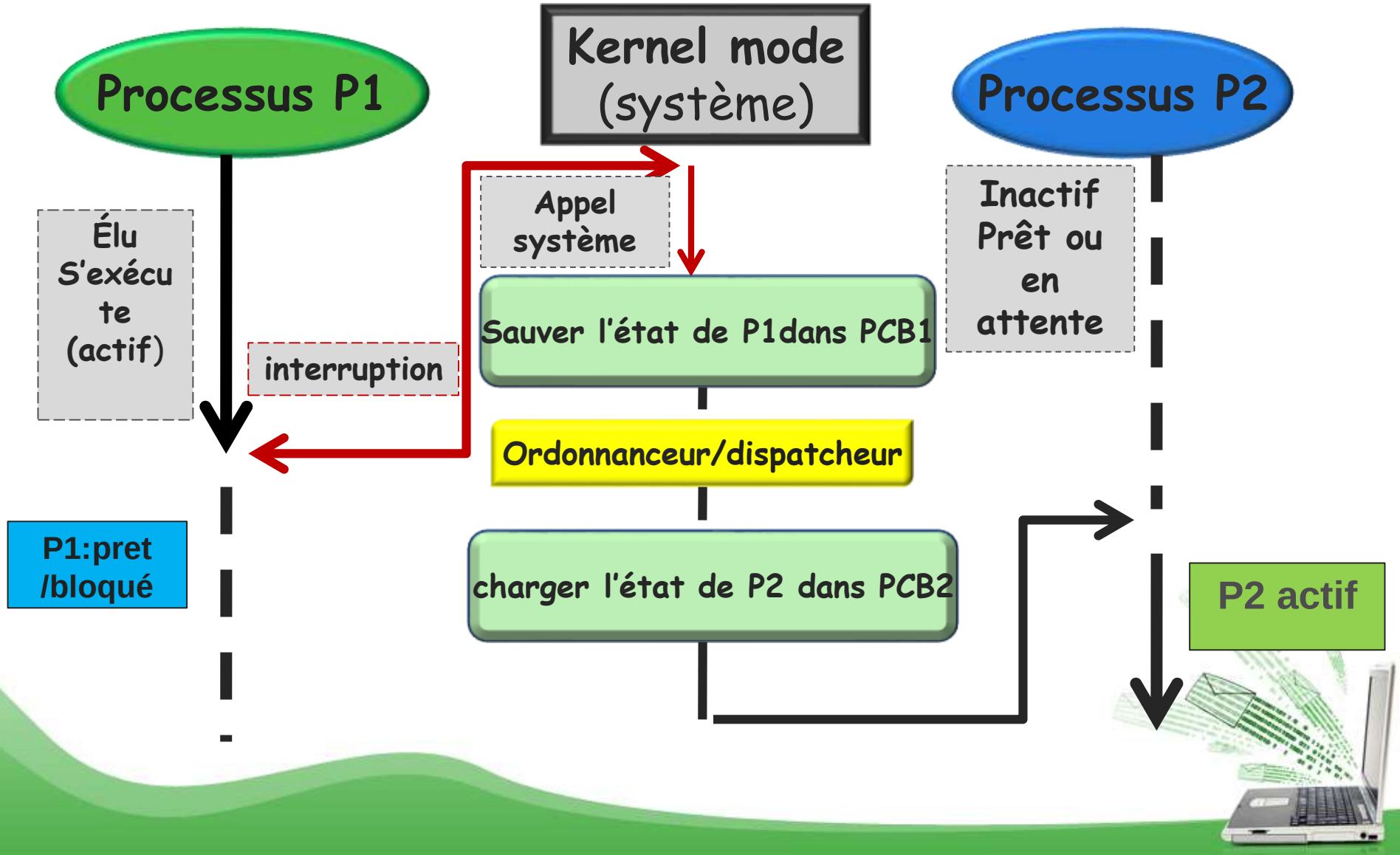
CO	mode			

PCB (processus)

Le contexte d'un processus (PCB)



Mécanisme de commutation de contexte

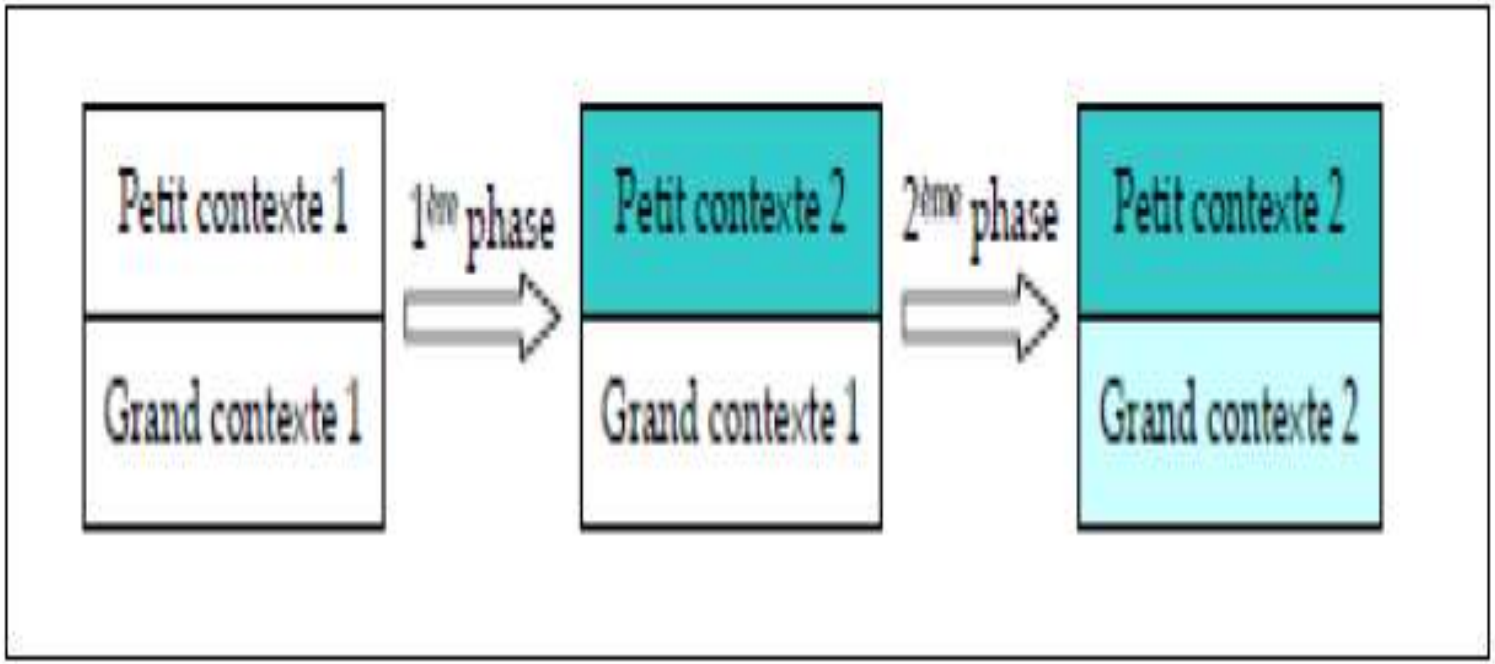


Mécanisme de commutation de contexte

La commutation du contexte se fait en deux phases :

- La première phase consiste à commuter le petit contexte (PSW) par une instruction indivisible.
- La deuxième phase consiste quant à elle à commuter le grand contexte par celui du nouveau processus.

Mécanisme de commutation de contexte



La cause d'une commutation de contexte

La commutation de contexte constitue la réponse du système à l'occurrence d'une **interruption** qui est un signal généré par le matériel comme conséquences à un événement qui s'est produit durant l'exécution du programme.

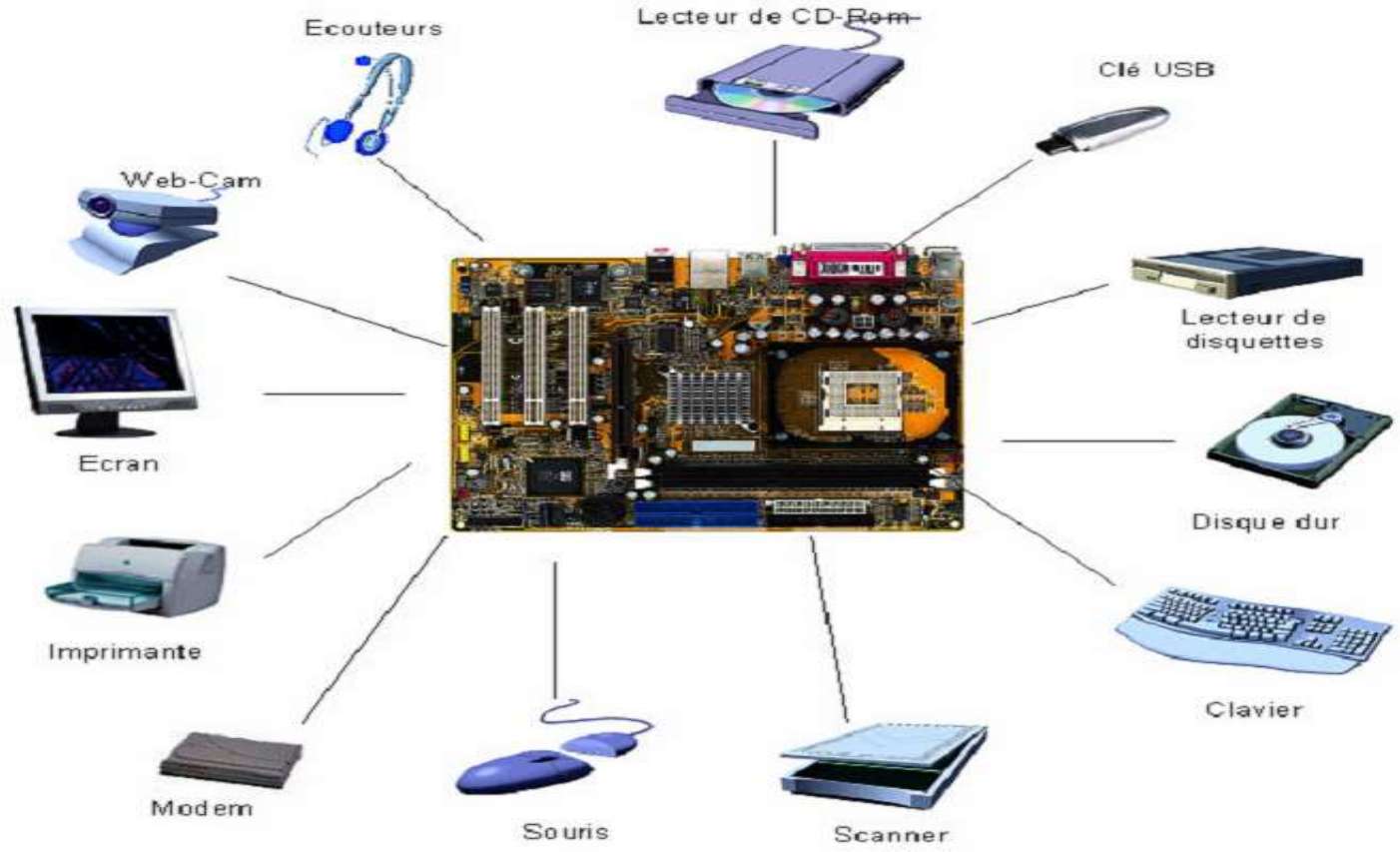
Les interruptions

Une interruption est un signal déclenché par un événement interne à la machine ou externe, provoquant l'arrêt d'un programme en cours d'exécution

Types d'interruptions

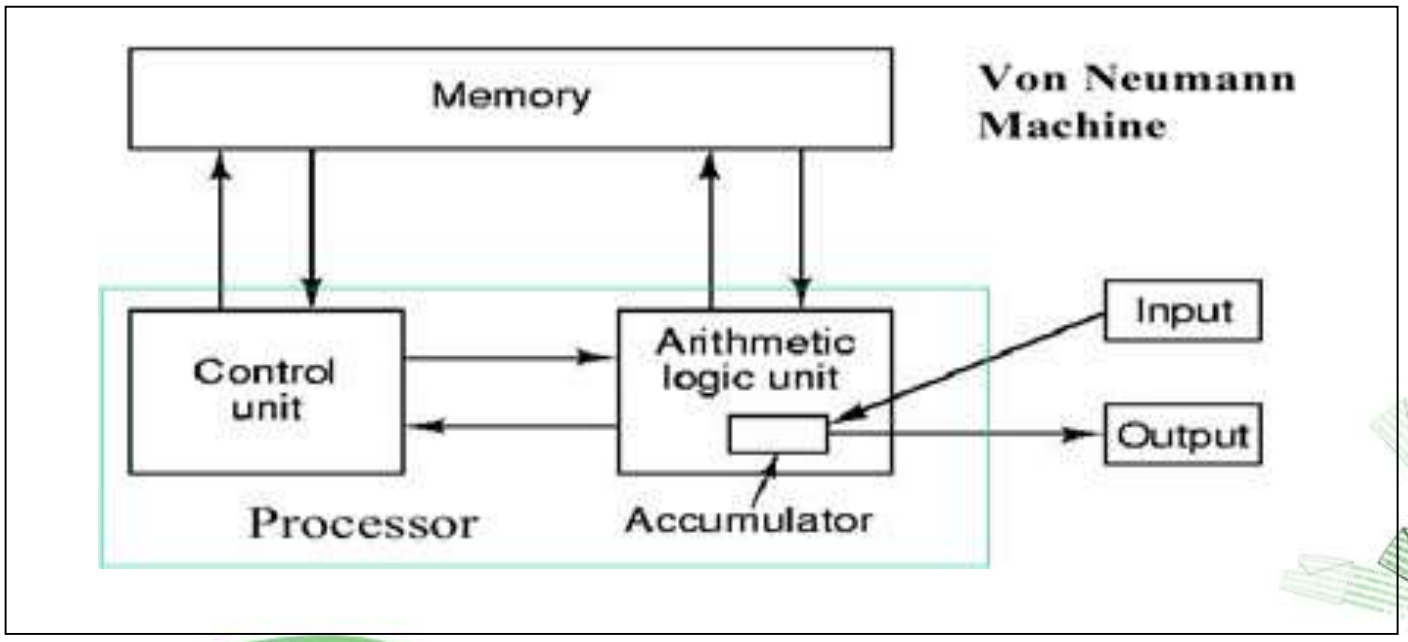
- interruptions matérielles ou externe
- interruptions internes ou déroutement
- interruptions logiques ou appel système (SVC)

Les interruptions matérielles ou externe



Interruptions internes ou déroutement

erreur d'adressage, code opération inexistant, problème de parité…)



Interruptions logiques ou appel système (SVC)

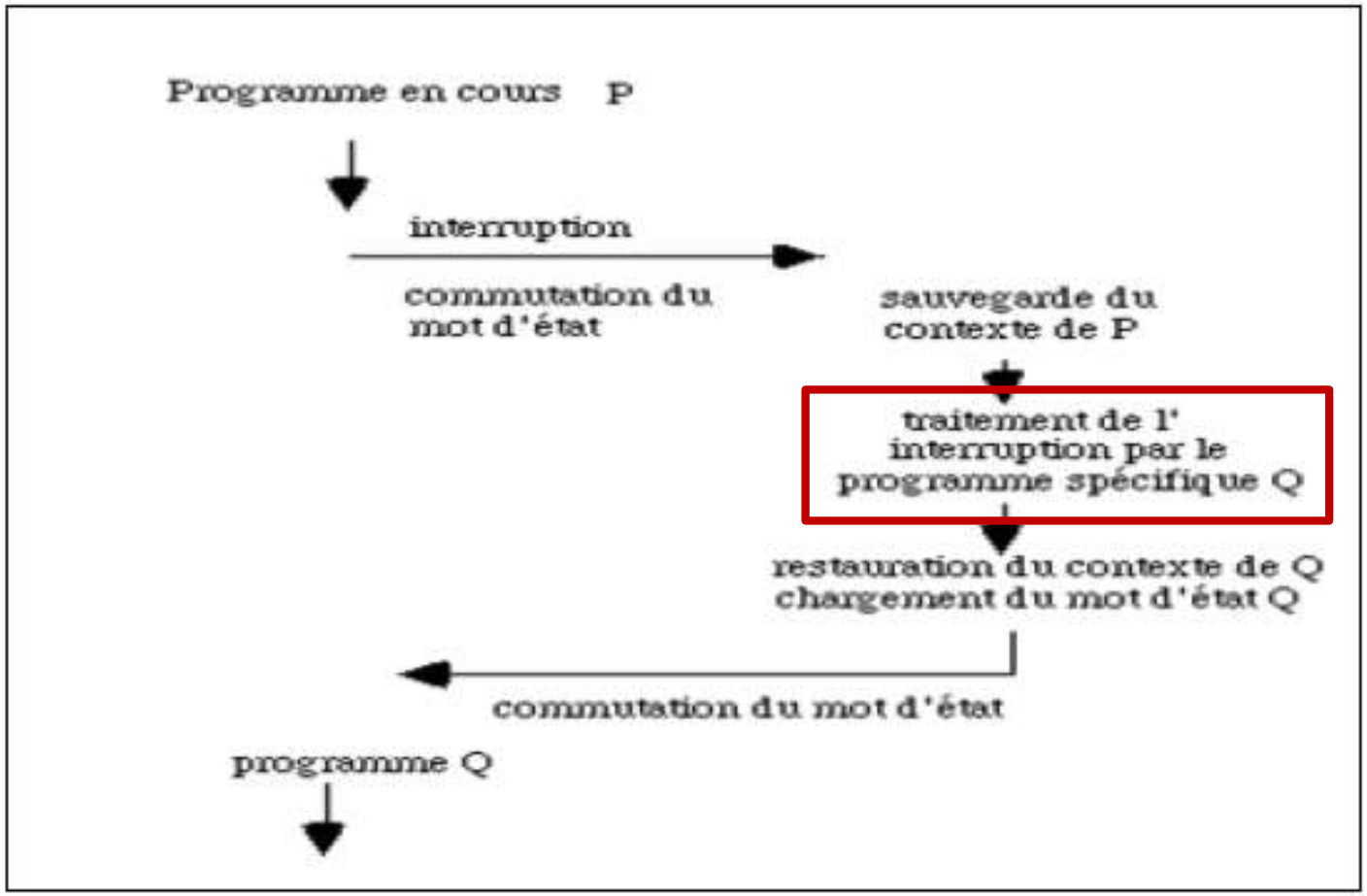
Processus

```
void consommateur () {
while (TRUE)
  {
    if (compteur == 0)
                sleep() ;
    retirer_objet()
    compteur = compteur - 1 ;
    if (compteur == N-1)
          wakeup (producteur) ;
    consommer_objet(..) ;
  }
}
```

