GESTION DE LA MÉMOIRE PRINCIPALE

SE

Madame Khaoula ElBedoui-Maktouf

2ème année Ingénieur Informatique

Plan

I. **Préambule**

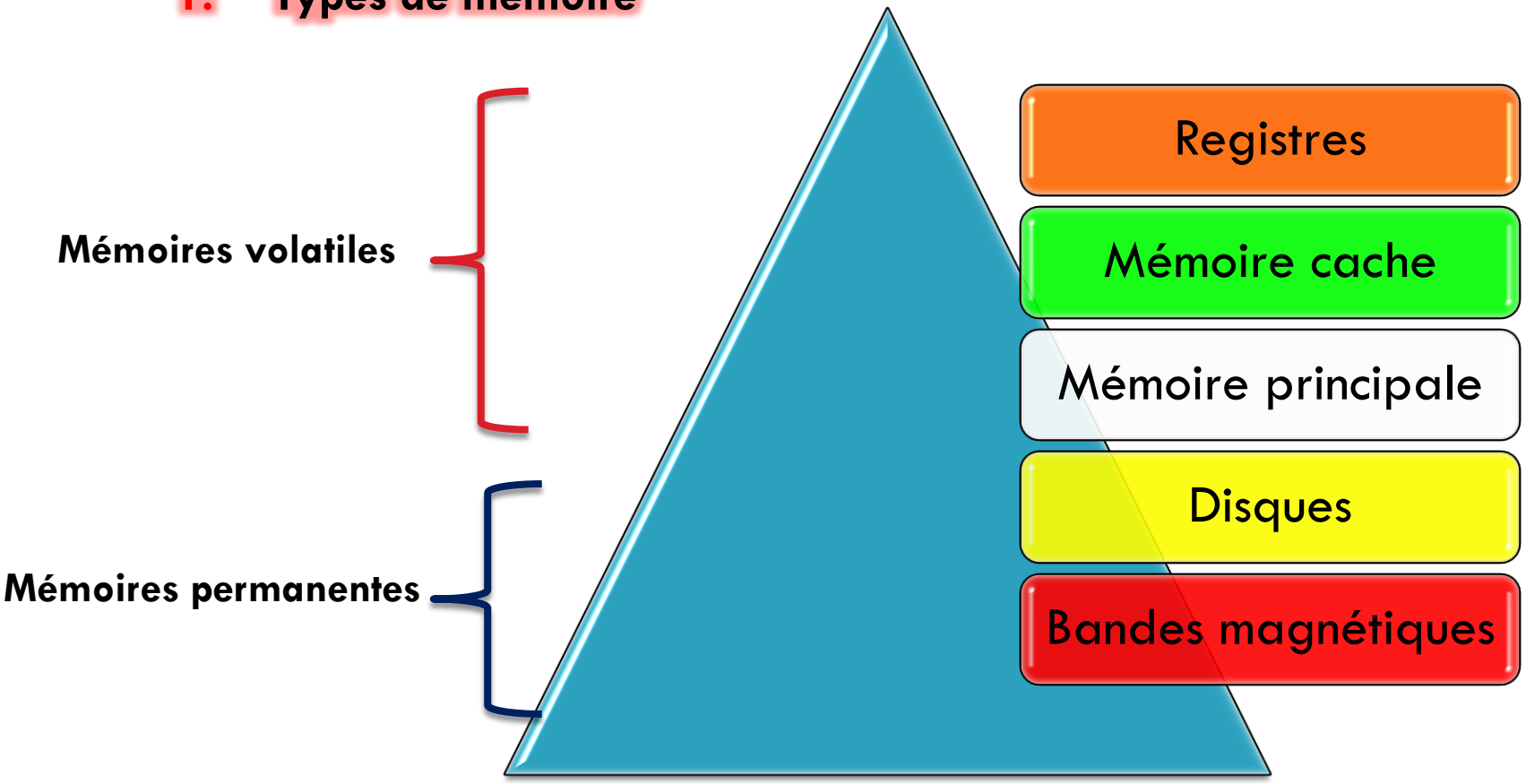
II. **Gestion de la mémoire uniforme**

III. **Gestion de la mémoire virtuelle**

IV. **Etude de Cas (Linux et Windows)**

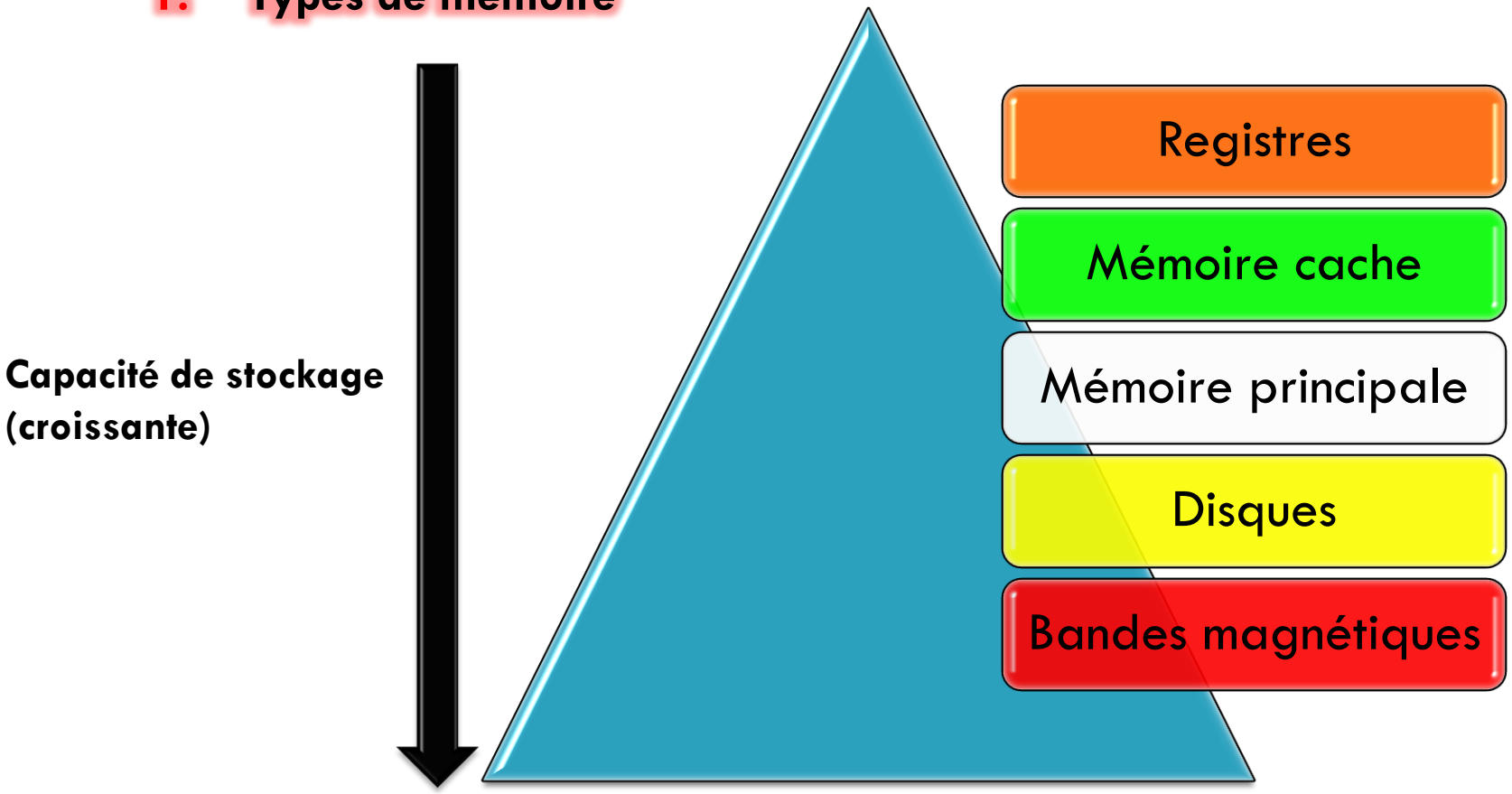
Préambule

1. Types de mémoire



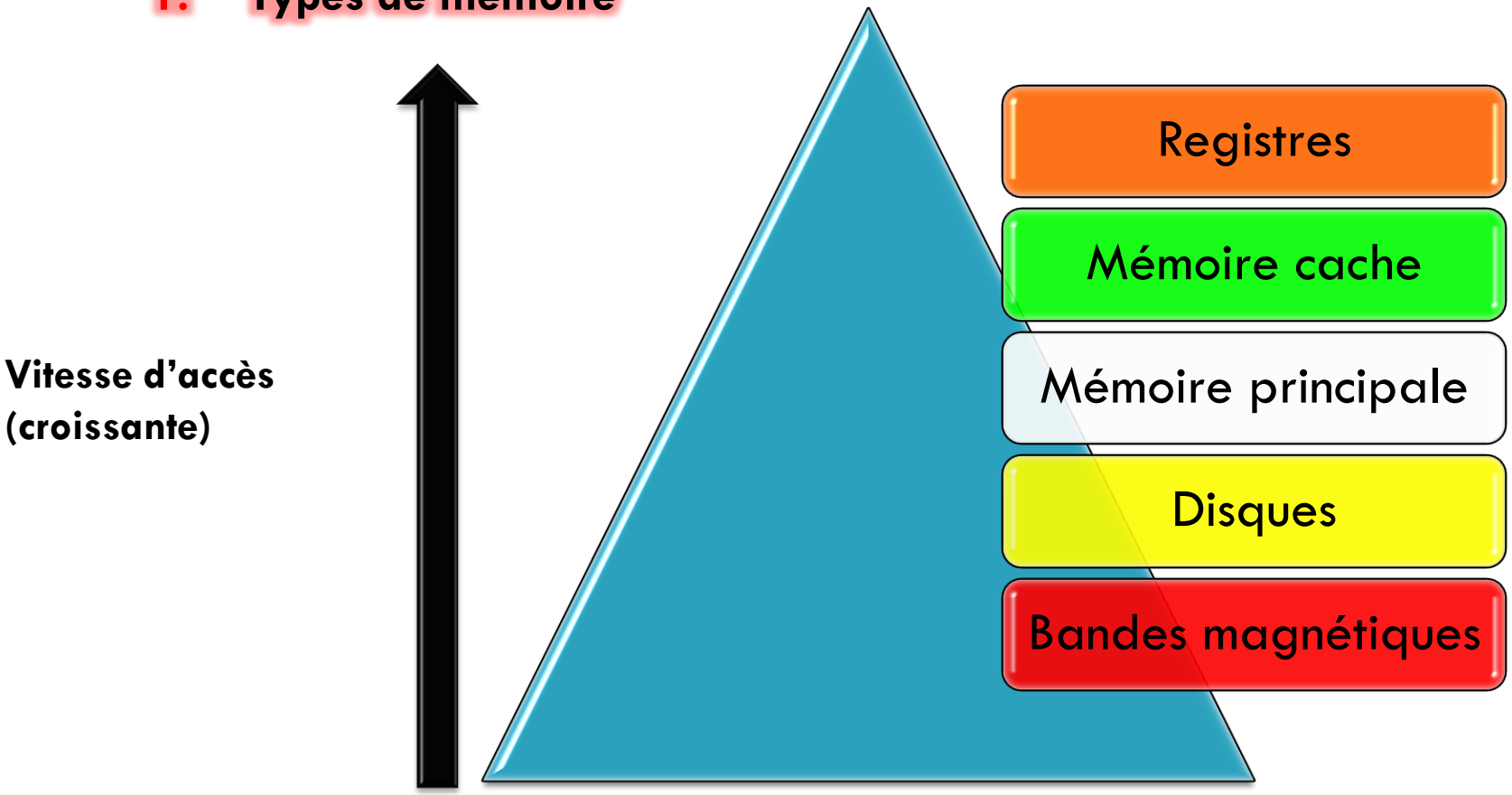
Préambule

1. Types de mémoire



Préambule

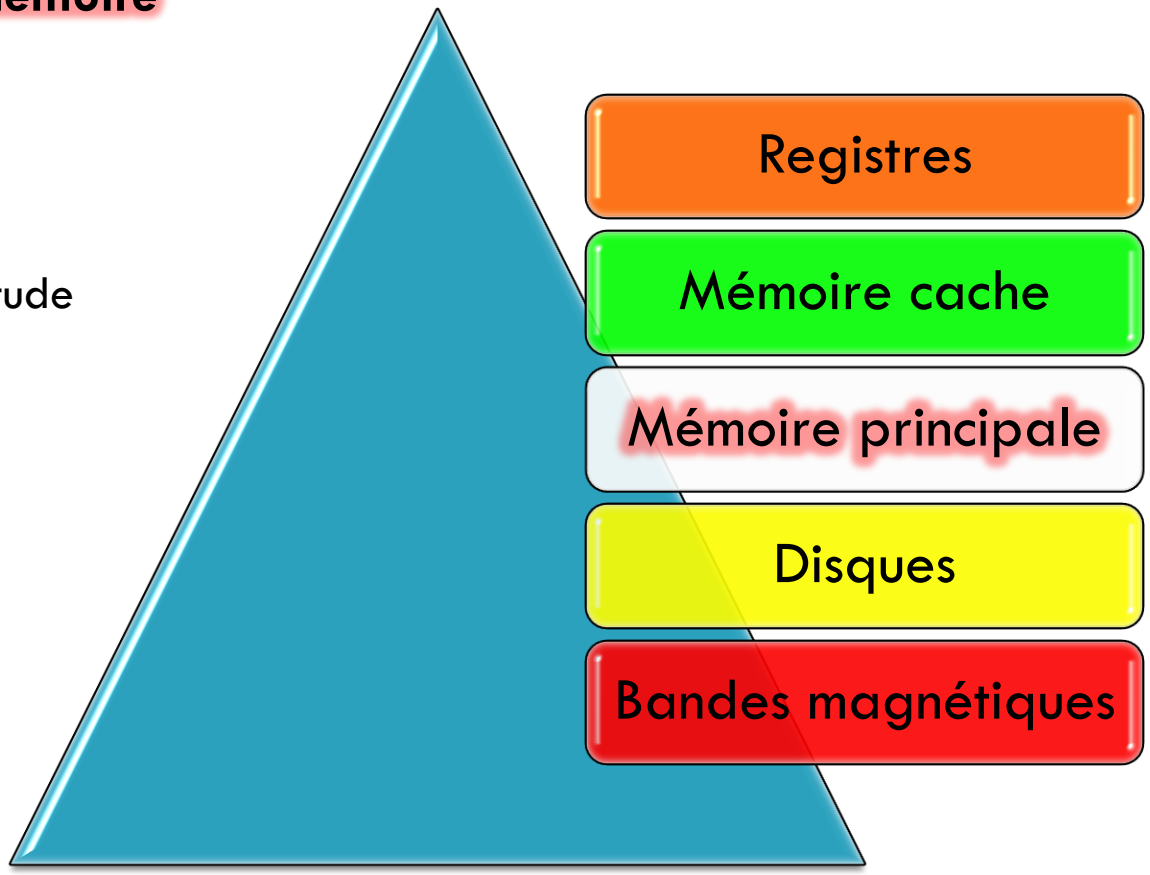
1. Types de mémoire



Préambule

1. Types de mémoire

L'objectif de ce cours est l'étude de **mémoire principale**



Préambule

2. Adressage

Mémoire principale **= RAM +ROM**

≈ RAM

≈ mémoire physique

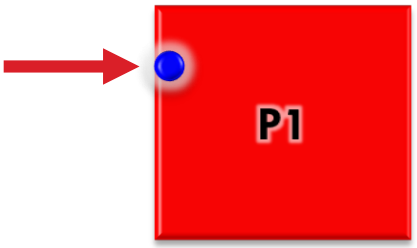
Préambule

2. Adressage

Chaque processus a un espace d'adressage dans lequel il fait ses accès aux instructions et aux données