SVC

Système d'exploitation

Le traitement d'une interruption

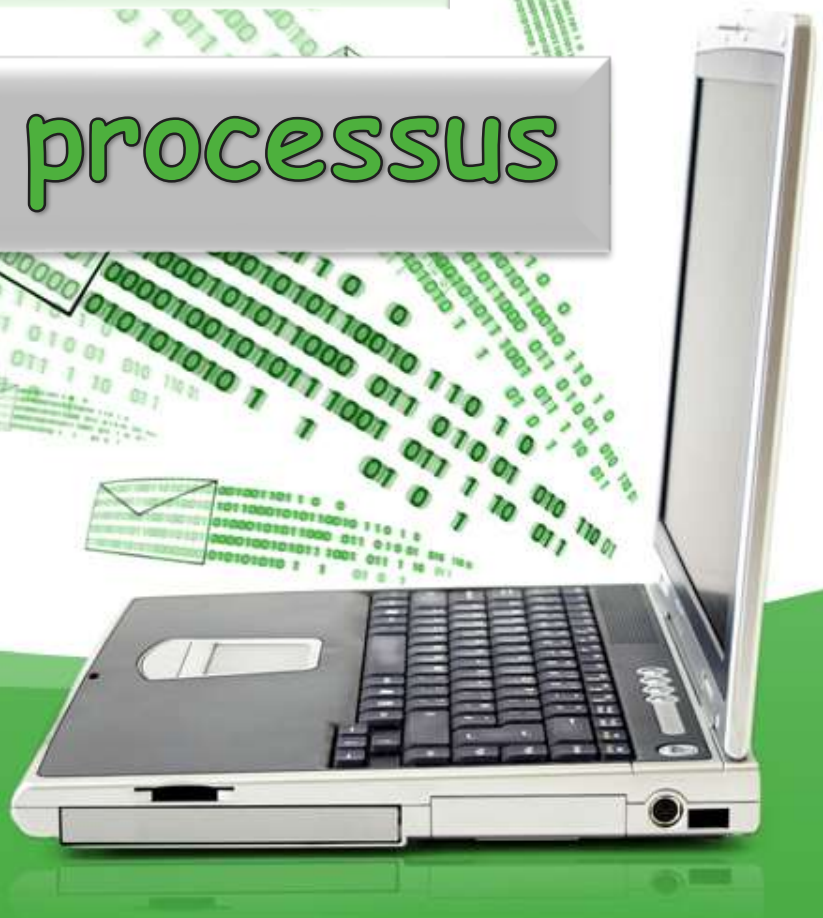


Les opérations sur les interruptions

- **Masquage des interruptions:** une interruption masquée n'est pas ignorée : elle est prise en compte dès qu'elle est démasquée.
- **Désarmer une interruption** :elle sera ignorée. Par défaut, les interruptions sont évidemment armées.

Chapitre 3

La Gestion des processus



Introduction

Le S.E choisit un processus qui deviendra actif parmi ceux qui sont prêts. Tout processus qui se bloque en attente d'un événement (par exemple l'attente de la frappe d'un caractère au clavier lors d'un scanf) passe dans l'état bloqué tant que l'événement attendu n'est pas arrivé. Lors de l'occurrence de cet événement, le processus passe dans l'état prêt. Il sera alors susceptible de se voir attribuer le processeur pour continuer ses activités.

Introduction

- Le changement d'état d'un processus peut être provoqué par :

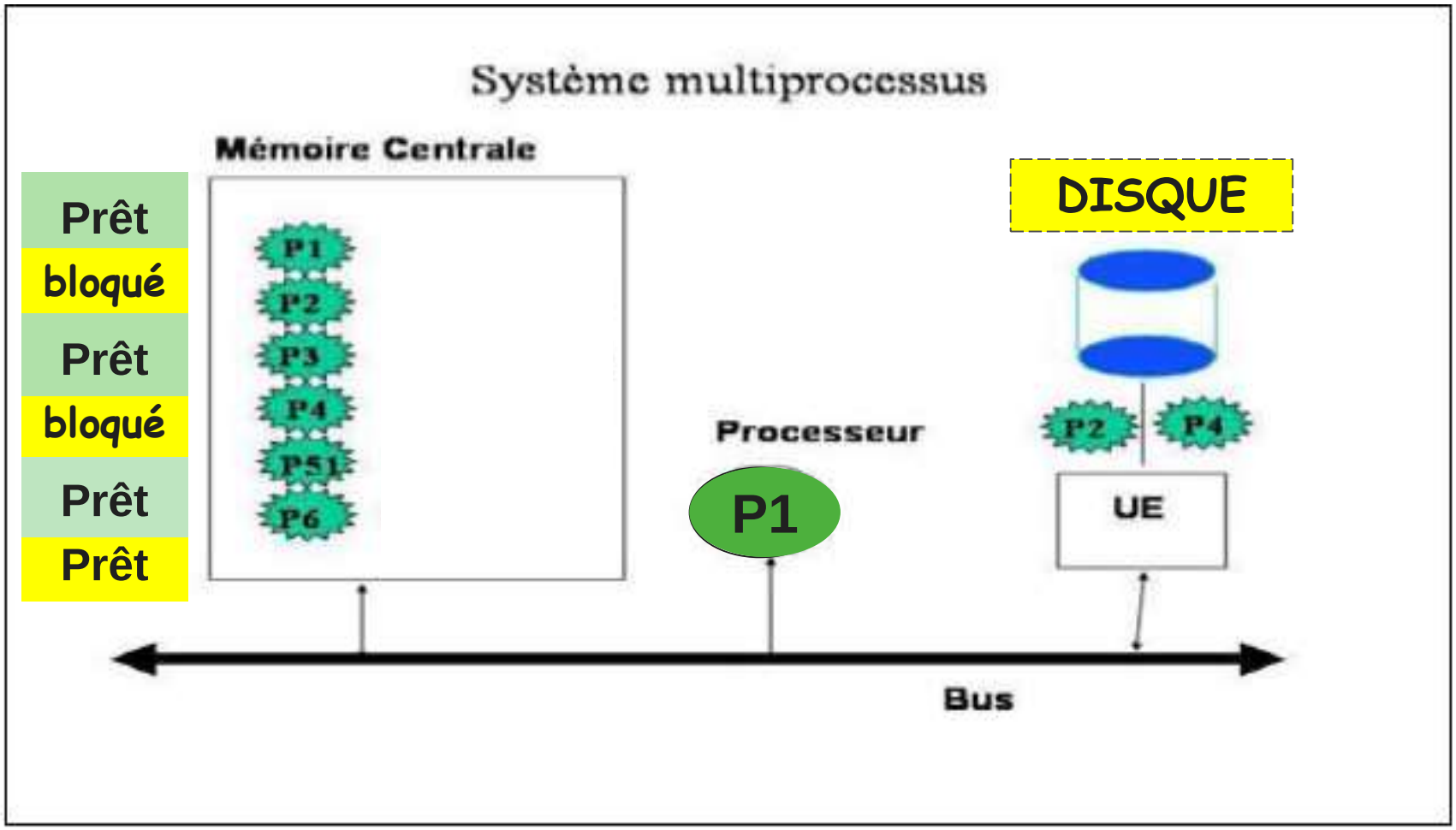
a) un autre processus (qui lui a envoyé un signal, par exemple)

b) le processus lui-même (appel à une fonction d'entrée-sortie bloquante,...)

c) une interruption (fin de quantum, terminaison d'entrée-sortie, ...)

- le passage de l'état actif à l'état prêt est provoqué par le système en fonction de sa politique **d'ordonnancement** (fin de quantum, préemption du processus actif si un processus plus prioritaire devient prêt)

Qu'est-ce que l'ordonnancement de processus ?



Qu'est-ce que l'ordonnancement de processus ?

Définition : La fonction d'ordonnancement gère le partage du processeur entre les différents processus en attente pour s'exécuter, c'est-à-dire entre les différents processus qui sont dans l'état prêt. L'opération d'élection consiste à allouer le processeur à un processus.

Qu'est-ce que l'ordonnancement de processus ?

D'une manière plus concrète, cette tâche est prise en charge par deux routines système en l'occurrence le ***Dispatcheur*** **et** ***L'ordonnanceur (Scheduleur).***

le Dispatcheur

Il s'occupe de l'allocation du processeur à un processus sélectionné par l'Ordonnanceur du processeur. Une fois allouer, le processeur doit réaliser les tâches suivantes :

A. **Commutation de contexte :** sauvegarder le contexte du processus qui doit relâcher le processeur
B. **Commutation du mode d'exécution** : basculer du mode Maître (mode d'exécution du dispatcheur) en mode utilisateur
C. **Branchement** : se brancher au bon emplacement dans le processus utilisateur pour le faire démarrer.

L'Ordonnanceur

deux types d'Ordonnanceurs :

a) **Ordonnanceur du processeur** : c'est un Ordonnanceur court terme opère sur une ensemble du processus présents en mémoire.

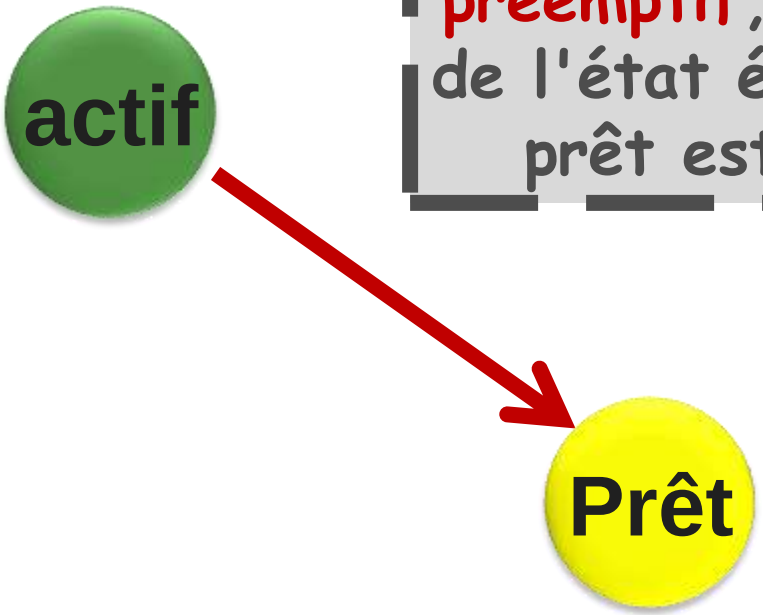
b) ***Ordonnanceur de travail*** : ou Ordonnanceur long terme, utilisé en cas d'insuffisance de **mémoire**, son rôle est de sélectionné le sous ensemble de processus stockés sur un disque et qui vont être chargés en mémoire.

Ordonnancement préemptif ou non préemptif

La préemption correspond à une opération de réquisition du processeur, c'est-à-dire que le processeur est retiré au processus élu alors que celui-ci dispose de toutes les ressources nécessaires à la poursuite de son exécution. Cette réquisition porte le nom de préemption.

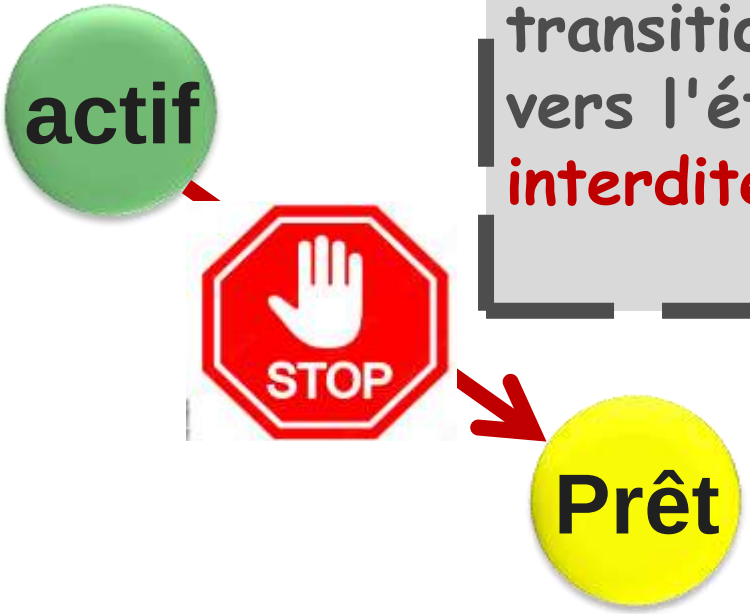
Ordonnancement préemptif

si l'ordonnancement est **préemptif**, la transition de l'état élu vers l'état prêt est **autorisée**



Ordonnancement non préemptif

si l'ordonnancement est **non préemptif**, la transition de l'état élu vers l'état prêt est **interdite**



Objectifs des politiques d'ordonnancement

- Systèmes de traitements par lots. Le but est de maximiser le débit du processeur ; c'est-à-dire le nombre moyen de processus traités par unité de temps.
- le taux d'occupation du processeur

Critères d'évaluation de performances

- **Le rendement d'un système** : est le nombre de processus exécutés par unité de temps.
- **Le temps de réponse** : est le temps qui s'écoule entre le moment où un processus devient prêt à s'exécuter et le moment où il finit de s'exécuter (temps d'accès mémoire + temps d'attente dans la file des processus éligibles + temps d'exécution dans l'unité centrale + temps d'attente + temps d'exécution dans les périphériques d'entrée/sortie).

Critères d'évaluation de performances

Le temps de réponse moyen décrit la moyenne des dates de fin d'exécution

$$TRM = \sum_{i=1}^{n} TRi \,/_{n} \text{ , avec } TR_i = \text{date fin} - \text{date arrivée.}$$

❑**Le temps d'attente** :est le temps passé dans la file des processus éligibles. On utilise en général un temps moyen d'attente calculé pour tous les processus mis en jeu pendant une période d'observation donnée.

$$TAM = \sum_{i=1}^{n} TAi \,/_{n} \text{ , avec } TA_i = TR_i - \text{temps d'exécution}$$

Ordonnancement non préemptif

- Ordonnancement FCFS (first come first Served)
- L'algorithme du Plus Court d'abord(SJF)

- Ordonnancement basé sur les priorités

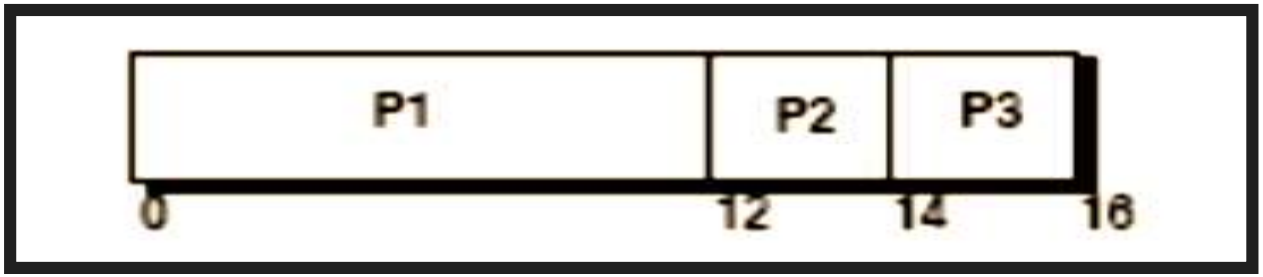
Ordonnancement FCFS

Dans cet algorithme ; connu sous le nom **FIFO** (First In, First Out) ; les processus sont rangés dans la file d'attente des processus prêts selon leur ordre d'arriver. Les règles régissant cet ordonnancement sont :

- Quand un processus est prêt à s'exécuter, il est mis en queue de la file d'attente des processus prêts.
- Quand le processeur devient libre, il est alloué au processus se trouvant en tête de file d'attente des processus prêts.

Ordonnancement FCFS

Exemple: On a trois processus 1, 2, 3 à exécuter de durées d'exécution respectives 12 , 2 et 2 unités de temps.



L'algorithme du Plus Court d'abord(SJF)

Cet algorithme (**en anglais Shortest Job First : SJF**) affecte le processeur au processus possédant le temps d'exécution le plus court. Si plusieurs processus ont la même durée, une politique FIFO sera alors utilisée pour les départager.

- **Problème** : comment prédire la durée d'exécution des processus a priori ?
- **Solutions :** faire de la prédiction sur la durée d'exécution des processus. faire accompagner la demande d'exécution de chaque programme d'une durée maximum d'exécution autorisée qui est utilisée comme durée d'exécution.

L'algorithme du Plus Court d'abord(SJF)

Exemple :On soumet au système quatre processus P1, P2, P3 et P4 dont les durées d'exécution sont données par le tableau suivant

Processus	Date d'arrivée	Temps CPU
P1	0	6
P2	2	3
P3	3	7
P4	3	4

Ordonnancement basé sur les priorités

Dans cet algorithme, les processus sont rangés dans la file d'attente des processus prêt par ordre décroissant de priorité. L'ordonnancement dans ce cas est régit par les règles suivantes :

A. Quand un processus est admis par le système, il est inséré dans la file d'attente des processus prêts à sa position approprie (dépend de la valeur de priorité).
B. Quand le processeur devient libre, il est alloué au processus se trouvant en tête de file d'attente des processus prêts (le plus prioritaire).
C. Un processus élu relâche le processeur s'il se termine ou se bloque (E/S ou autre).

Ordonnancement basé sur les priorités

Exemple: On dispose de 5 processus ayant des priorités différentes, comme le montre ce tableau :

processus	Durée d'exécution	Priorité
P1	10	2
P2	1	4
P3	2	2
P4	1	1
P5	5	3

Ordonnancement préemptif

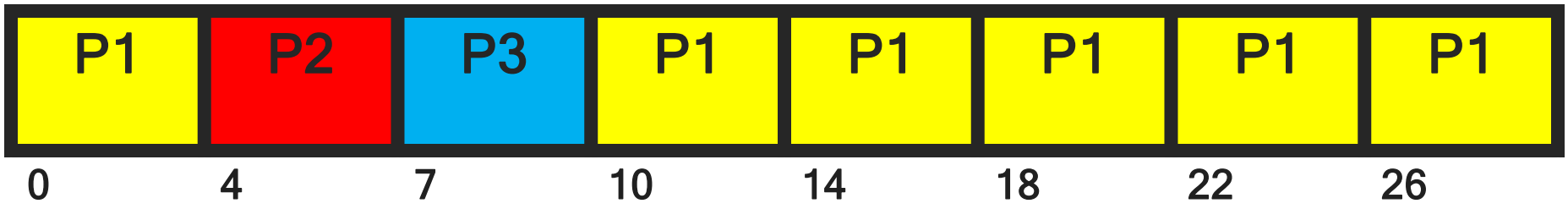
A. L'algorithme de Round Robin(Tourniquet)
B. Ordonnancement SRTF (plus court temps d'exécution restant PCTER) .
C. avec priorité (avec réquisition)

L'algorithme de Round Robin (RR) (Tourniquet)

L'algorithme tourniquet, appelé aussi Round Robin, a été conçu pour des systèmes à temps partagé. Il alloue le processeur aux processus à tour de rôle , pendant une tranche de temps appelée quantum. Dans la pratique le quantum s'étale entre 10 et 100 ms.

L'algorithme de Round Robin (RR) (Tourniquet)

Exemple : On dispose de 3 processus P1, P2 et P3 ayant comme durée d'exécutions, respectivement **24**, **3** et **3** ms. En utilisant un algorithme Round Robin, avec un quantum de4 ms, on obtient le diagramme de Gantt suivant :

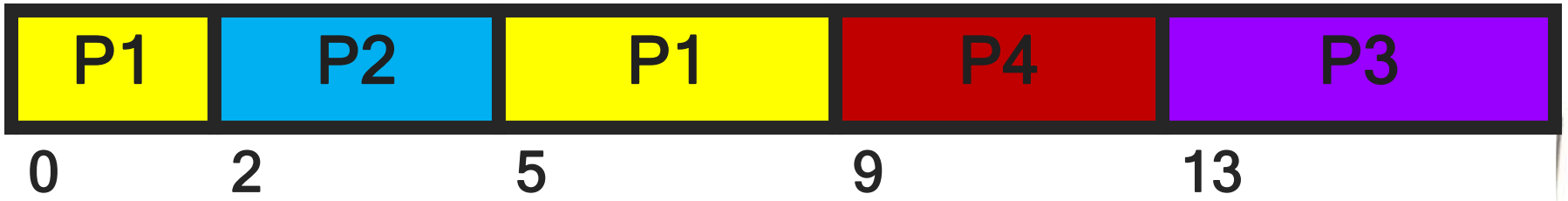


Ordonnancement SRTF

choisit le processus dont le temps d'exécution restant est le plus court. C'est une variante de l'algorithme SJF Cet algorithme est non implantable parce qu'il suppose, entre autres, connu le temps d'exécution réel de chaque processus pour pouvoir calculer le temps restant.

Ordonnancement SRTF

Processus	Date d'arrivée	Temps CPU
P1	0	6
P2	2	3
P3	3	7
P4	3	4



L'ordonnancement avec priorité (avec réquisition)

On associe une priorité à chaque processus et on choisit dans la file des processus prêts le processus qui a la priorité la plus haute. Si 2 processus ont la priorité la plus haute égale, on choisit le 1er dans la file d'attente. Lorsqu'un processus arrive dans l'état prêt, le système d'exploitation regarde si la priorité de ce processus est strictement supérieure à celui du processus en exécution.

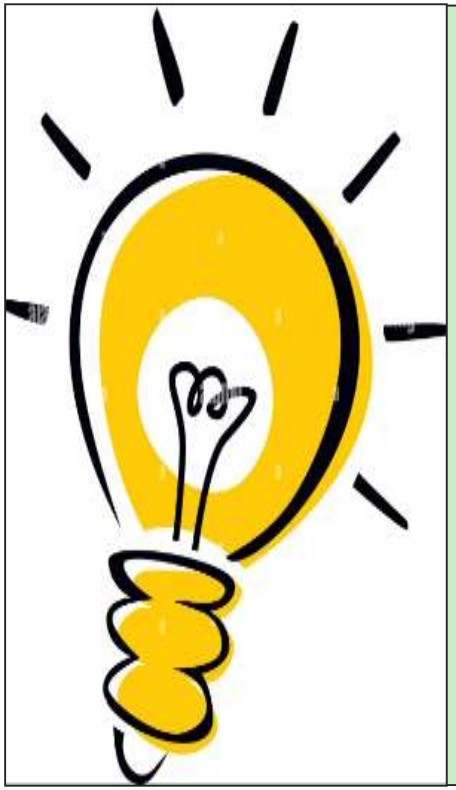
L'ordonnancement avec priorité (avec réquisition)

Processus	Date d'arrivée	Temps CPU	Priorité
P1	0	6	2
P2	2	3	4
P3	3	7	3
P4	3	4	2

Priorités dynamiques

Si le système est très chargé, les processus de faible priorité n'obtiennent jamais le CPU(**famine** ou starvation), pour pallier à ce défaut, la priorité doit être un paramètre dynamique qui permettra de changer le processus de file (files réactives),l'ajustement dynamique de la priorité d'un processus est calculé par l'ordonnanceur.

Priorités dynamiques



- Recyclage avec priorité variable
- Les files multi niveaux
- Les files multi niveaux avec feedback

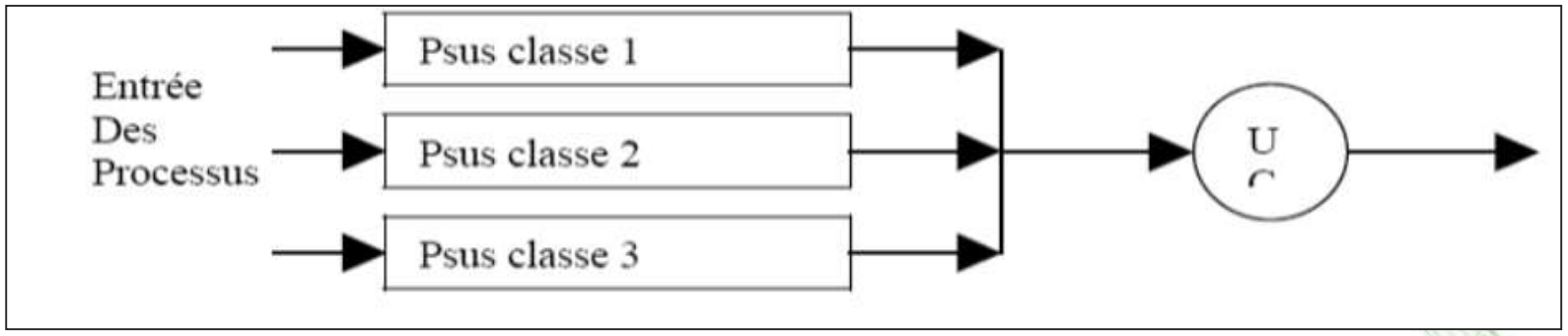
Recyclage avec priorité variable

Les priorités sont calculées de deux façons différentes:

- Si un processus est arrêté par un événement E (entrée-sortie, fonctions wait, pause ...), il est réactivé avec une priorité qui dépend du type de cet événement.
- La priorité des processus prêts est directement liée à la consommation de temps CPU et au temps qui s'est écoulé depuis leur dernier accès au processeur.

Les files multi niveaux

Les processus sont regroupés par classe, et à chacune est associée une priorité et politique d'allocation.



Les files multi niveaux avec feedback

Plusieurs queues (politiques) les processus peuvent passer d'une queue à une autre processus prêt sont gérer dans plusieurs files de priorités, et il est possible de faire passer un processus d'une file à une autre.

Chapitre 4

La gestion de la mémoire

Introduction

La mémoire principale est le lieu où se trouvent les programmes et les données quand le processeur les exécute. On l'oppose au concept de mémoire secondaire, représentée par les disques, de plus grande capacité, où les processus peuvent séjourner avant d'être exécutés. La nécessité de gérer la mémoire de manière optimale est toujours fondamentale, car en dépit de sa grande disponibilité, elle n'est, en général, jamais suffisante. Nous verrons qu'il y a plusieurs schémas pour la gestion de la mémoire, avec leurs avantages et inconvénients.

La Mémoire centrale

Le terme mémoire désigne un composant destiné à contenir une certaine quantité de données, et à en permettre la consultation.

La mémoire centrale, dite aussi la mémoire vive (**Random Access Memory, RAM**), L'octet représente la plus petite quantité de mémoire adressable. Plusieurs types de mémoires sont utilisés, différentiables par leur technologie (DRAM, SRAM, ...etc.), et leur forme (SIMM, DIMM, ...etc.).Il s'agit d'une mémoire volatile

La Mémoire centrale

Deux types d'opérations peuvent s'effectuer sur les mots mémoires ; à savoir la **lecture (Read)** et **l'écriture (Write)** :

- **Pour lire la mémoire**: une adresse doit être choisie à partir du bus d'adresse et le bus de contrôle doit déclencher l'opération. Les données à l'adresse choisie se retrouvent sur le bus de données.
- **Pour écrire la mémoire**: Les données à l'adresse choisie sont remplacées par celles sur le bus de données.

Allocation de la mémoire centrale et multiprogrammation

On définit le **degré de multiprogrammation** comme étant le nombre de processus présents en mémoire centrale. trois problèmes sont à résoudre vis-à-vis de la mémoire centrale :

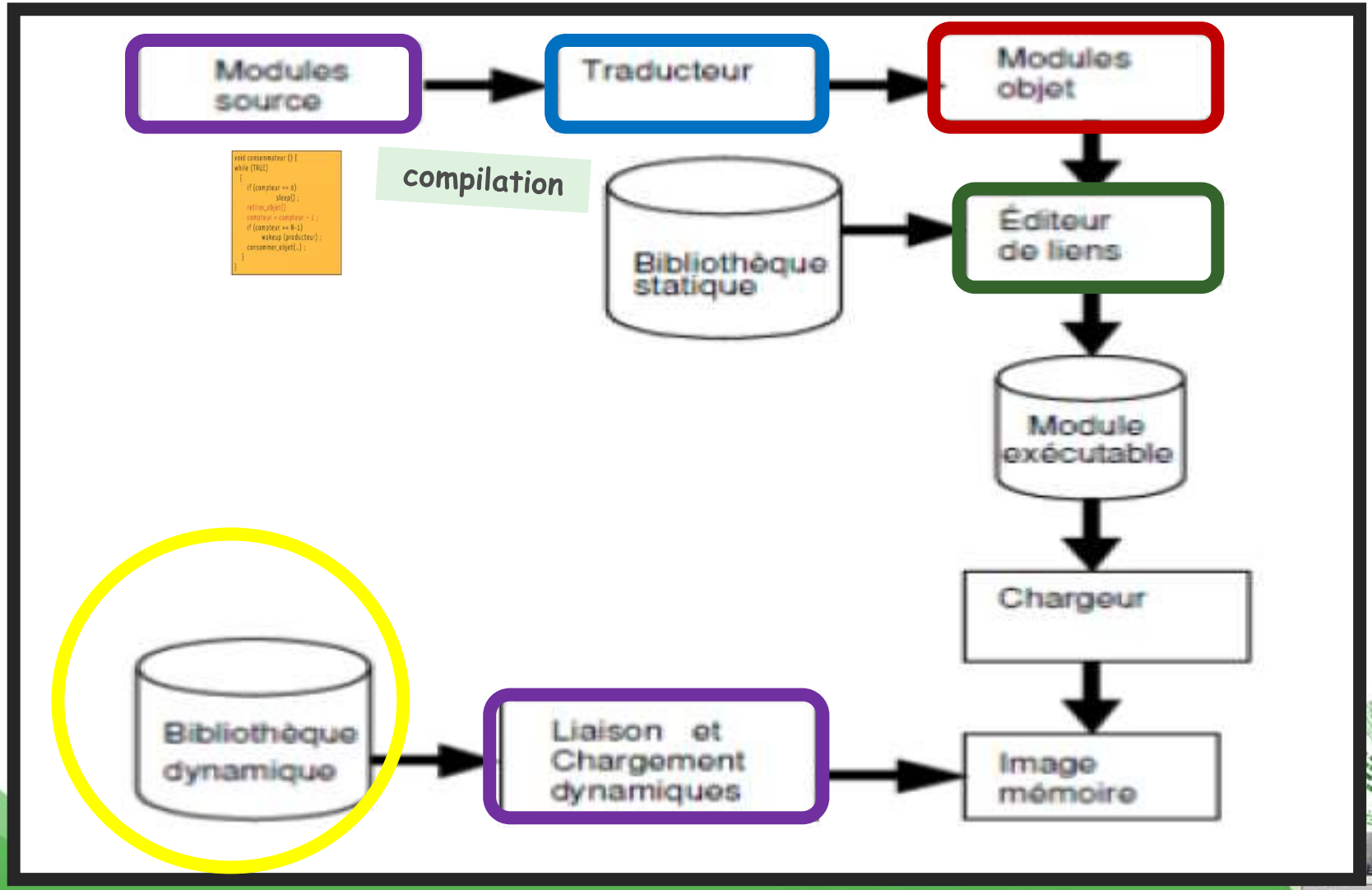
- Il faut définir un espace d'adressage indépendant pour chaque processus.
- Il faut protéger les espaces d'adressage des processus les uns vis-à-vis des autres
- Il faut allouer de la mémoire physique à chaque espace d'adressage.

Différentes méthodes d'allocation mémoire

Allocation contiguë : pour la première famille, un programme est un ensemble de mots contigus insécable. L'espace d'adressage du processus est linéaire.

Allocation non contiguë : pour la seconde famille, un programme est un ensemble de mots contigus sécable, c'est-à-dire que le programme peut être divisé en plus petits morceaux, chaque morceau étant lui-même un ensemble de mots contigus. Chaque morceau peut alors être alloué de manière indépendante.

Caractéristiques liées au chargement d'un programme



Caractéristiques liées au chargement d'un programme

Type d'adresse :

- **adresses symboliques** : au niveau de Code source Par exemple : int compteur
- **adresses traduites** : pour le Module objet : Par exemple le 50ème mot depuis le début d'espace mémoire.
- **adresses absolues** : Module exécutable, chargement : Par exemple l'emplacement mémoire situé à l'adresse 7777

Espace d'adressage logique ou physique

- **L'espace d'adressage logique** : L'unité centrale manipule des adresses logiques (emplacement relatif).Les programmes ne connaissent que des adresses logiques, ou virtuelles. L'espace d'adressage logique (virtuel) est donc un ensemble d'adresses pouvant être générées par un programme.