Préambule

2. Adressage

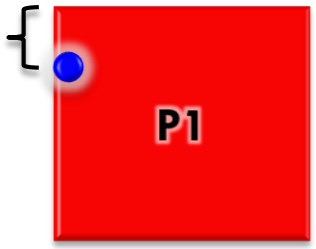


Chaque processus a un espace d'adressage dans lequel il fait ses accès aux instructions et aux données

L'accès à une instruction ou à une donnée se passe en précisant sa **référence** (position) dans cet espace

Préambule

2. Adressage



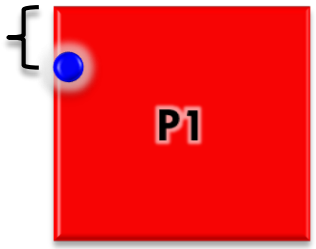
Chaque processus a un espace d'adressage dans lequel il fait ses accès aux instructions et aux données

L'accès à une instruction ou à une donnée se passe en précisant sa **référence** (position) dans cet espace

Adresse logique **est une référence par rapport au processus lui-même**

Préambule

2. Adressage



Chaque processus a un espace d'adressage dans lequel il fait ses accès aux instructions et aux données

L'accès à une instruction ou à une donnée se passe en précisant sa **référence** (position) dans cet espace

Adresse logique **est une référence par rapport au processus lui-même**

est indépendante de la position de processus en RAM

Préambule

2. Adressage

Chaque processus a un espace d'adressage dans lequel il fait ses accès aux instructions et aux données

L'accès à une instruction ou à une donnée se passe en précisant sa **référence** (position) dans cet espace

Adresse logique **est une référence par rapport au processus lui-même**

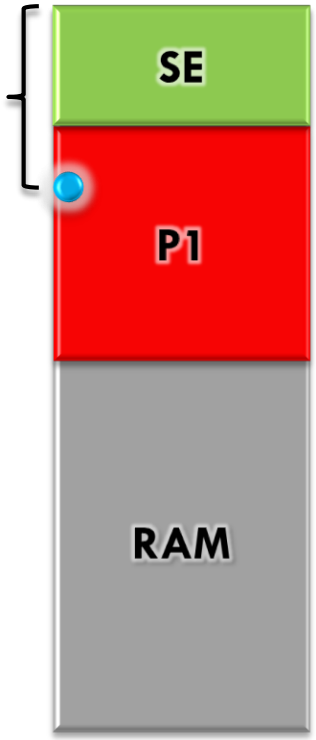
est indépendante de la position de processus en RAM

L'ensemble des adresses logiques d'un processus forme son

Espace d'adressage logique

Préambule

2. Adressage



Une fois placé en mémoire centrale, toute

adresse logique aura son correspondant en adresse physique

Ainsi, l'adresse physique désigne la position

en mémoire physique

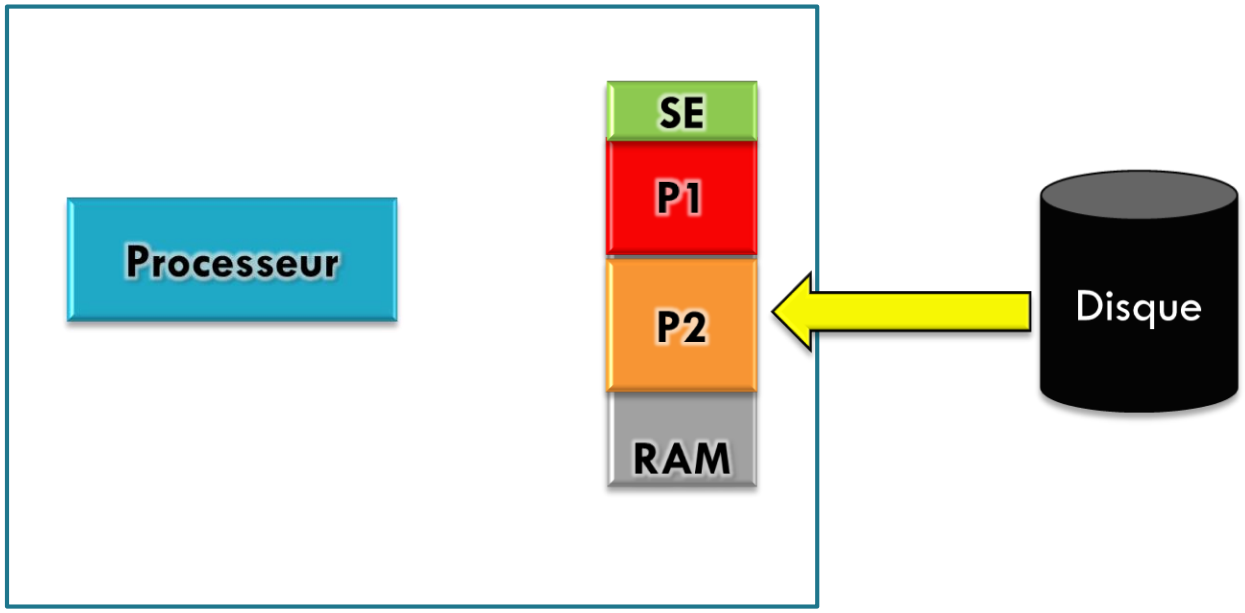
Autrement dit :

L'adresse physique est une référence par rapport à la mémoire physique (RAM)

Préambule

2. Adressage

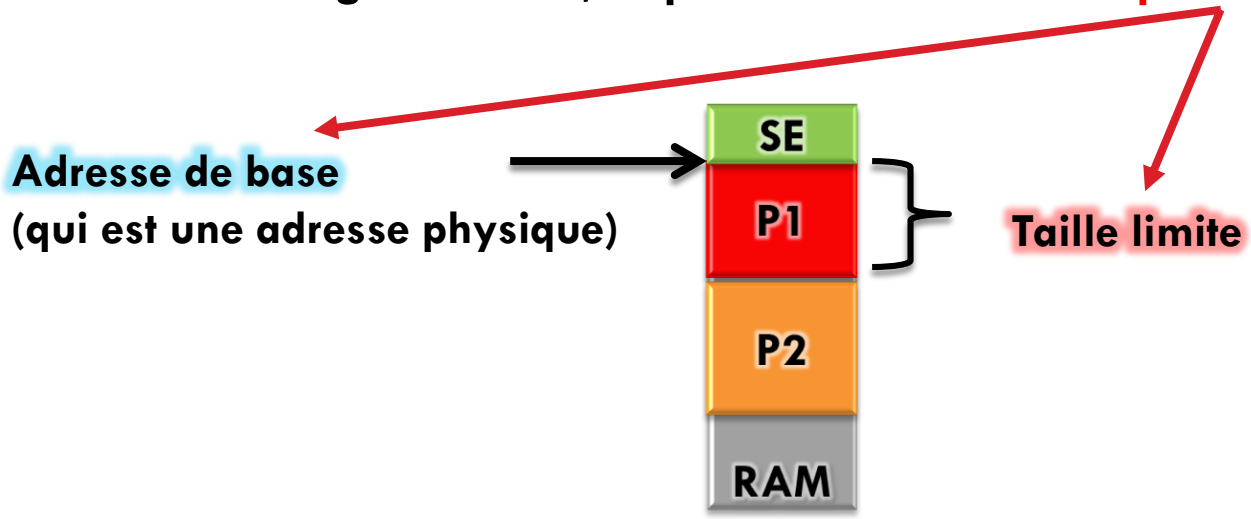
En effet, pour pouvoir s'exécuter, le processus doit être chargé en RAM



Préambule

2. Adressage

Une fois chargé en RAM, le processus aura deux **paramètres** :