L'espace d'adressage physique : L'unité mémoire manipule des adresses physiques (emplacement mémoire). Elles ne sont jamais vues par les programmes utilisateurs. L'espace d'adressage physique est un ensemble d'adresses physiques correspondant à un espace d'adresses logiques.

Le passage d'une adresse logique vers une adresse physique

- **Pendant la compilation** : Si l'emplacement du processus en mémoire est connu, le compilateur génère des adresses absolues.
- **Pendant le chargement** : après la compilation un Code translatable si l'emplacement du processus en mémoire n'est pas connu.
- **Pendant l'exécution** : déplacement dynamique du processus possible. Utilise des fonctions du matériel.

La gestion de la mémoire

La gestion de la mémoire a deux objectifs principaux :

- d'abord le partage de mémoire physique entre les programmes et les données des processus prêts
- ensuite la mise en place des paramètres de calcul d'adresses, permettant de transformer une adresse virtuelle en adresse physique.

La gestion de la mémoire

Pour ce faire le gestionnaire de la mémoire doit remplir plusieurs tâches :

- Connaître l'état de la mémoire (les parties libres et occupées de la mémoire).
- Allouer de la mémoire à un processus avant son exécution.
- Récupérer l'espace alloué à un processus lorsque celui-ci se termine.

La gestion de la mémoire

- Traiter le va-et-vient (swapping) entre le disque et la mémoire principale lorsque cette dernière ne peut pas contenir tous les processus.
- Posséder un mécanisme de calcul de l'adresse physique (absolue) car,en général, les adresses générées par les compilateurs et les éditeurs de liens sont des adresses relatives ou virtuelles.

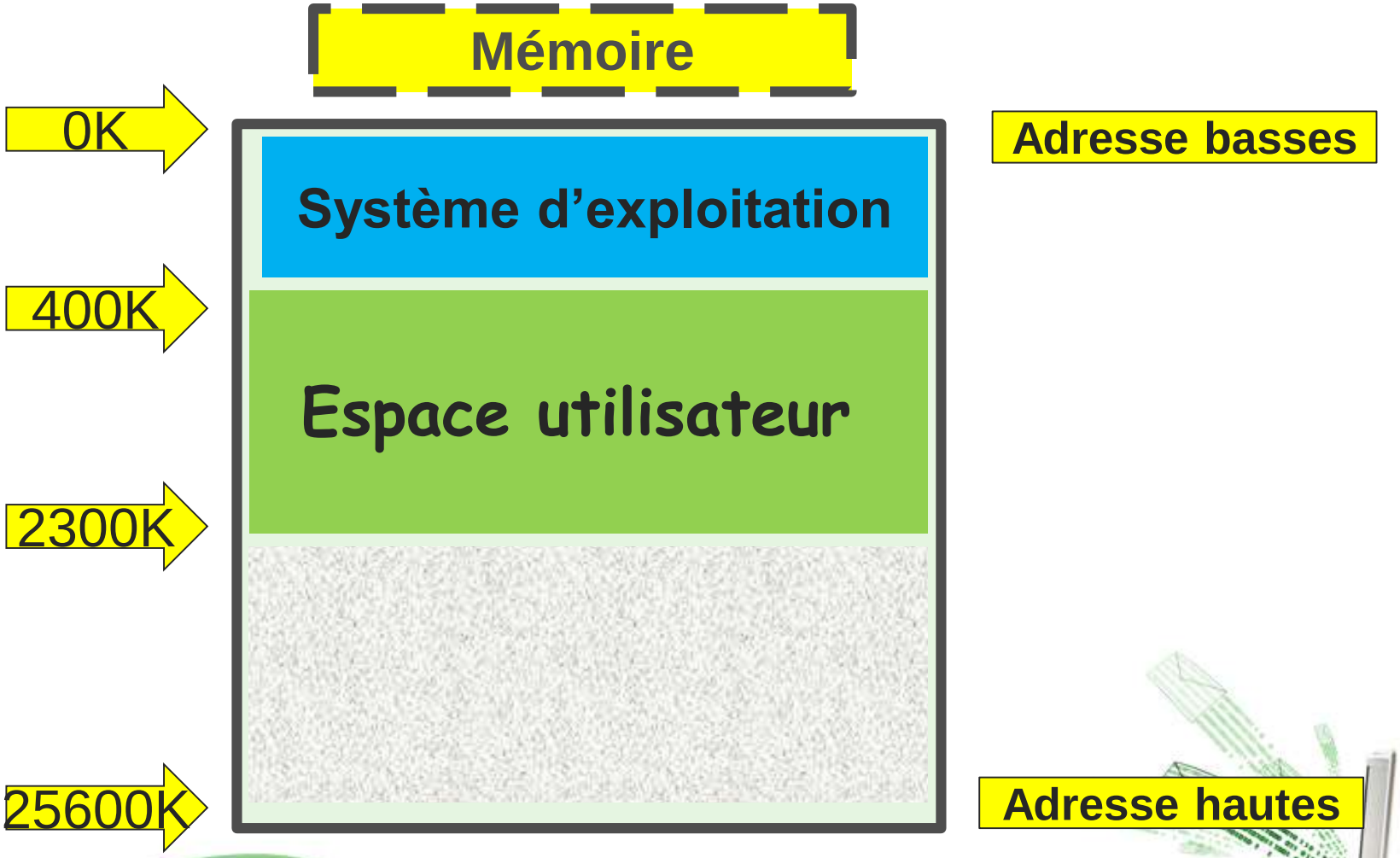
Allocation contiguë

- **Monobloc** (Single Contiguous Store Allocation)
- **Partitions multiples** (Multiple-Partition Allocation)

Monobloc

Dans ce cas, la mémoire est subdivisée en deux partitions contiguës, une pour le système d'exploitation résident souvent placer en mémoire basse avec le vecteur d'interruptions et l'autre pour le processus utilisateur.

Monobloc


Mémoire
0K
400K
2300K
25600K
Système d'exploitation
Espace utilisateur
Adresse basses
Adresse hautes

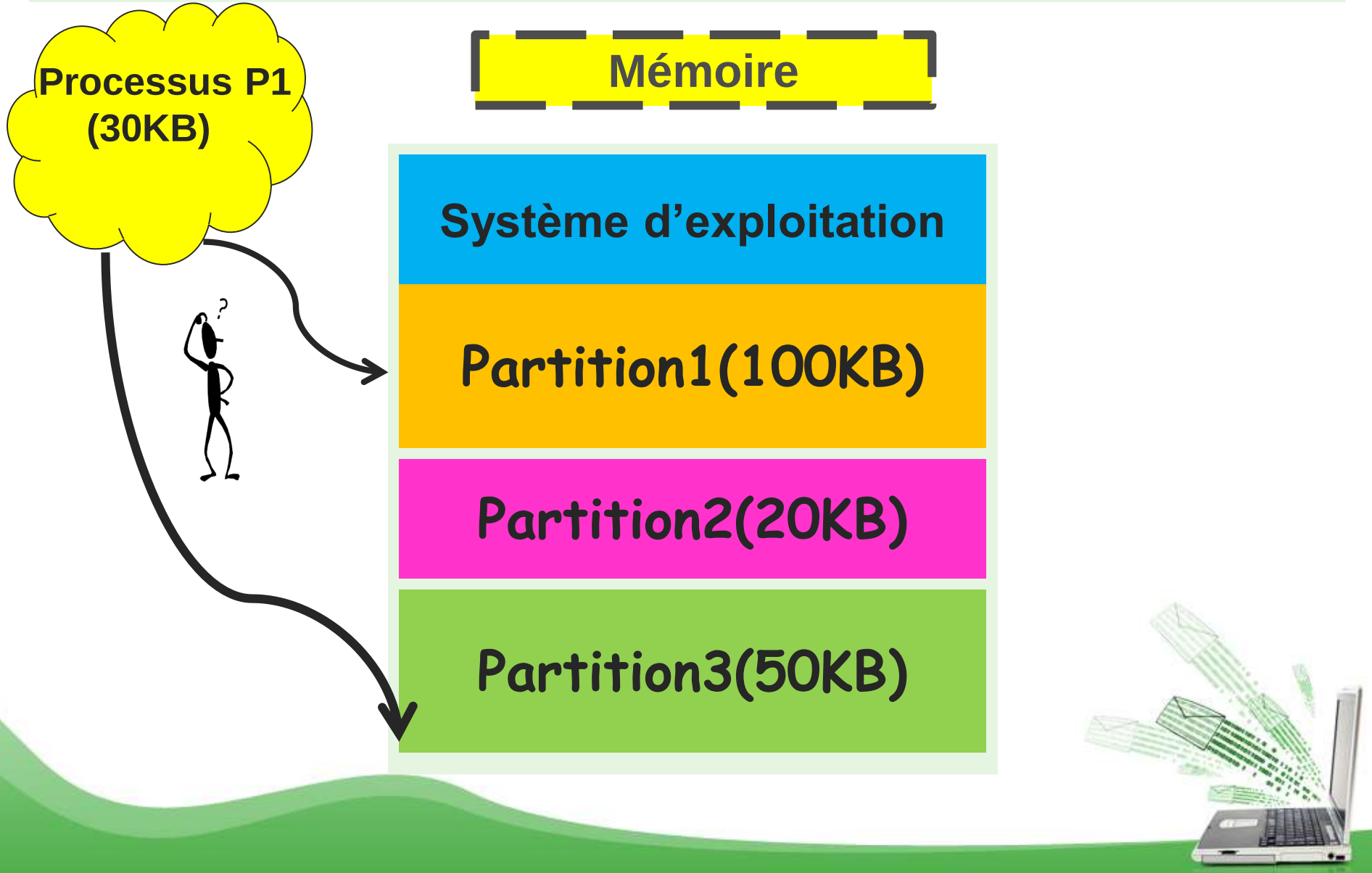
Partitions multiples

Cette stratégie constitue une technique simple pour la mise en œuvre de la multiprogrammation. La mémoire principale est divisée en régions séparées ou partitions mémoires ; chaque partition dispose de son espace d'adressage. Le partitionnement de la mémoire peut être **statique (fixe)** ou **dynamique(variable).** **Chaque processus est chargé entièrement en mémoire.**

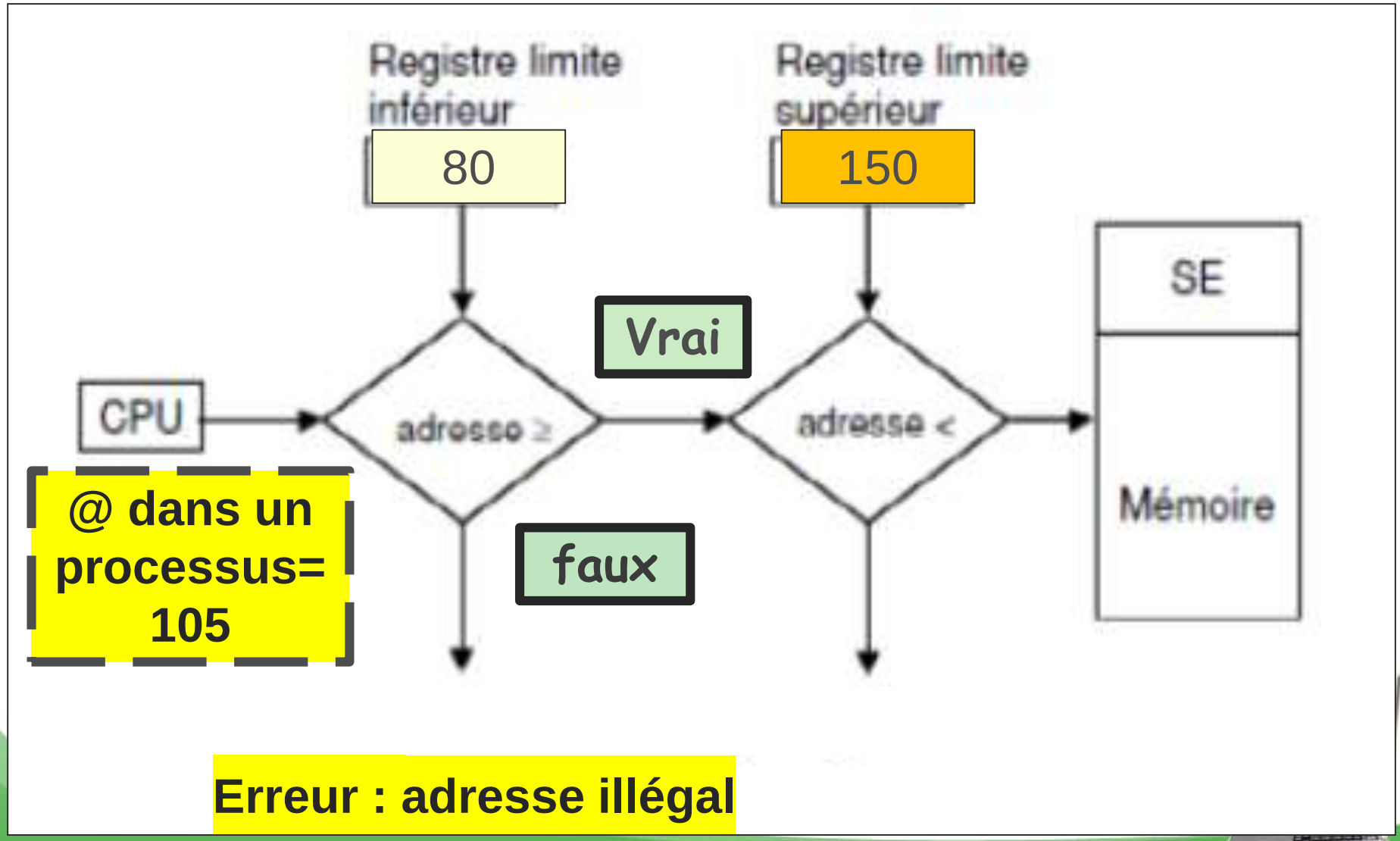
Partitions multiples: Partitions fixes

Dans ce schéma, plusieurs programmes peuvent partager la mémoire. Pour chaque utilisateur il existe une partition. La méthode la plus simple consiste à diviser la mémoire en partitions qui peuvent être de tailles égales ou inégales. On appelle ceci Mutiprogramming with a Fixed Number of Tasks **(MFT).Avec MFT, le nombre et la taille des partitions sont fixés à l'avance.**

Partitions multiples: Partitions fixes



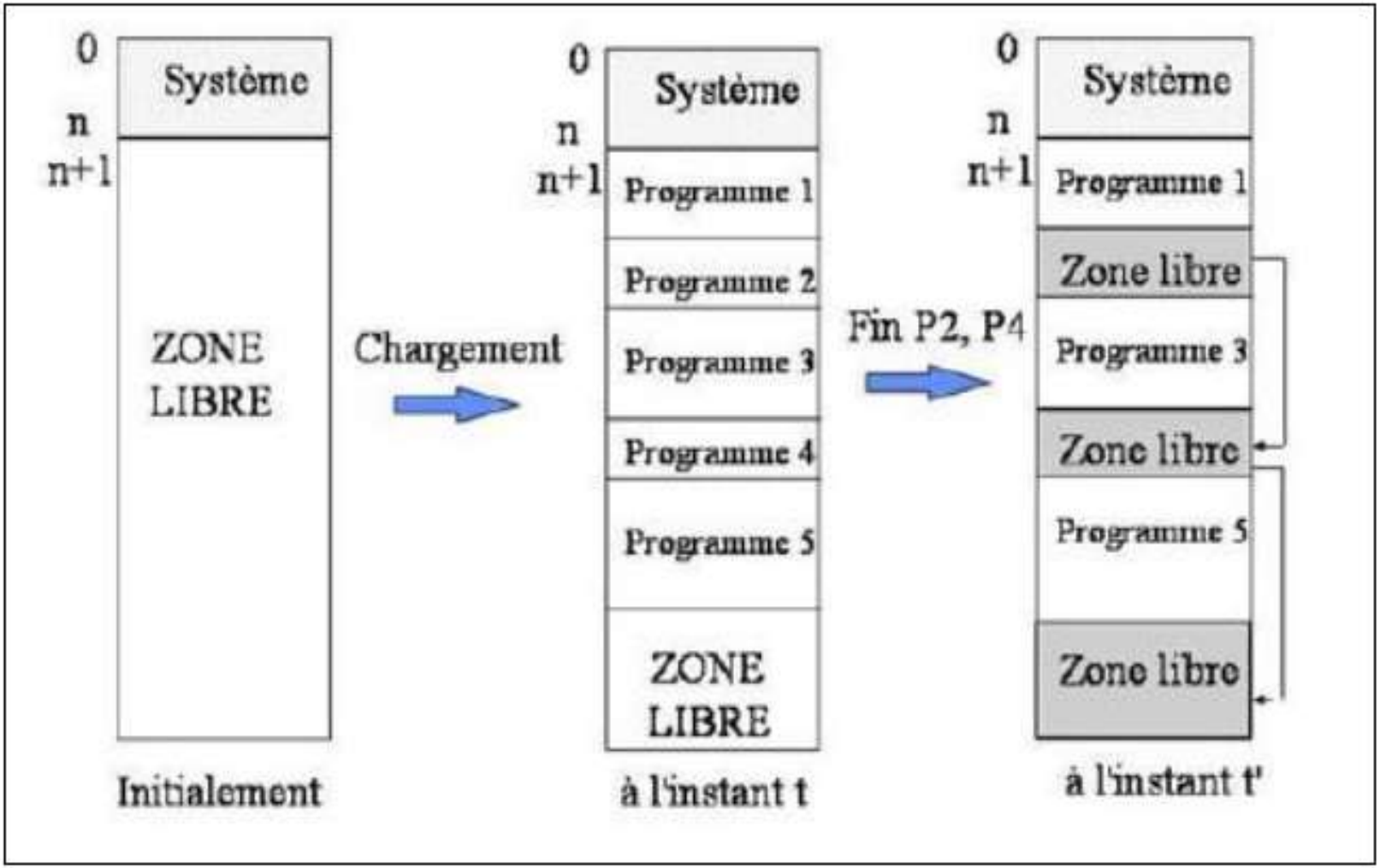
Partitions multiples: Partitions fixes



Partitions multiples: Partitions variables

Dans ce cas, la mémoire est découpée dynamiquement, suivant la demande.
Ainsi, à chaque programme est allouée une partition exactement égale à sa taille.
Quand un programme termine son exécution, sa partition est récupérée par le système pour être allouée à un autre programme complètement ou partiellement selon la demande.