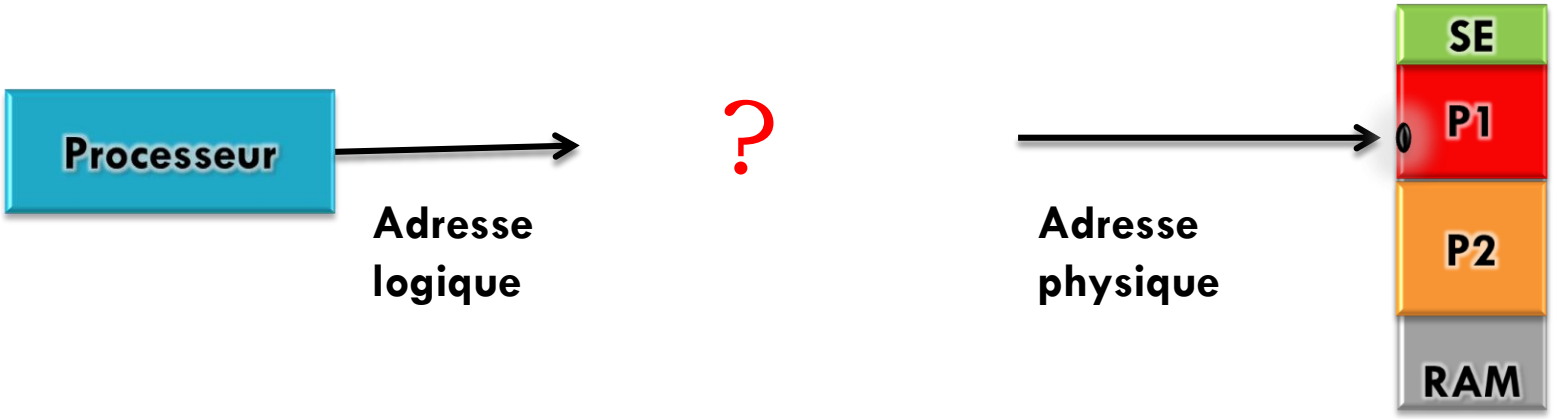


Préambule

2. Adressage

Cependant, le processeur travaille souvent avec les adresses logiques.

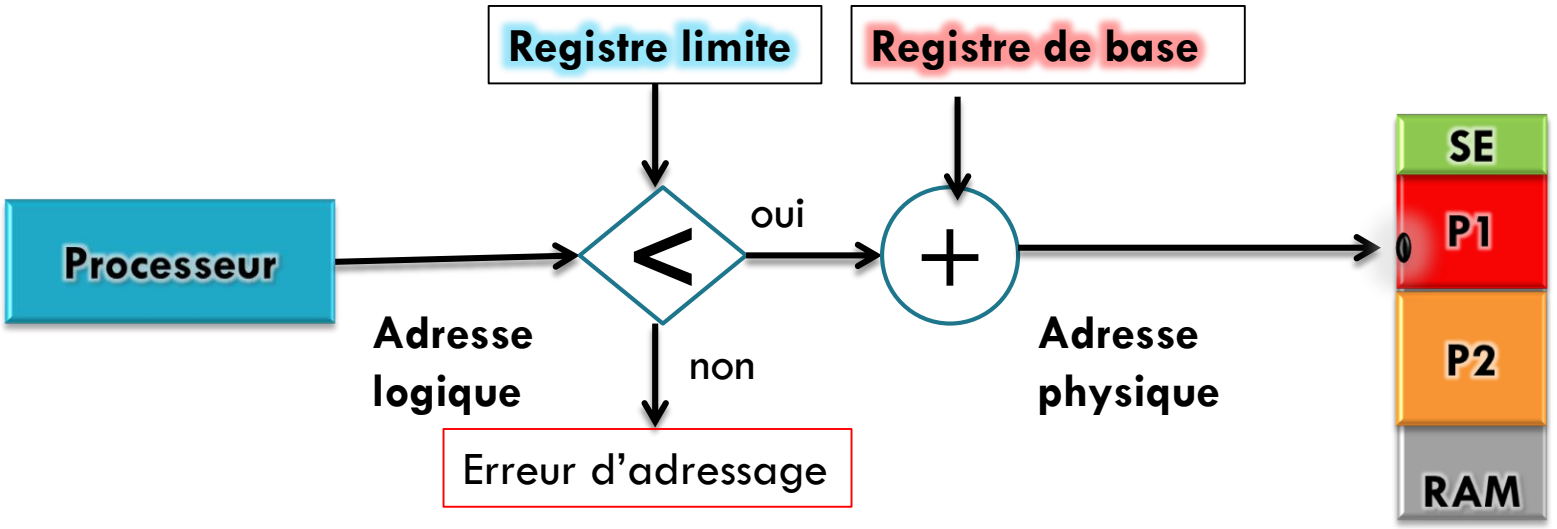
La question qui se pose est Comment traduire cette @ logique en une @ physique



Préambule

2. Adressage

Pour ce faire, on va se servir de :

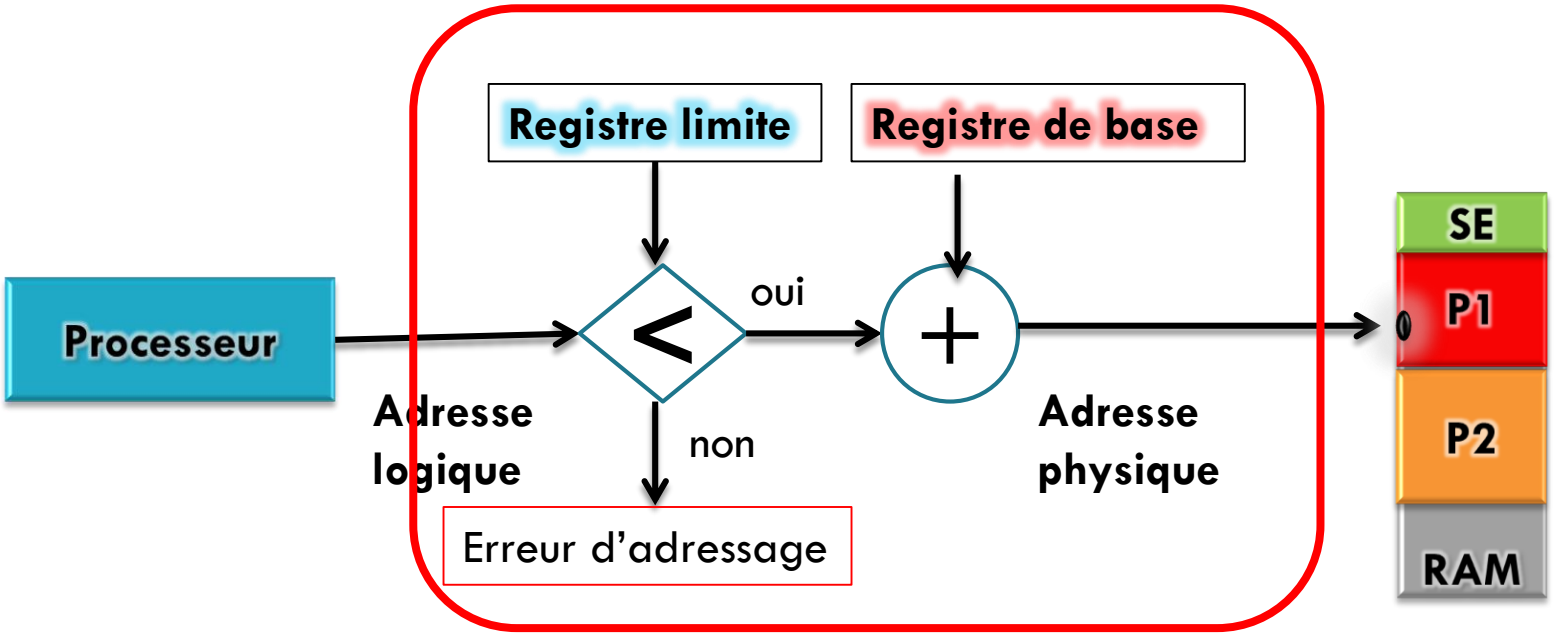


Registre limite contient la taille limite du processus actif et indique la taille de son espace d'adressage.

Registre de base contient l'adresse de base du processus actif et indique son adresse de début en RAM.

Préambule

2. Adressage



MMU (Memory Manager Unit)

Ce dispositif qui fait la transformation d'adresse est dit MMU

Gestion de la mémoire uniforme

La gestion de la mémoire centrale peut être faite en considérant le principe de la mémoire uniforme ou de la mémoire virtuelle.

Gestion de la mémoire uniforme

La gestion de la mémoire centrale peut être faite en considérant le principe de la mémoire uniforme ou de la mémoire virtuelle.

La gestion de la mémoire uniforme suppose que la taille de processus ne doit pas dépasser la taille de la mémoire physique.

Gestion de la mémoire uniforme

La gestion de la mémoire centrale peut être faite en considérant le principe de la mémoire uniforme ou de la mémoire virtuelle.

La gestion de la mémoire uniforme suppose que la taille de processus ne doit pas dépasser la taille de la mémoire physique.

Dans cette partie, on va voir la gestion de la mémoire uniforme et ce dans les deux cas suivants :

- **Cas de la monoprogrammation**
- **Cas de la multiprogrammation**

Gestion de la mémoire uniforme

1. Cas de la monoprogrammation

Dans ce cas

Un seul processus est chargé en RAM

Gestion de la mémoire uniforme

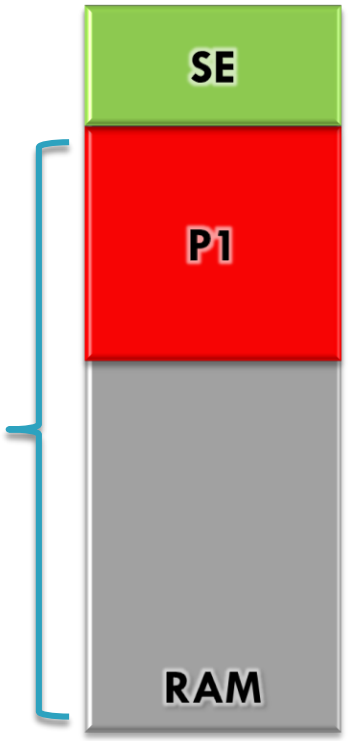
1. Cas de la monoprogrammation

a) Sans recouvrement

La gestion est faite sans recouvrement

Si la taille de processus ne dépasse pas

la taille de l'espace utilisable



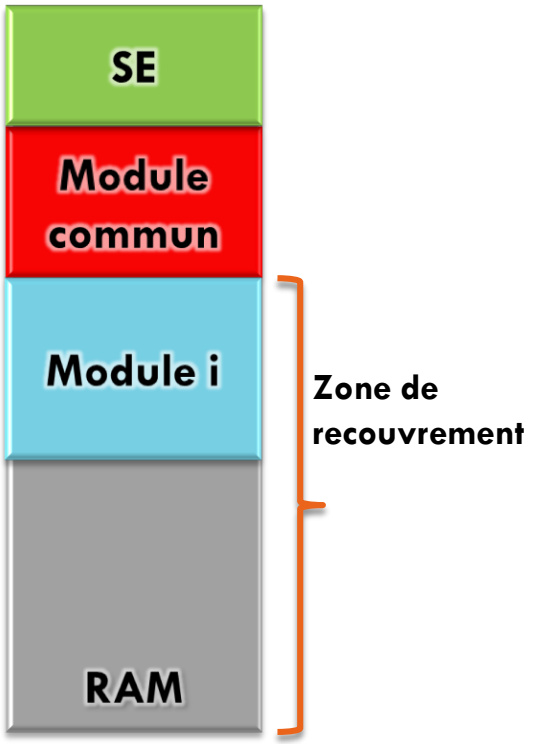
Gestion de la mémoire uniforme

1. Cas de la monoprogrammation

b) Avec recouvrement

Si la taille de processus dépasse la taille de l'espace utilisable

Alors le programmeur doit diviser son processus en des modules (overlays) :



Gestion de la mémoire uniforme

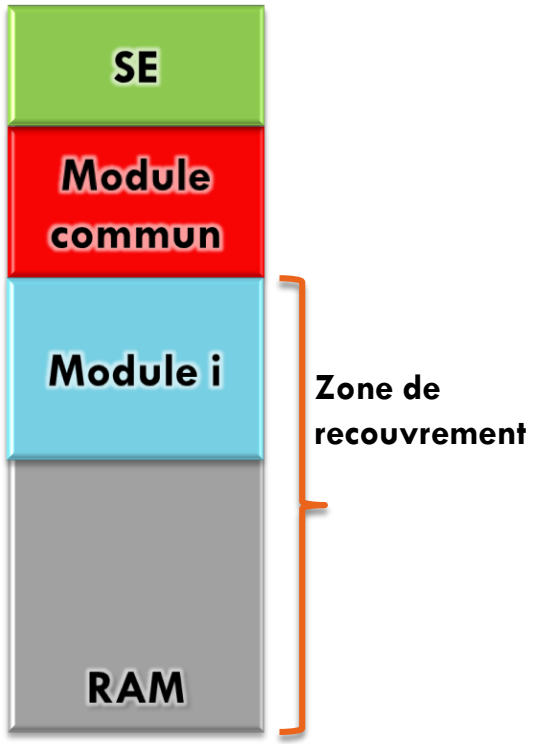
1. Cas de la monoprogrammation

b) Avec recouvrement

Si la taille de processus dépasse la taille de l'espace utilisable

Alors le programmeur doit diviser son processus en des modules (overlays) :

- module commun
- et des modules de recouvrements



Les modules de recouvrement sont chargés en RAM à la demande

Gestion de la mémoire uniforme

2. Cas de la multiprogrammation

Dans ce cas

Plusieurs processus peuvent êtres chargés en RAM

Gestion de la mémoire uniforme

2. Cas de la multiprogrammation

Dans ce cas

Plusieurs processus peuvent êtres chargés en RAM

La gestion de ces processus en RAM se fait avec :

- **des partitions fixes** ou
- **des partitions variables**

Gestion de la mémoire uniforme

2. Cas de la multiprogrammation

a) Partions fixes ou statiques

La RAM est subdivisée en des parties

- de tailles,
- de nombres
- et de localisations

Fixes.

Gestion de la mémoire uniforme

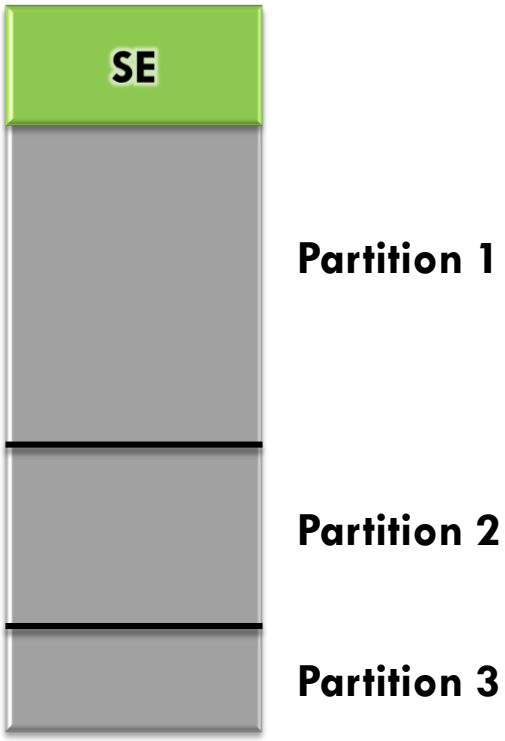
2. Cas de la multiprogrammation

a) Partions fixes ou statiques

La RAM est subdivisée en des parties

- de tailles,
- de nombres
- et de localisations

Fixes.