Partitions multiples: Partitions variables



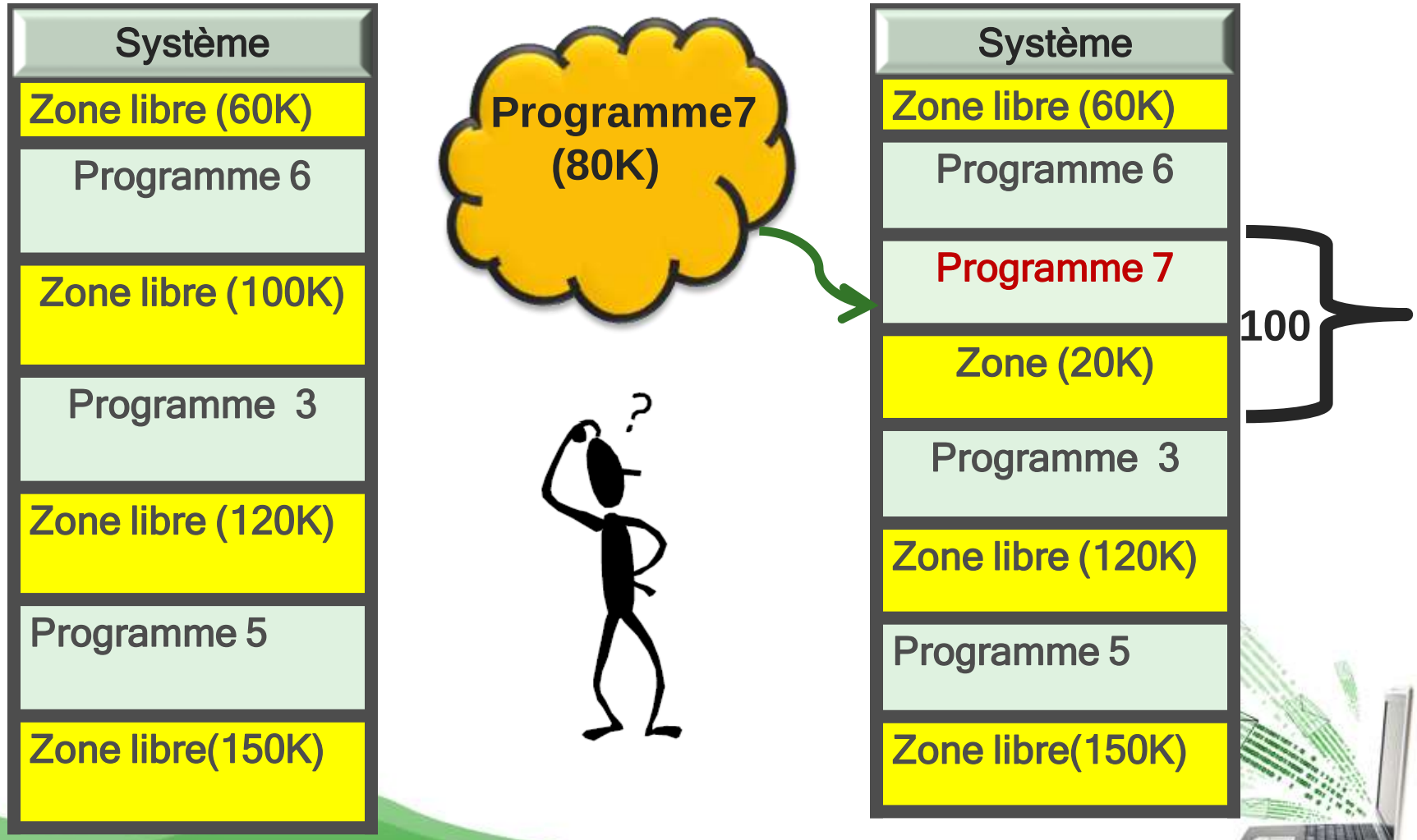
Partitions multiples: Partitions variables

Algorithme de placement : Il existe plusieurs algorithmes afin de déterminer l'emplacement d'un programme en mémoire. Le but de tous ces algorithmes est de maximiser l'espace mémoire occupée, autrement dit, diminuer la probabilité de situations où un processus ne peut pas être servi, même s'il y a assez de mémoire.

Partitions multiples: Partitions variables

- First-fit (le premier trouvé)
- Next-fit (le prochain trouvé)
- Best-fit (le meilleur choix)
- Worst-fit (le plus mauvais choix)

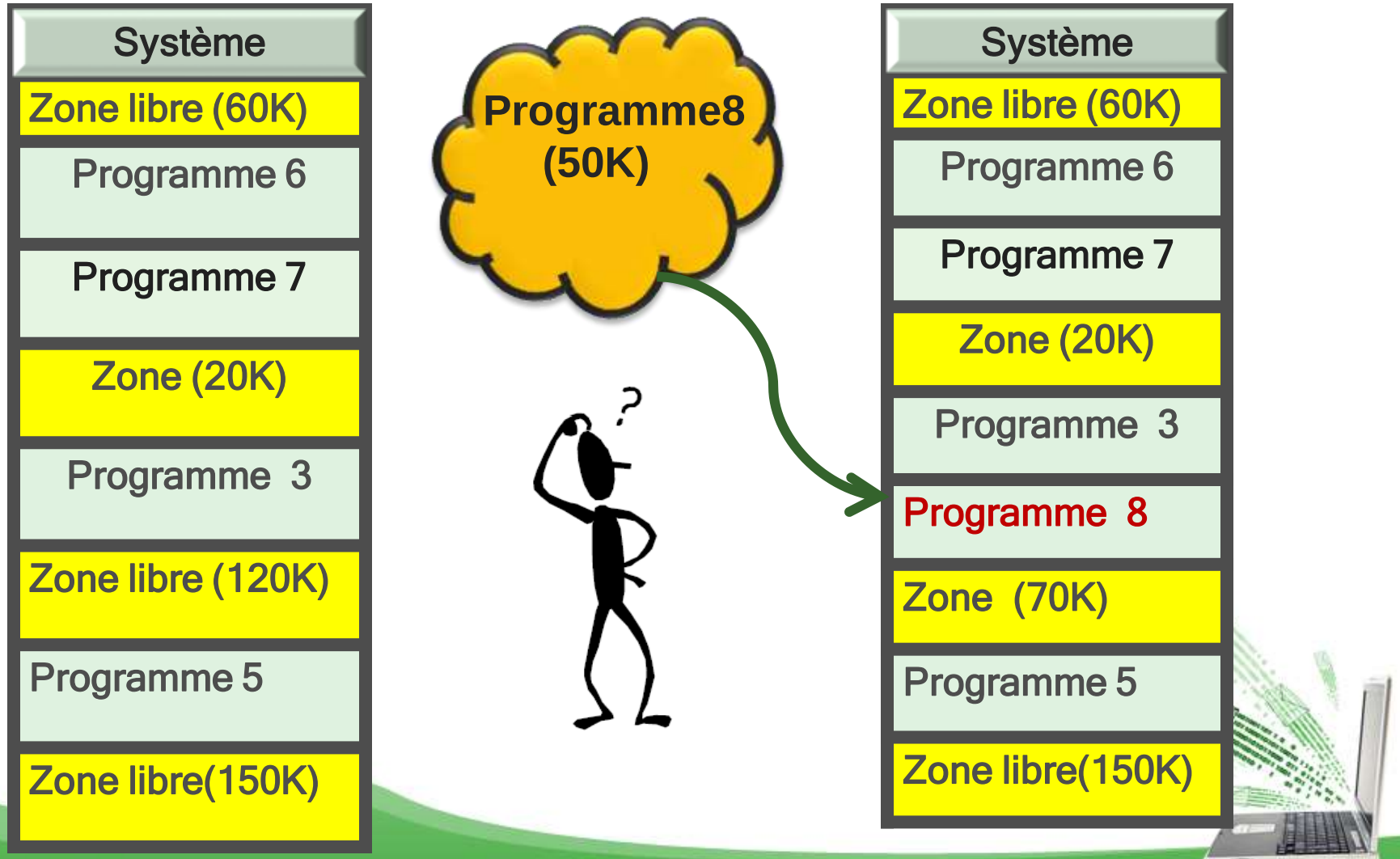
First-fit (le premier trouvé)



Next-fit (le prochain trouvé)

Cet algorithme est une variante du précédent où le programme est mis dans le premier bloc de mémoire suffisamment grand à partir du dernier bloc alloué.

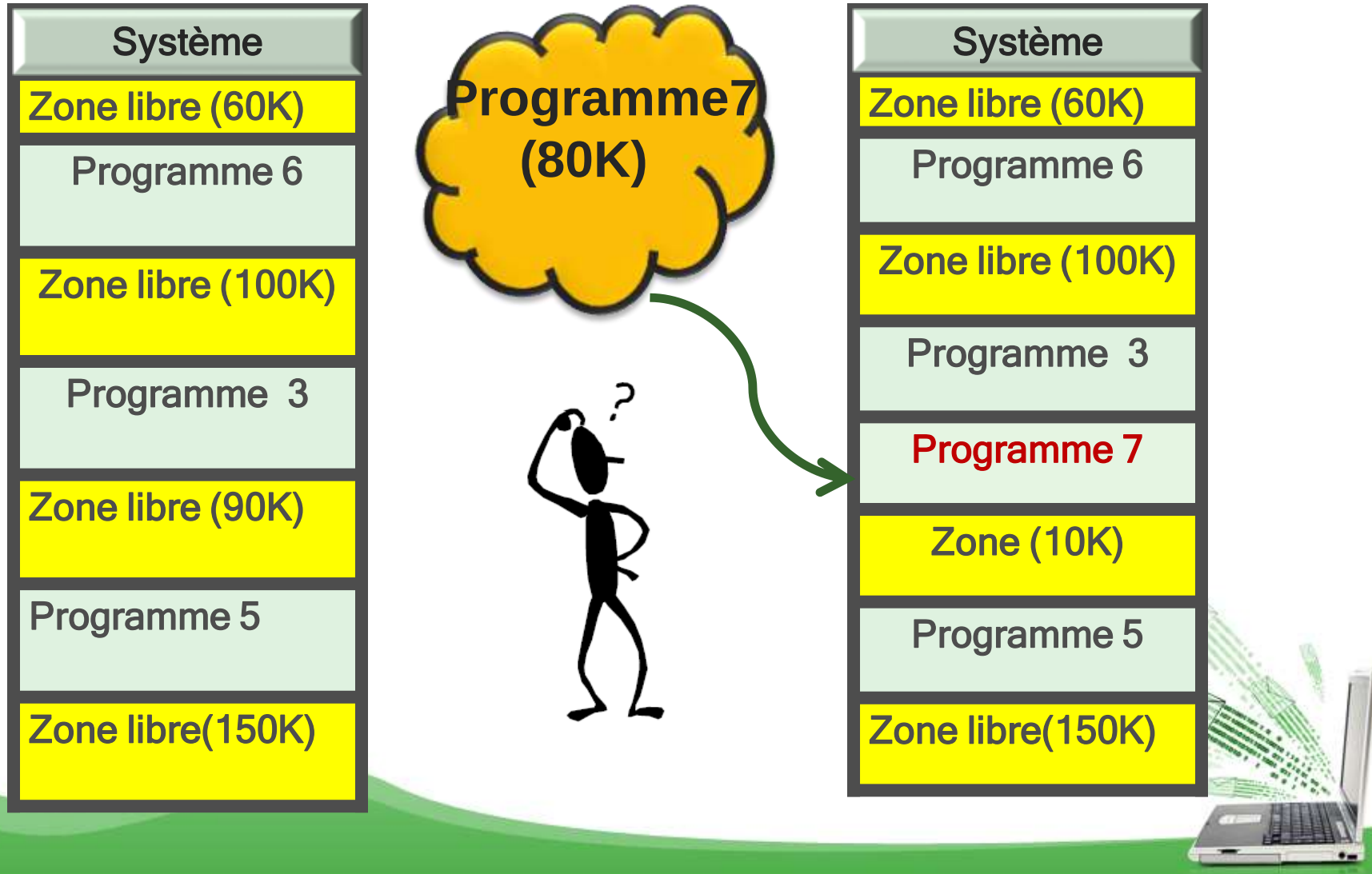
Next-fit (le prochain trouvé)



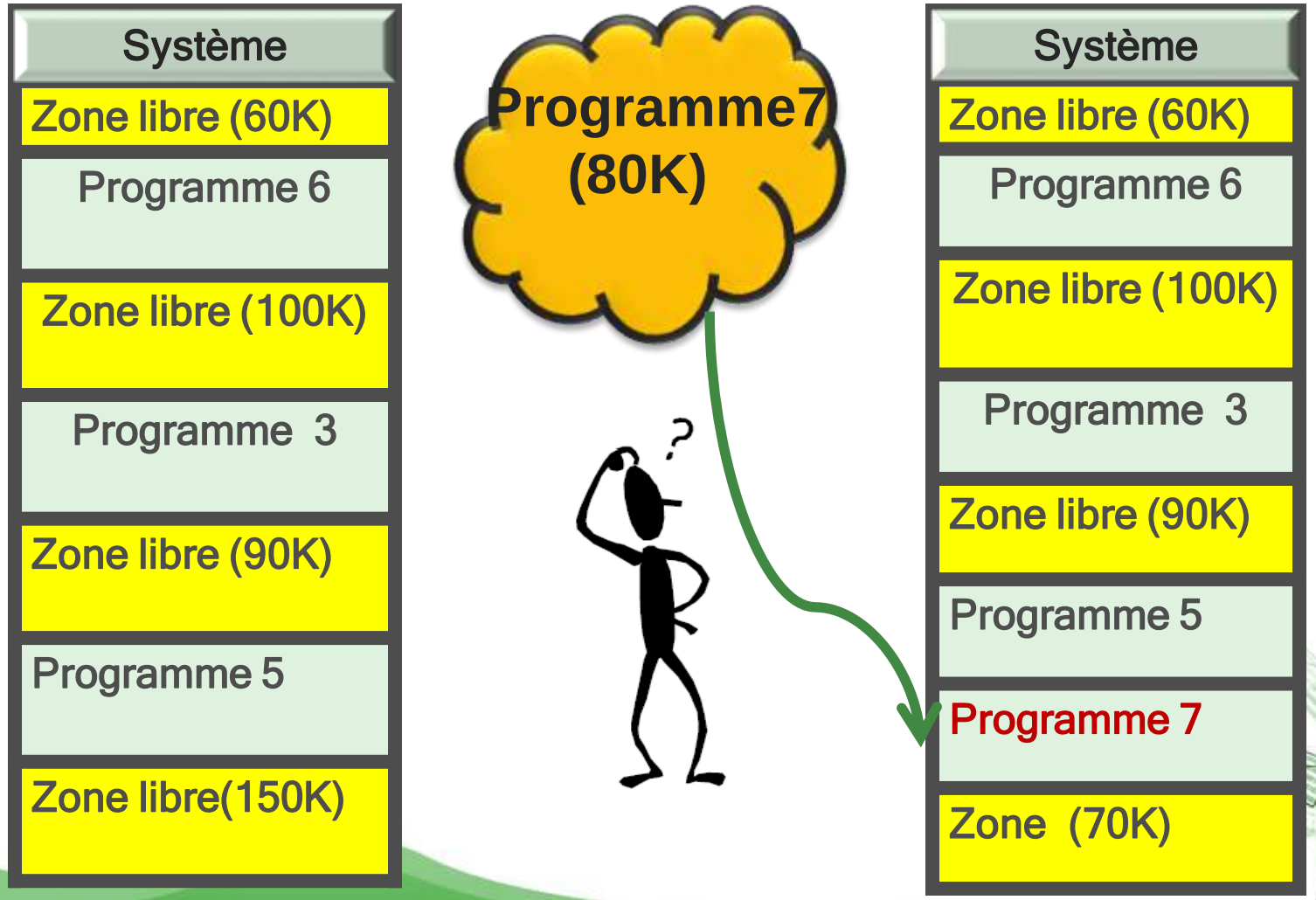
Best-fit (le meilleur choix)

Le programme est mis dans le bloc de mémoire le plus petit dont la taille est suffisamment grande pour l'espace requis.