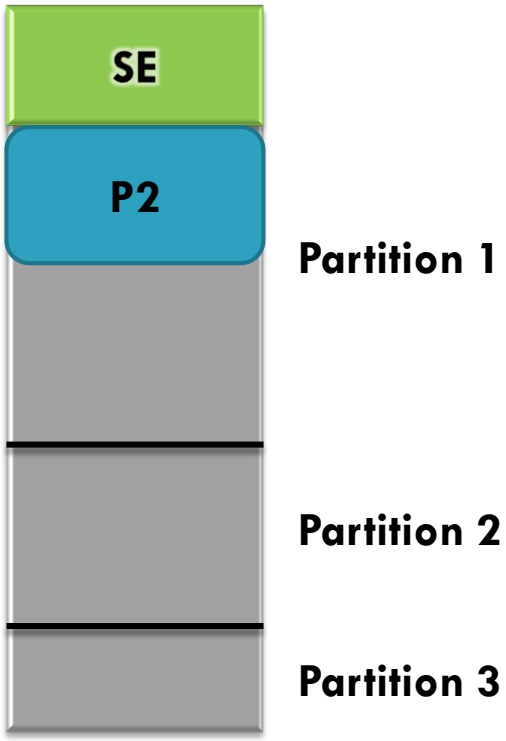


Gestion de la mémoire uniforme

2. Cas de la multiprogrammation

a) Partions fixes ou statiques

Chaque partition peut recevoir au maximum un processus.



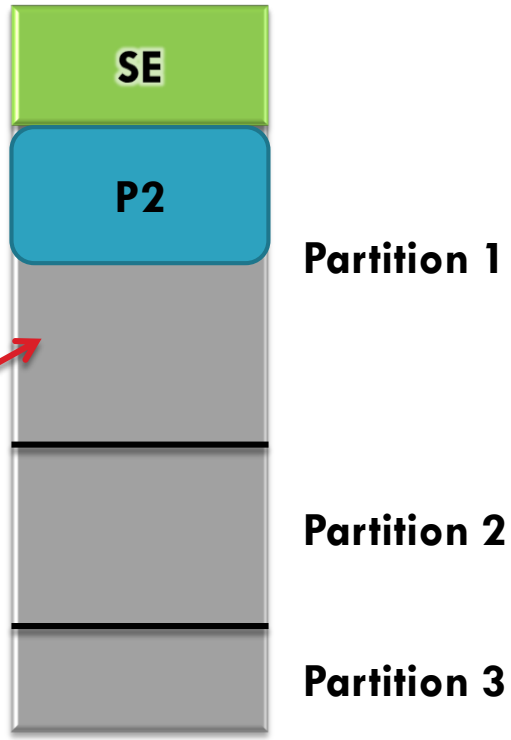
Gestion de la mémoire uniforme

2. Cas de la multiprogrammation

a) Partions fixes ou statiques

Chaque partition peut recevoir au maximum un processus.

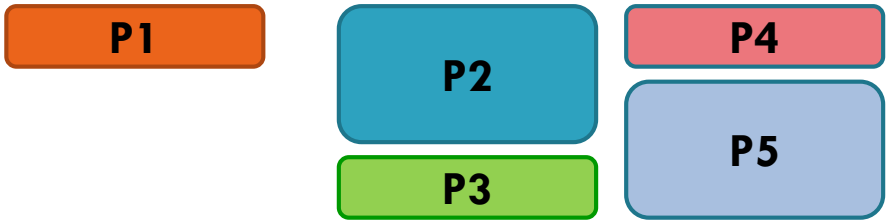
Risque de fragmentation interne



Gestion de la mémoire uniforme

2. Cas de la multiprogrammation

a) Partions fixes ou statiques



Si on a plusieurs processus comment

les répartir entre les partitions ?

Gestion de la mémoire uniforme

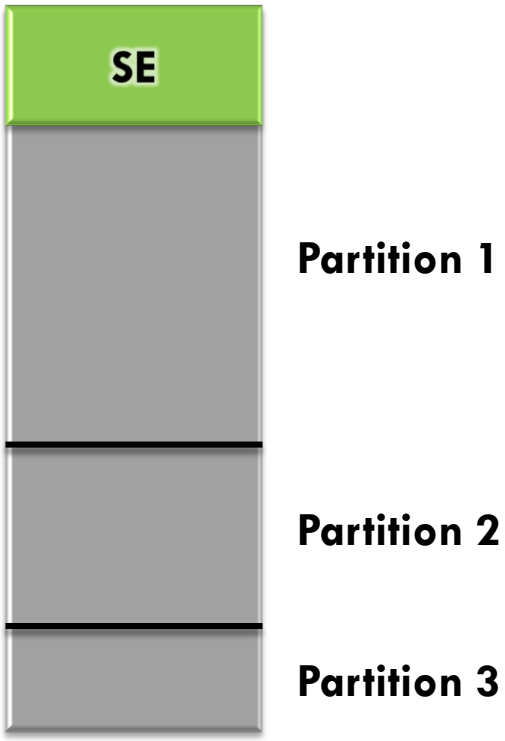
2. Cas de la multiprogrammation

a) Partions fixes ou statiques

- Une file d'attente par partition

Chaque processus est affecté à la file

de la plus petite partition possible



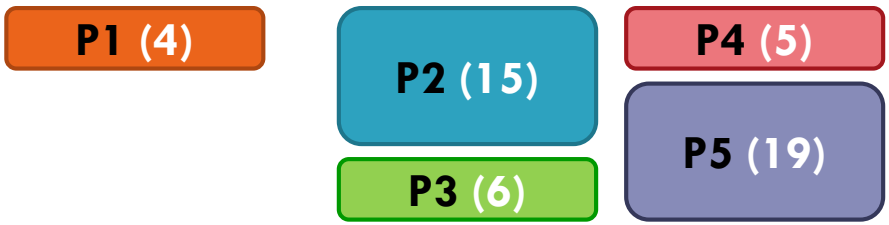
Gestion de la mémoire uniforme

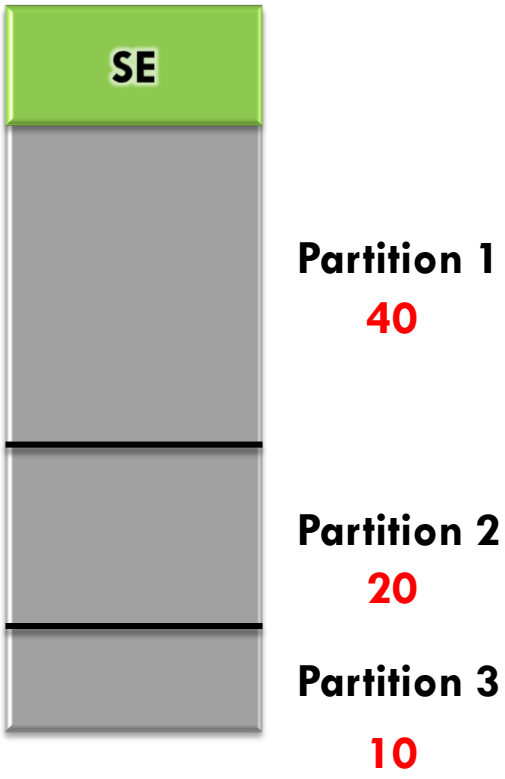
2. Cas de la multiprogrammation

a) Partions fixes ou statiques

- Une file d'attente par partition

Pour l'exemple suivant où doit-on donc placer les processus ?





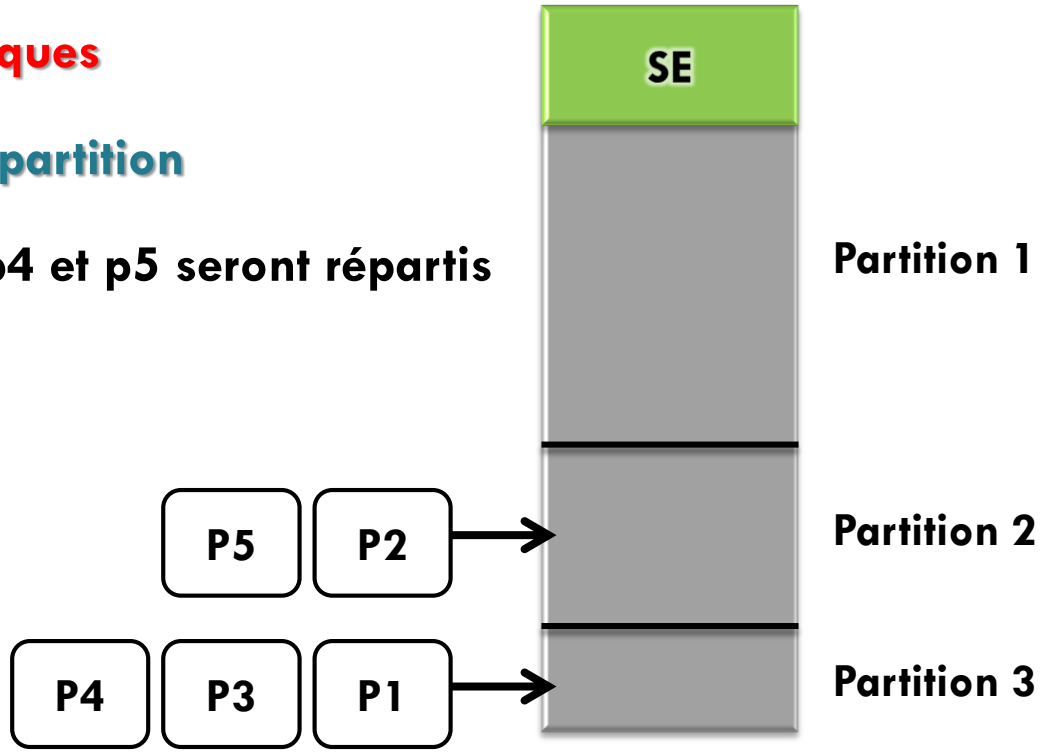
Gestion de la mémoire uniforme

2. Cas de la multiprogrammation

a) Partions fixes ou statiques

❖ **Une file d'attente par partition**

Les processus p1, p2, p3, p4 et p5 seront répartis comme suit :



Gestion de la mémoire uniforme

2. Cas de la multiprogrammation

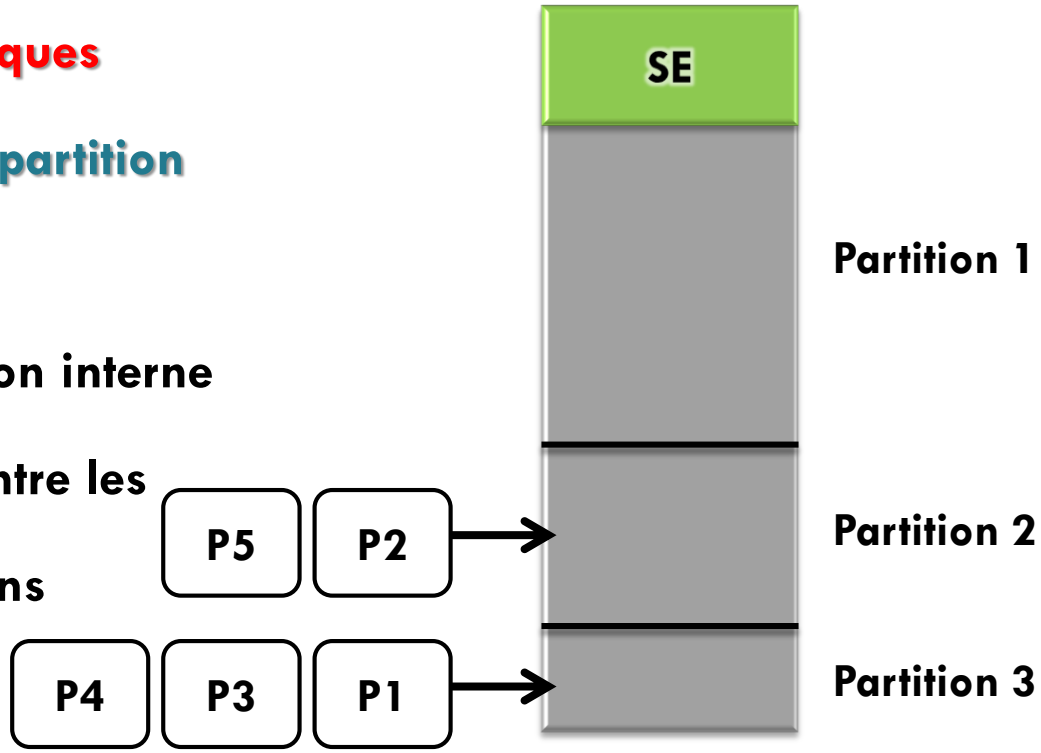
a) Partions fixes ou statiques

❖ **Une file d'attente par partition**

Cette répartition :

+ Minimise la fragmentation interne

- Risque un déséquilibre entre les files d'attentes des partitions (certaines sont pleines, d'autres sont vides)



Gestion de la mémoire uniforme

2. Cas de la multiprogrammation

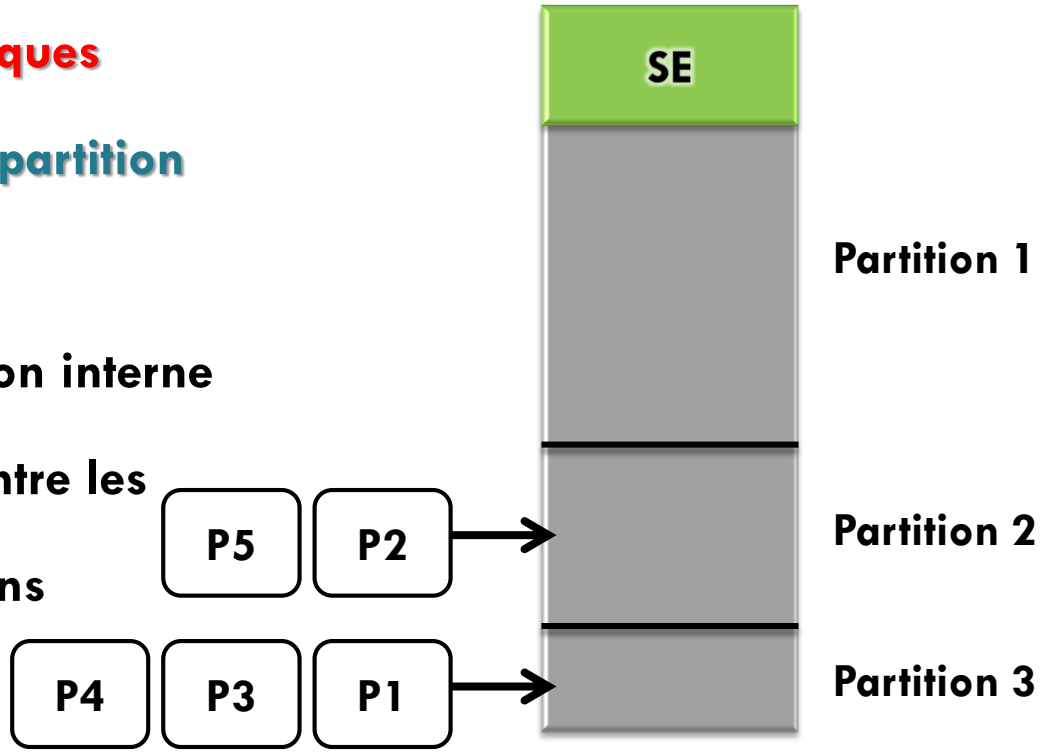
a) Partions fixes ou statiques

❖ **Une file d'attente par partition**

Cette répartition :

+ Minimise la fragmentation interne

- Risque un déséquilibre entre les files d'attentes des partitions (certaines sont pleines, d'autres sont vides)