Best-fit (le meilleur choix)



Worst-fit (le plus mauvais choix)

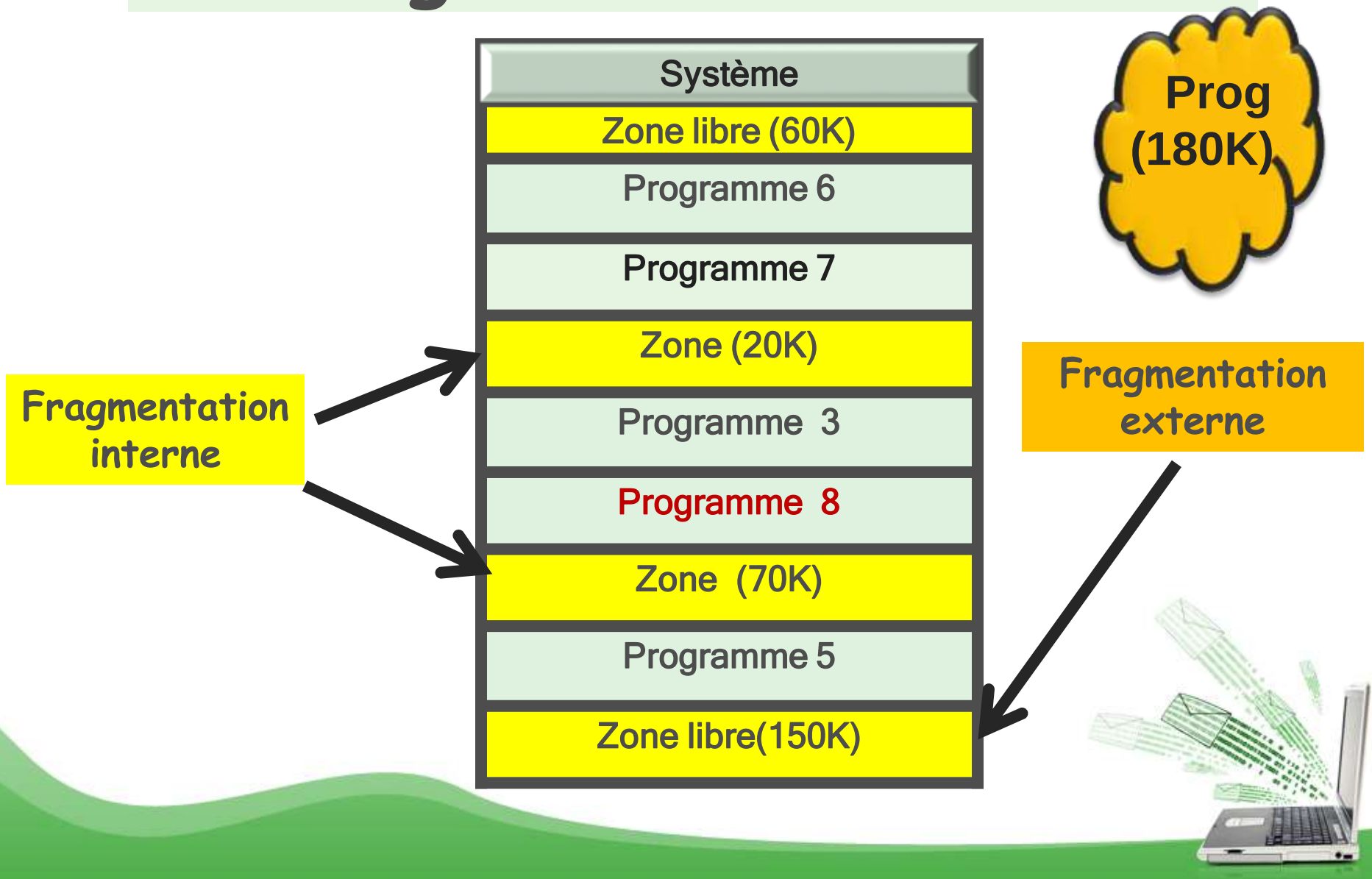


La fragmentation mémoire

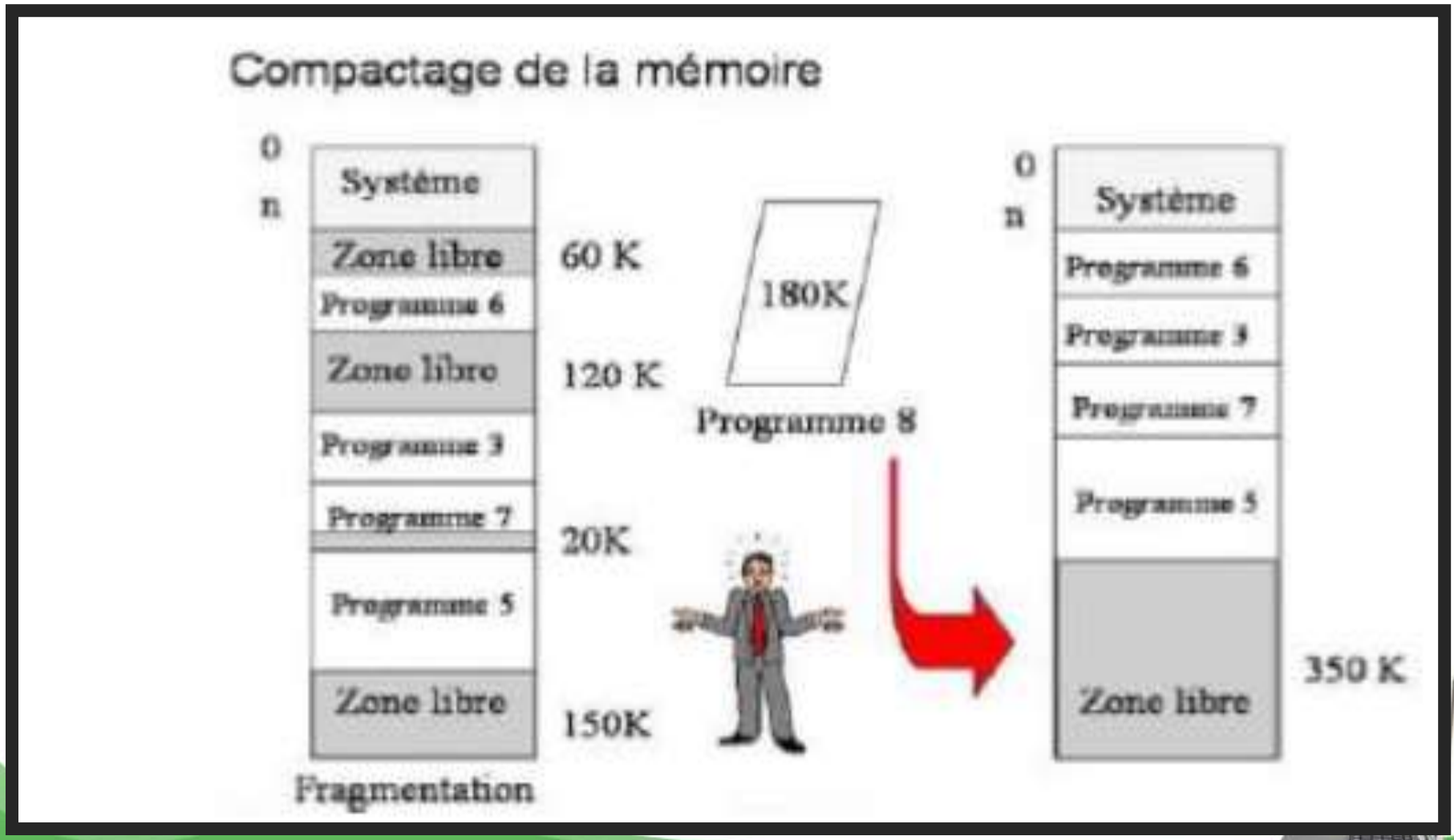
deux types d'espaces de mémoire non utilisés:

- **La fragmentation interne:** la partie d'une partition non utilisée par un processus est nommée fragmentation interne.
- **La fragmentation externe :** Les partitions qui ne sont pas utilisées, engendrent la fragmentation externe.

La fragmentation mémoire



Le compactage : une solution au problème de la fragmentation externe



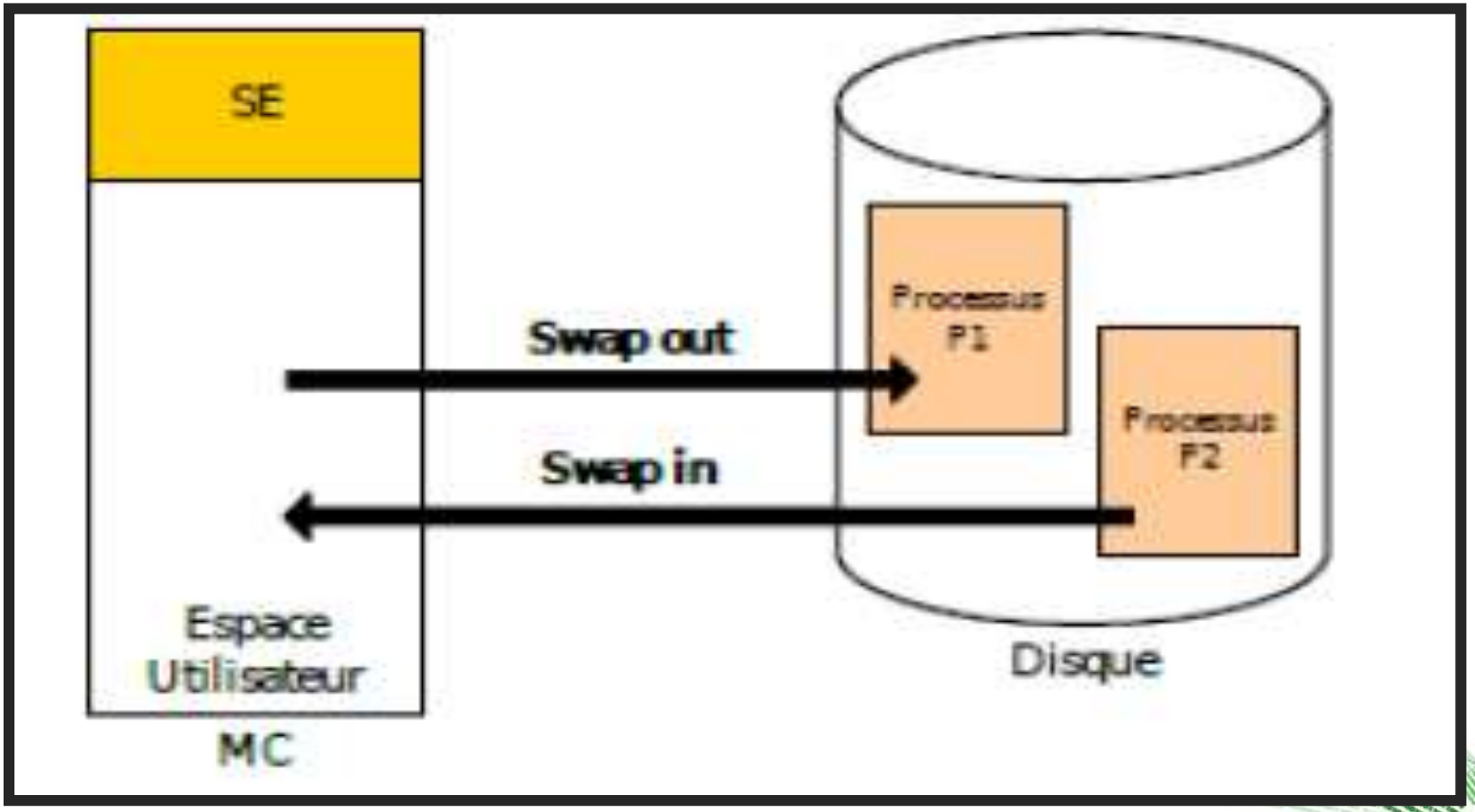
Va-et-vient (Swapping)

Puisque la mémoire ne peut pas contenir les processus de tous les utilisateurs, il faut placer quelques uns sur le disque, en utilisant le **système de va-et-vient**.

Va-et-vient (Swapping)

- **Swap-Out**: Un processus qui est inactif (soit bloqué, soit préempté) peut donc être déplacé temporairement sur une partie réservée du disque, cette **opération est connue sous le nom de Swap-Out**.
- **Swap-In**: Le processus déplacé sur le disque sera ultérieurement rechargé en mémoire pour lui permettre de poursuivre son exécution ; on parle dans ce cas d'une opération **swap-In**

Va-et-vient (Swapping)



Allocation non contiguë

L'objectif principal de l'allocation non contiguë est de pouvoir charger un processus en exploitant au mieux l'ensemble des trous mémoire. Un programme est divisé en morceaux dans le but de permettre l`allocation séparée de chaque morceau. Les morceaux sont beaucoup plus petits que le programme entier et donc permettent une utilisation plus efficace de la mémoire.

Allocation non contiguë

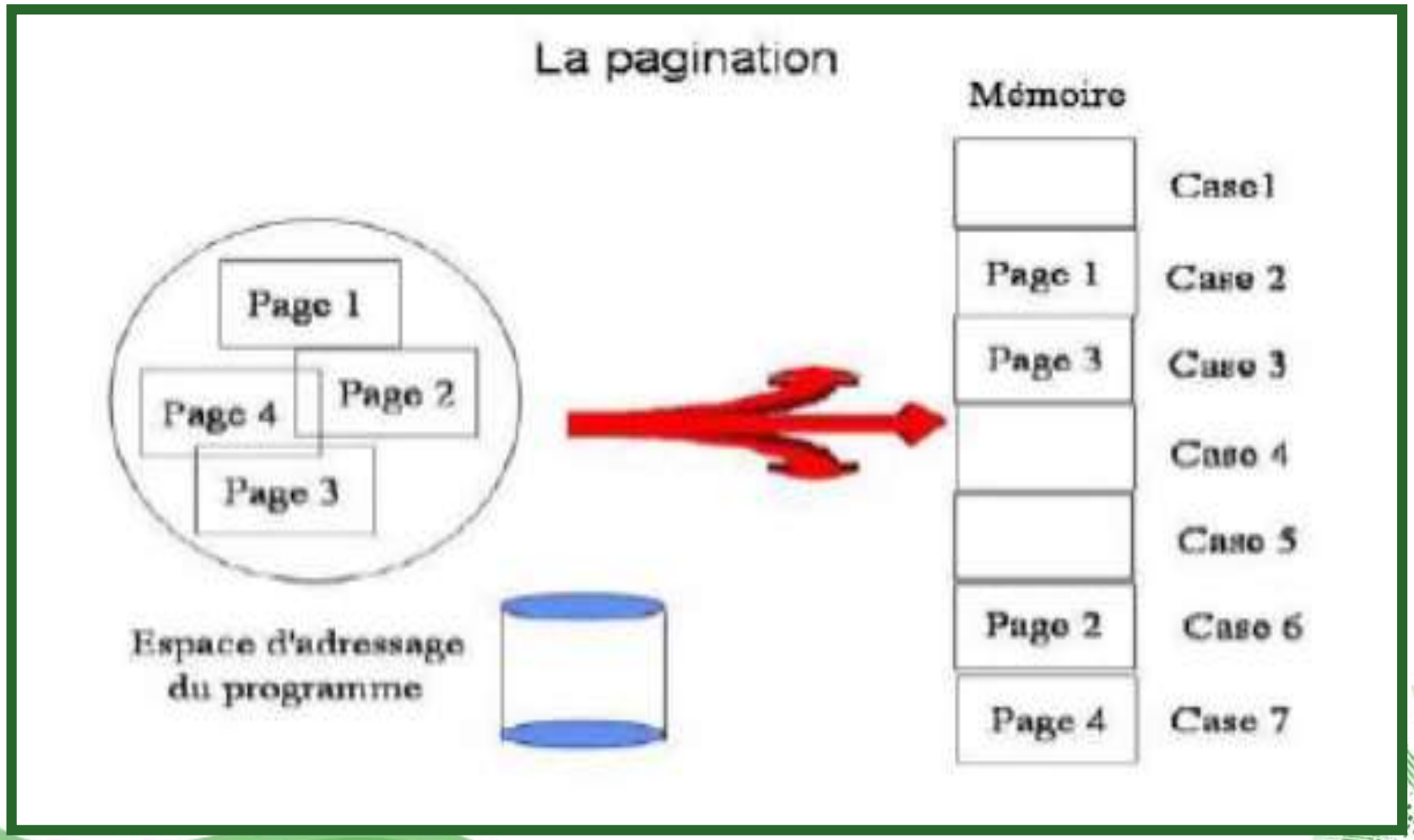
Les gestionnaires mémoire modernes utilisent deux mécanismes fondamentaux basés sur l'allocation non contiguë:

- **La pagination**
- **La segmentation**
- Ces deux techniques peuvent être combinées, on parle dans ce cas de la technique de **segmentation paginée.**

La pagination

Les pgms sont découpés en parties que l'on nomme pages. La mémoire physique est, elle aussi, Découpée en parties appelés cadres ou cases, de même taille. Pour exécuter un pgm , le SE charge en mémoire centrale, uniquement, une ou plusieurs pages. Cette partie est dite résidente. Les pages sont chargées en MC à la demande, donc elles ne sont pas Forcément contiguës.

La pagination



La pagination

- Traduction des adresses virtuelles en adresses réelles
- La pagination à la demande
- Algorithmes de remplacement

Traduction des adresses virtuelles en adresses réelles

La conversion de l'adresse virtuelle en adresse réelle est effectuée, généralement par le circuit appelé MMU (circuit matériel de gestion mémoire).L'adresse virtuelle est 1 couple composé de n° de la page et d'un déplacement relatif au début de la page. L'adresse réelle est 1 couple composé d'un n° de la case et d'un déplacement au sein de la case. La correspondance entre les pages et les cases est mémorisée dans une table appelée Table des pages

Traduction des adresses virtuelles en adresses réelles

L'adresse virtuelle

N °PAGE	DEPLACEMENT

Table des pages de P1

N page	N case		
1			
2			
3			
4			

L'adresse réelle

N °case	DEPLACEMENT

Traduction des adresses virtuelles en adresses réelles

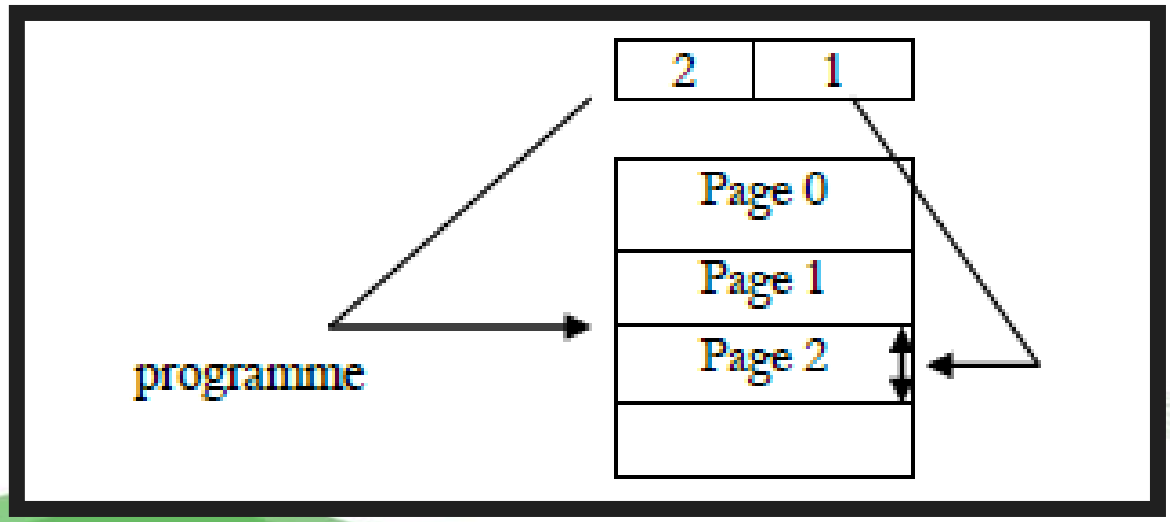
- Les deux informations, numéro de page et déplacement dans le page, peuvent être calculées à partir de l'adresse logique A du mot, en se basant sur la taille T de la page : L'adresse A est égale à (p, d) telle que

P= A/T où / est la division entière.

D= A modulo T.

Traduction des adresses virtuelles en adresses réelles

Exemple :
Soit un mot d'adresse virtuelle 2001
dans un système paginé de taille 1000 octets
l'adresse virtuelle paginée =(N°p=2,D=1)



Structure de la table des pages

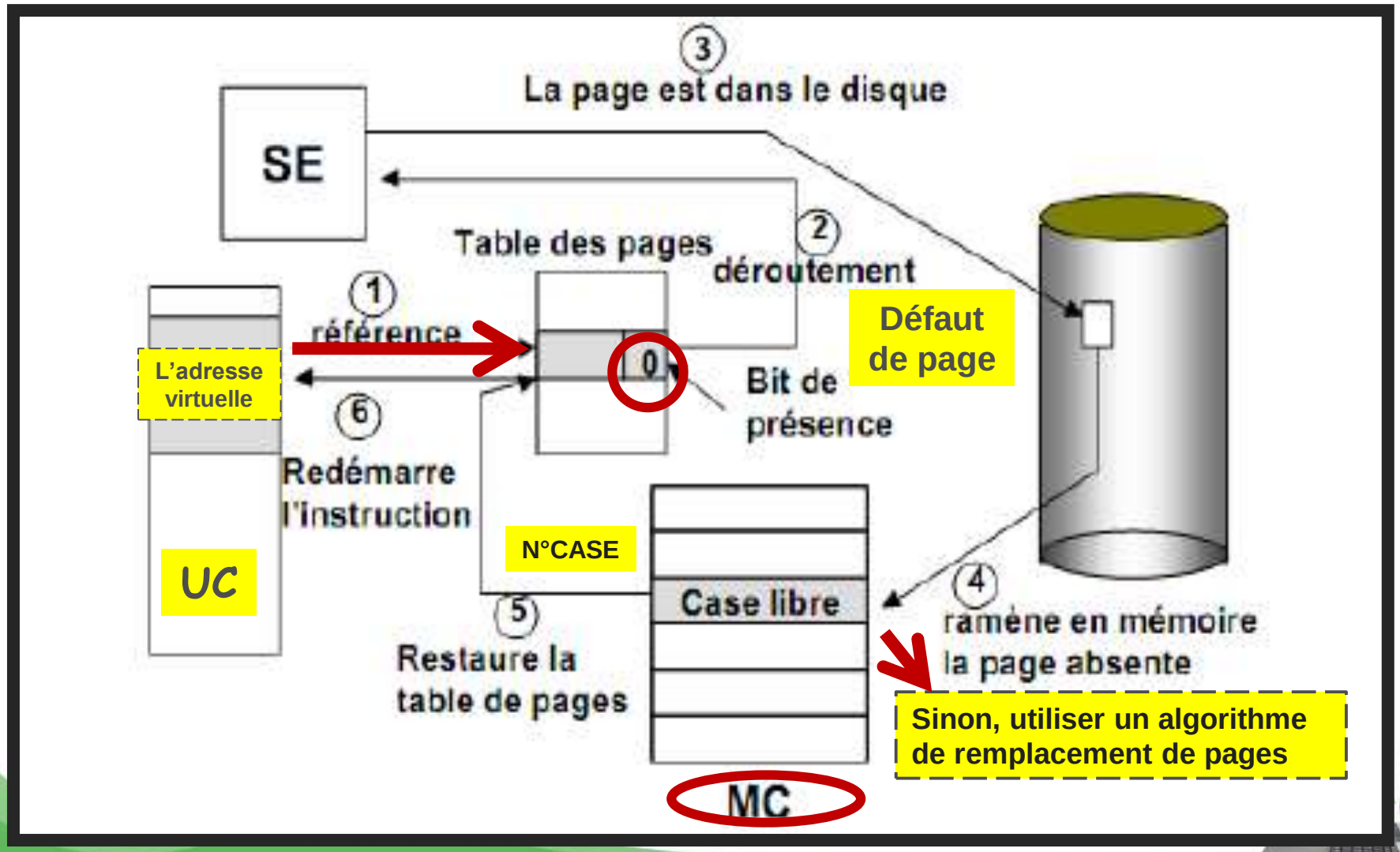
Chaque entrée de la table des pages est composée de plusieurs champs

- P : bit de présence de la page en MC ;
- R : bit de référence de la page ;
- Pr : bits de protection (L/E, exécution, droits d'accès) ;
- M : bit de modification de la page (nécessité de la réécrire sur le disque) ;
- N° de la case correspondant à la page.

La pagination à la demande

Lorsqu'une adresse référencée appartient à une page n'ayant pas de copie en mémoire , on dit qu'on à un défaut de page .le système cherche alors sur le disque une copie de la page demandée, la charge en mémoire en déplaçant une autre page et met à jour la table des pages. cette méthode de gestion de la mémoire, qui consiste à charger une page seulement si elle est référencée, est appelée **pagination à la demande**

La pagination à la demande



Algorithmes de remplacement

A la suite d'un **défaut de page**, le SE doit ramener en MC la page manquante à partir du disque. S'il n'y a pas de cases libres, il doit retirer une page de la MC pour la remplacer par celle demandée. D'où la question : Quelle est la page à retirer de manière à minimiser le nombre de défauts de page ?

Algorithmes de remplacement

Quand une page n'est pas en mémoire...

A. Trouver l'**emplacement de la page désirée sur disque.**

B. Trouver un **cadre de page (case) libre.**

(a) S'il existe un cadre libre, l'utiliser.

(b) Sinon, utiliser un **algorithme de remplacement de pages pour** sélectionner un cadre (victime).

(c) Enregistrer la page victime dans le disque, modifier les de pages.

C. **Lire la page désirée dans le cadre libéré, modifier les tables de pages**

D. Redémarrer le processus utilisateur.

Algorithme de remplacement de page FIFO

une MC à 3 cases et la suite de références suivantes :
W= [7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1]

Case\Page	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
0	**7**	7	7	**2**	2	2	2	**4**	4	4	**0**	0	0	0	0	0	0	**7**	7	7
1		**0**	0	0	0	**3**	3	3	**2**	2	2	2	2	**1**	1	1	1	1	**0**	0
2			**1**	1	1	1	**0**	0	0	**3**	3	3	3	3	**2**	2	2	2	2	**1**

On constate 15 défauts de page.
L'algorithme est rarement utilisé car il génère beaucoup de défauts de page.

Algorithme de remplacement de page la moins récemment utilisée (LRU)

Case\Page	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
0	**7**	7	7	**2**	2	2	2	**4**	4	4	**0**	0	0	**1**	1	1	1	1	1	1
1		**0**	0	0	0	0	0	0	**3**	3	3	3	3	**0**	0	0	0	0	0	0
2			**1**	1	1	**3**	3	3	**2**	2	2	2	2	2	2	2	**7**	7	7	

Stratégie de remplacement optimal

la page victime sera la page qui ne sera pas référencée dans le futur immédiat

Demandes	1	2	3	1	4	1	2	5	1	2	3	4	5
Case 1	1	1	1	1	1	1	1	1	1	1	1	4	4
Case 2		2	2	2	2	2	2	2	2	2	2	2	2
Case 3			3	3	3	3	3	3	3	3	3	3	3
Case 4					4	4	4	5	5	5	5	5	5
Défaut de page	D	D	D		D			D				D	

Stratégie de NFU : not frequently used (Least Frequently Used)

la page victime est la page qui n'est pas fréquemment utilisée

Demandes	1	2	3	1	4	1	2	5	1	2	3	4	5
Case 1	1	1	1	1	1	1	1	1	1	1	1	1	1
Case 2		2	2	2	2	2	2	2	2	2	2	2	2
Case 3			3	3	3	3	3	5	5	5	5	4	4
Case 4					4	4	4	4	4	4	3	3	5
Défaut de page	D	D	D		D			D			D	D	D

Stratégie de MFU : Most frequently used

la page victime est la page qui est fréquemment utilisée

Demandes	1	2	3	1	4	1	2	5	1	2	3	4	5
Case 1	1	1	1	1	1	1	1	5	5	5	5	4	4
Case 2		2	2	2	2	2	2	2	1	1	1	1	5
Case 3			3	3	3	3	3	3	3	2	2	2	2
Case 4					4	4	4	4	4	4	3	3	3
Défaut de page	D	D	D		D			D	D	D	D	D	D

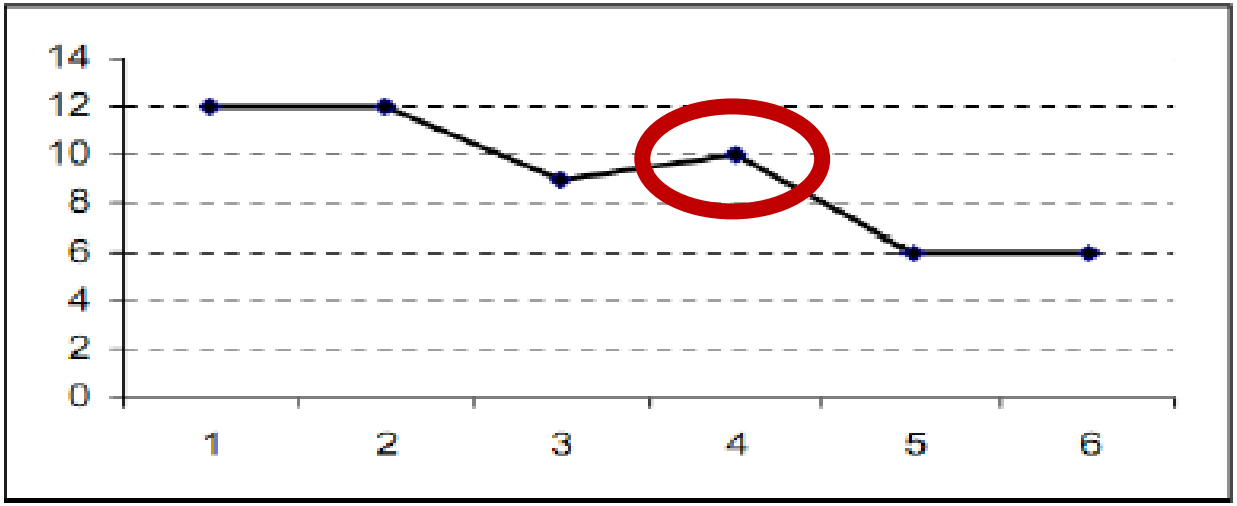
Algorithmes de remplacement

plusieurs autres algorithmes exemple

- RANDOM
- Seconde chance

L'anomalie de Belady

Afin d'illustrer les problèmes rencontrés avec un algorithme de remplacement FIFO, on peut envisager la chaîne de références suivantes : 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5. La figure suivante montre le nombre de défauts de pages en fonction des cadres de pages disponibles.



Anomalie de Belady

Cadres de pages	Défauts de pages
1	12
2	12
3	9
4	10
5	6
6	6

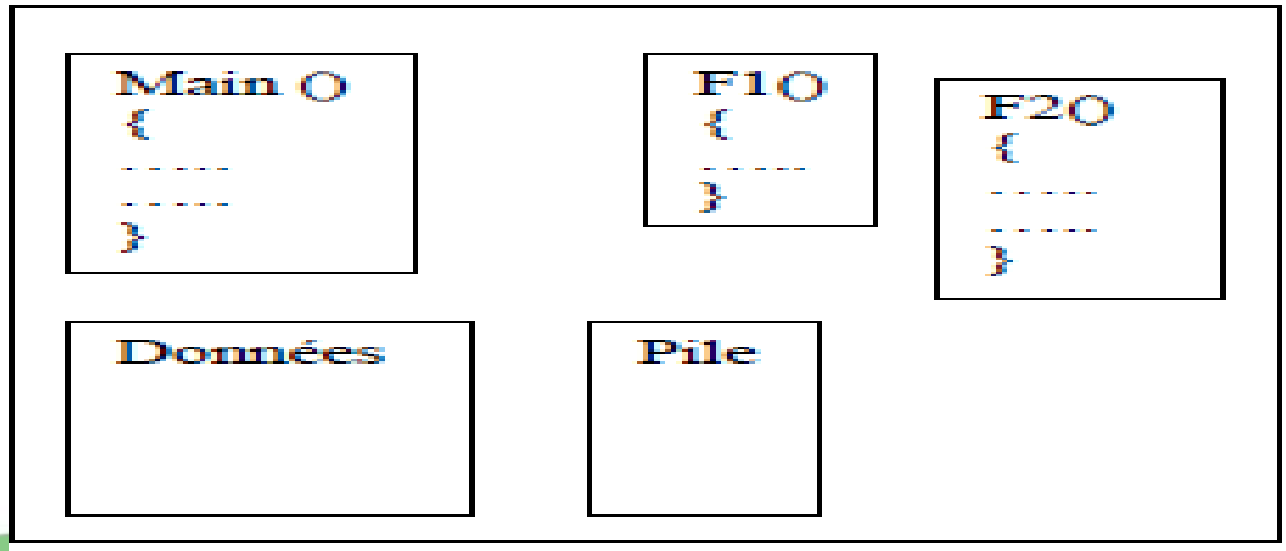
La segmentation

La pagination constitue un découpage de l'espace d'adressage du processus qui ne correspond pas à l'image que le programmeur a de son programme. Pour le programmeur, un programme est généralement constitué des données manipulées par ce programme, d'un programme principal, de procédures séparées et d'une pile d'exécution. La segmentation est un découpage de l'espace d'adressage qui cherche à conserver cette vue du programmeur.

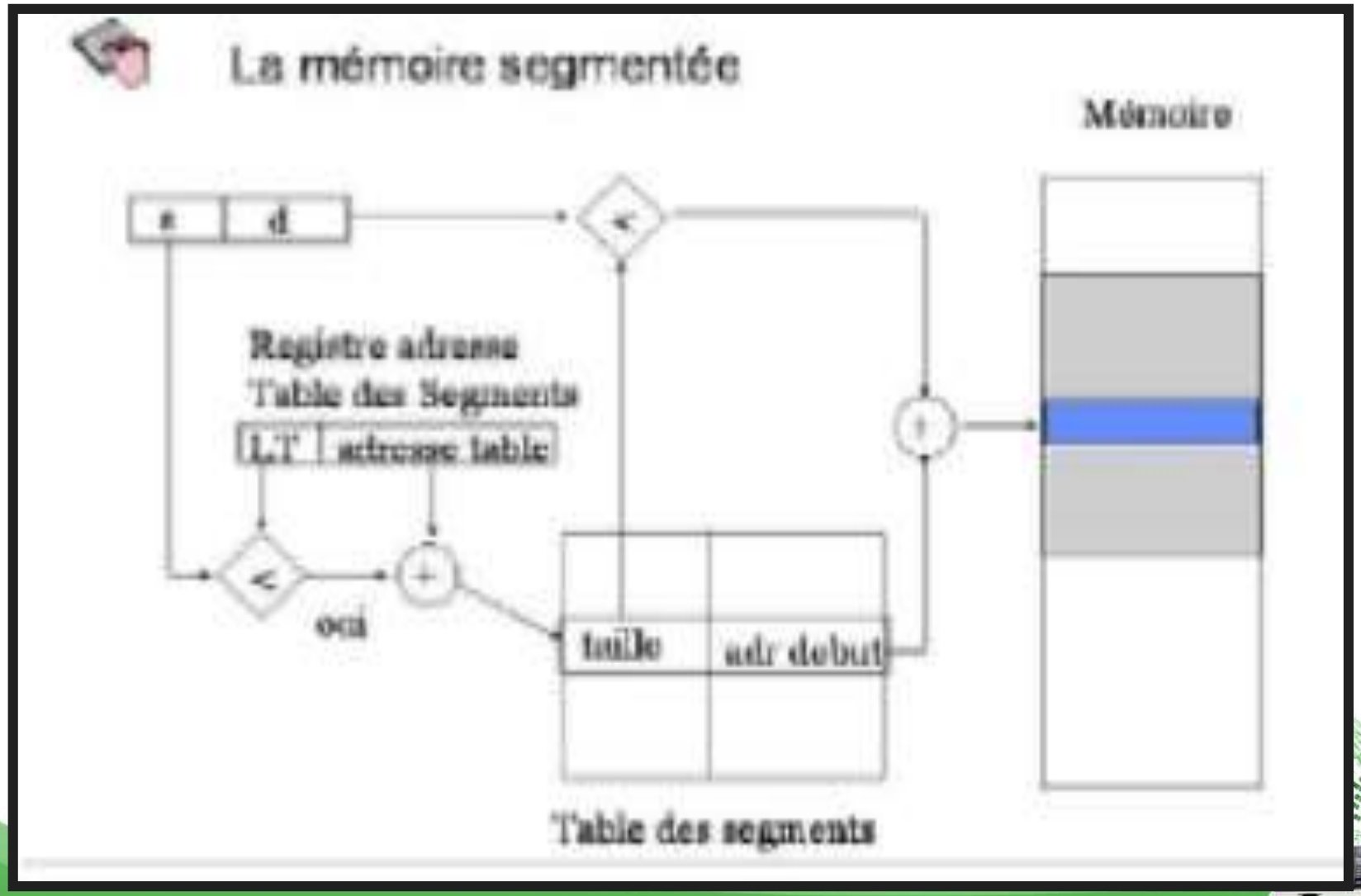
Traduction de l'adresse segmentée vers l'adresse physique

L'adresse d'un octet est donc formée par le couple :

- n°de segment pour le mot
- D=déplacement relativement au début du segment



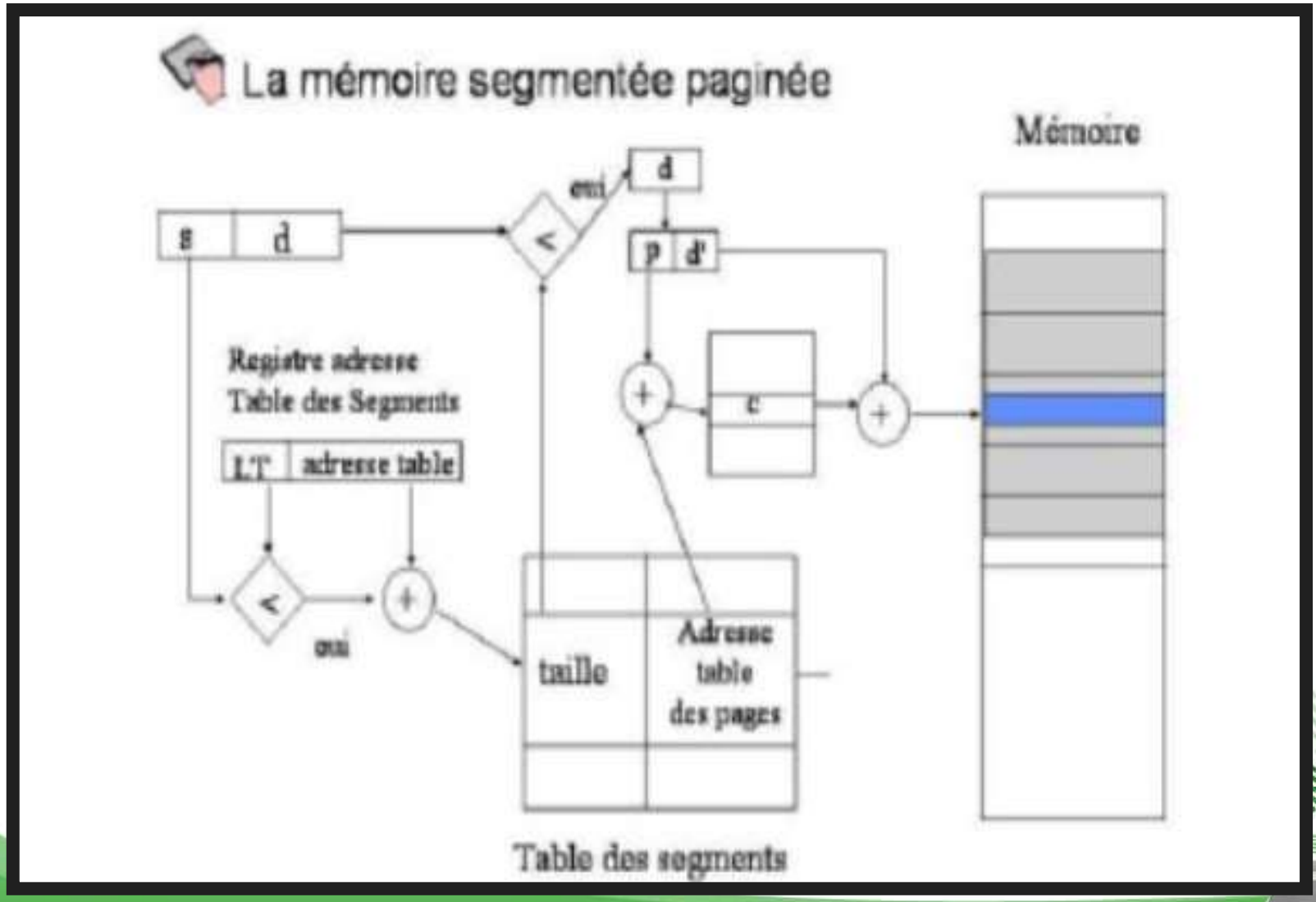
Traduction de l'adresse segmentée vers l'adresse physique



Segmentation et Pagination

Dans la cas où pagination et segmentation sont simultanément employées, une table des segments est définie pour chaque segment de l'espace d'adressage du processus. Chaque segment est à son tour paginé, il existe donc une table des pages pour chaque segment. Ainsi une entrée de la table des segments ne contient plus l'adresse du segment correspondant en mémoire physique mais contient l'adresse de la table des pages en mémoire physique pour ce segment.

Segmentation et Pagination



Exercice 2(TD6)

On considère la table des segments suivante pour un processus P1 :

segment	base	longueur
0	540	234
1	1254	128
2	54	328
3	2048	1024
4	976	200

Exercice 2(TD6)

- Calculer les adresses réelles correspondant aux adresses virtuelles suivantes, ou signaler les éventuelles erreurs d'adressage qui engendreront un déroutement puis le postage à P1 d'un signal du type *segmentation fault:*

• (0, 128), (1, 100), (2, 465), (3, 888), (4, 100),
(4, 344)

- L'adresse virtuelle (4,200) est-elle valide ?

Exercice 2(TD6)

- (0:128) : déplacement valide (128<234).
  Adr_physique = base + longueur = 540 + 128 = 668.
- (1:100) : déplacement valide (100<128).
  Adr_physique = base + longueur = 1254+ 100 = 1354.
- (2:465) : déplacement invalide (465>328).
- (3:888) : déplacement valide (888<1024).
  Adr_physique = base + limite = 2048 + 888 = 2936.
- (4:100) : déplacement valide (100<200).
  Adr_physique = base + limite = 976+ 100 = 1076.
- (4:344) : déplacement invalide car (344>200)

2) **Non. Dans un segment de longueur 200, les déplacements valides sont dans l'intervalle [0-199].**

Exercice : allocation non contiguë

Soit un système à partitions **fixes** de mémoire avec allocation contigüe.avec une mémoire constituée de zones vides dans l'ordre suivant :

Z1= 11Ko, **Z2**= 4 Ko, **Z3**= 24 Ko,
Z4= 18Ko, **Z5**=7Ko, **Z6**= 9 Ko,
Z7= 14 Ko et **Z8**= 15 Ko.

Indiquer quelle zone sera prise lors des requêtes dans l'ordre ci-dessous suivant les algorithmes précisés : **P1(12KO),P2(10KO),P3(9KO)**

Exercice : allocation non contiguë

Algorithme / Requête ↓	First-Fit	Best-Fit
12 Ko	Z3	Z7
10 Ko	Z1	Z1
9 Ko	Z4	Z6

Exercice : allocation contiguë

- Soit un système à partitions **variables** de mémoire avec allocation contigüe. A un instant donné les **partitions libres** sont 100K, 500k, 200K, 300k et 600K (par ordre croissant des adresses).

On considère une liste d'arrivée des processus qui demandent A(**212K**), **B(417K)**, **C(112K) et D(426K)**.

- Si aucun espace mémoire n'est suffisant pour contenir le bloc à allouer, la mémoire est compactée. Si, après compactage, l'allocation n'est toujours pas possible, alors l'allocation est refusée.
- Donner le comportement des algorithmes selon les stratégies First Fit, Best Fit et Worst Fit.

Exercice : allocation non contiguë

Etape initial (la mémoire vide) :

100K, 500k, 200K, 300k et 600K

100	500	200	300	600

Avec First-Fit

A→

100	A	288	200	300	600

B→

100	A	288	200	300	B	183

C→

100	A	C	176	200	300	B	183

D→ Pas de place convenable.

On a recours au compactage :

Donc, après compactage on a :

A	C	B	959

D→

A	C	B	D	533

Avec Best-Fit

A→	100	500		200		A	88	600
B→	100	B	83	200		A	88	600
C→	100	B	83	C	88	A	88	600
D→	100	B	83	C	88	A	88	D	174

Avec Worst-Fit

A→

100	500	200	300	A	388

B→

100	B	83	200	300	A	388

C→

100	B	83	200	300	A	C	276

D→ Pas de place disponible.

On a recours au compactage :

Donc, après compactage on a :

B	A	C	959

D→

B	A	C	D	533

Exercice 3 (TD6)

Solution :

Instant T	Processus	Type d'évènement	Zone mémoire
T= 0	A(300,55)	Chargement de A	1-300K (occupée)
T=10	B(400,35)	Chargement de B	301-700 K(occupée)
T=30	C(500,35)	Ne peut être chargé, il passe à la file attente	

Exercice 3 (TD6)

T=40	D(300,105)	Chargement de D	701-1000 K(occupée)
T=45	B(400,35)	Terminaison de B	301-700k(libre)
T=50	E(200,35)	Chargement de E	301-500K(occupée)
T=55	A(300,55)	Terminaison de A	1-300K(libre)
T=60	F(100,55)	Chargement de F	1-100k(occupée)
T=70	G(400,35)	Ne peut être chargé , il passe à la file d'attente	
T=85	E(200,35)	Terminaison	301-500k(libre)
T=85	C(500,35)	Chargement	101-600k(occupée)
T=90	H(700,35)	Ne peut être chargé, il passe à la file d'attente	
T=110	I(200,25)	Ne peut être chargé, il passe à la file d'attente	

Exercice 3 (TD6)

T=115	F(100,55)	Terminaison	1-100k(libre)
T=120	C(500,35)	Terminaison	101-600k(libre)
T=120	G(400,35)	Chargement	1-400k(occupée)
T=120	I(200,25)	chargement	401-600(occupée)
T=120	J(400,45)	Ne peut être chargé , il passe à la file d'attente	
T=145	D(300,105)	Terminaison	701-1000k(libre)
T=145	I(200,25)	Terminaison	401-600(libre)
T=145	J(400,45)	Chargement	401-800(occupée)
T=155	G(400,35)	Terminaison	1-400(libre)
T=190	J(400,45)	Terminaison	401-800(libre)
T=190	H(700,35)	Chargement	1-700 (occupée)
T=225	H(700,35)	Terminaison	1-700(libre)

Exercice supplémentaire

On suppose que l'état de la mémoire RAM est décrit par le tableau suivant

Les tailles sont en Ko, et les blocs en gras sont utilisés, alors que les autres sont libres.)
Des requêtes d'allocation de mémoire arrivent dans cet ordre l
: **20 Ko, 10 Ko, 5 Ko et 25 Ko**.

Exercice supplémentaire

Question 1 : A quelles adresses sont alloués les blocs si on utilise la politique : First Fit" , Best-Fit, Worst-Fit

Exercice supplémentaire

B. Allocation contiguë par partitions variables :

- **First-fit :**

Processus	Zone de chargement	Le reste de la zone libre
P1(20k)	Z4(30k)	10K
P2(10k)	Z2(10K)	0
P3(5k)	Le reste de Z4(10k)	5K
P4(25K)	Ne peut être chargé	

- **Best-fit :**

Processus	Zone de chargement	Le reste de la zone libre
P1(20k)	Z8(20K)	0
P2(10k)	Z2(10K)	0
P3(5k)	Z6(5K)	0
P4(25k)	Z4(30k)	5k

- **Worst-fit :**

Processus	Zone de chargement	Le reste de la zone libre
P1(20k)	Z4(30k)	10K
P2(10k)	Z8(20K)	10K
P3(5k)	Z12(20k)	15K
P4(25K)	Ne peut être chargé	

Exercice 3 (TD3)

Simulation d'instructions manquantes:

- On souhaite simuler un jeu d'instructions arithmétiques en virgule flottante sur une machine sans coprocesseur mathématique. Sans ce coprocesseur, l'exécution d'une instruction en virgule flottante (div, mult) par la machine provoque un déroutement pour instruction inexistante et l'arrêt du programme avec un message d'erreur.
- **Question : Ecrire les procédures permettant de mettre en place une simulation logicielle de ces instructions en virgule flottante : lorsqu'un programme exécutera une telle instruction,**
- une procédure simulant son exécution (rendant le même résultat) sera automatiquement appelée.

Solution

Procédure initialisation
Nouveau-deroutement :=<actif, maitre, masqué, **adr**[traiter-déroutement]>
Mep[ADD-FL] :=<actif ,esclave , démasqué, adr[ADD-flott]> ;
……..
……..
Mep[DIV-FL] :=<actif, esclave , démasqué , adr[DIV-flott] ;
FIN

Solution

Procédure traiter-déroutement

Début

Sauver (zone) /*pour sauvegarder le contexte des registres processeur */

Cas cause dans

.....

Instr.non existante: /*Si la cause de déroutement est instruction inexistante*/

Code_op :=MP[(ancien-déroutement.CO)].code_opération

Solution

```
Cas code_op dans
      ……
      ADD_FL : déterminer opérandes
                  Empiler (pile-usager, ancien-déroutement.CO+1) ;
                  Charger-mep(mep[ADD_FL])
      Sinon
    < traitement de l'erreur>
   FINCAS
FIN CAS
     Restaurer(Zone)
     Charger-mep(ancien-déroutement)
Fin traiter-déroutement
```