Gestion de la mémoire uniforme

2. Cas de la multiprogrammation

a) Partions fixes ou statiques

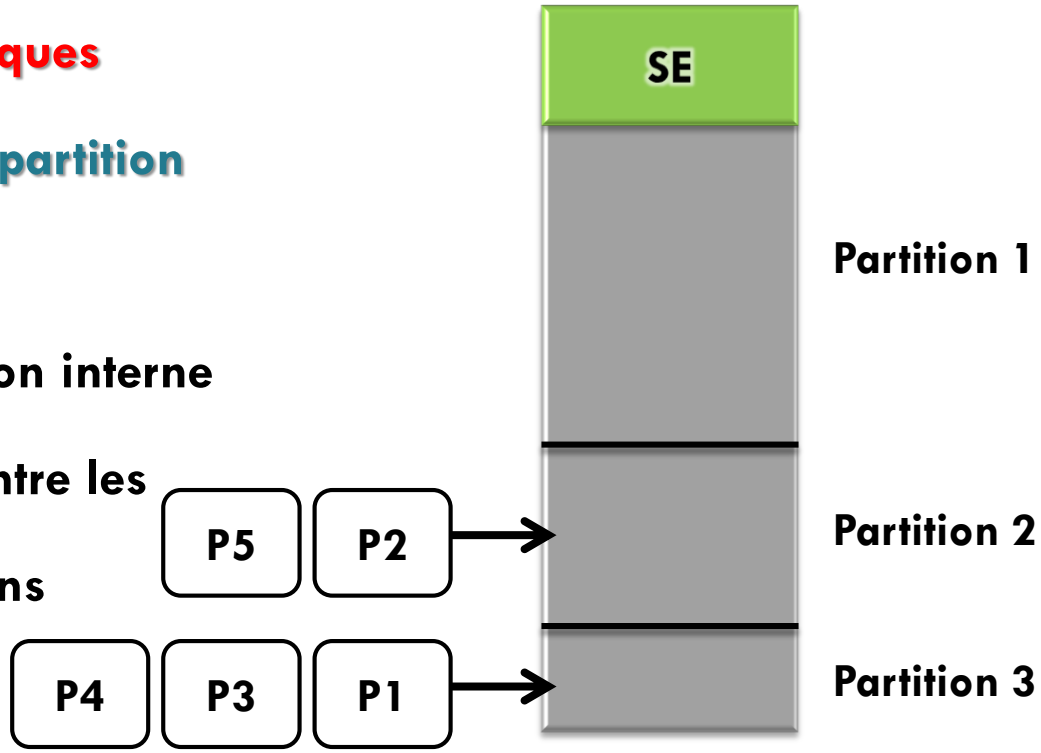
❖ Une file d'attente par partition

Cette répartition :

+ Minimise la fragmentation interne

- Risque un déséquilibre entre les files d'attentes des partitions

La partition 1 est vide et peut recevoir des processus

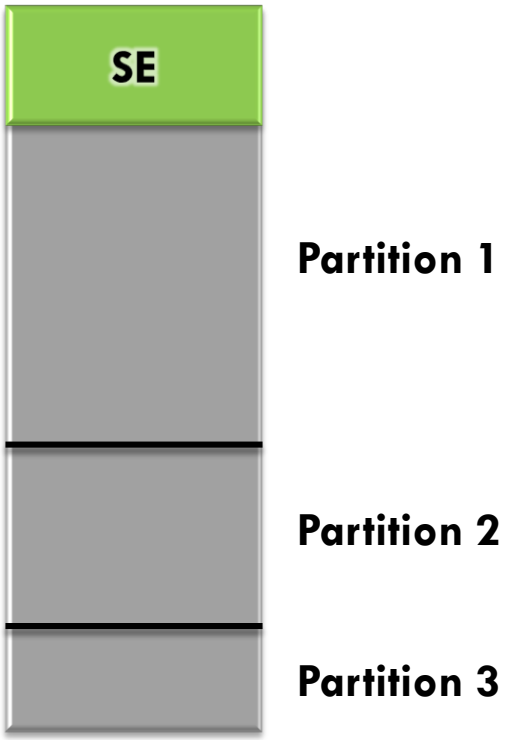


Gestion de la mémoire uniforme

2. Cas de la multiprogrammation

a) Partions fixes ou statiques

- Une file d'attente pour toutes les partitions



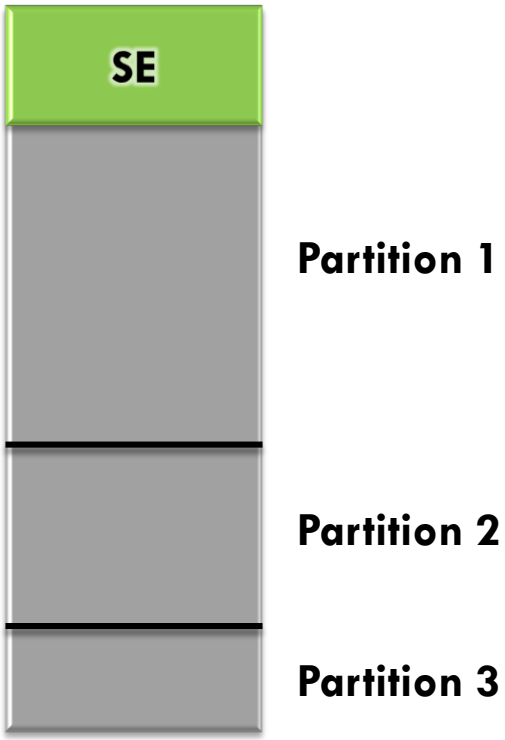
Gestion de la mémoire uniforme

2. Cas de la multiprogrammation

a) Partions fixes ou statiques

- Une file d'attente pour toutes les partitions

Comment répartir maintenant les processus?



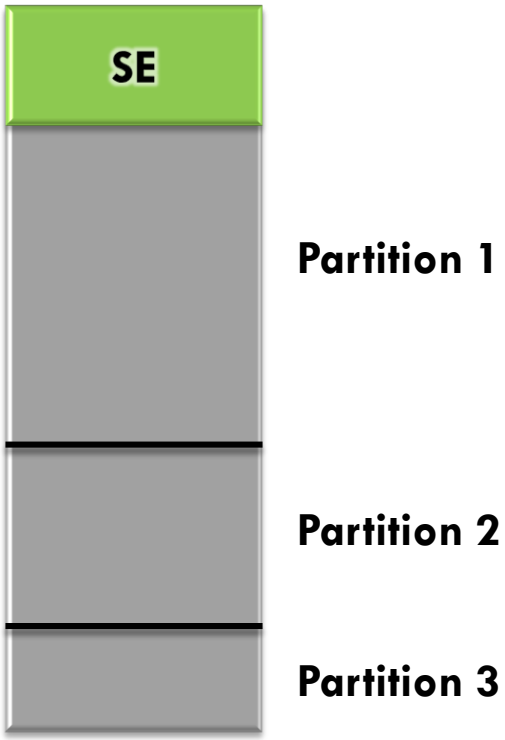
Gestion de la mémoire uniforme

2. Cas de la multiprogrammation

a) Partions fixes ou statiques

❖ Une file d'attente pour toutes les partitions

➢ Dès qu'une partition est libre, on lui affecte le premier processus possible

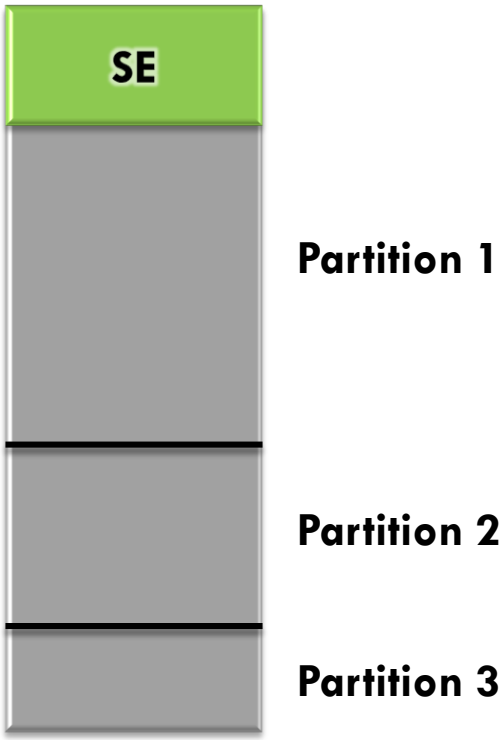


Gestion de la mémoire uniforme

2. Cas de la multiprogrammation

a) Partions fixes ou statiques

❖ Une file d'attente pour toutes les partitions



➢ Dès qu'une partition est libre, on lui affecte le premier processus possible

- Consommation des grandes partitions par les processus de petites tailles (risque d'augmenter la fragmentation interne)

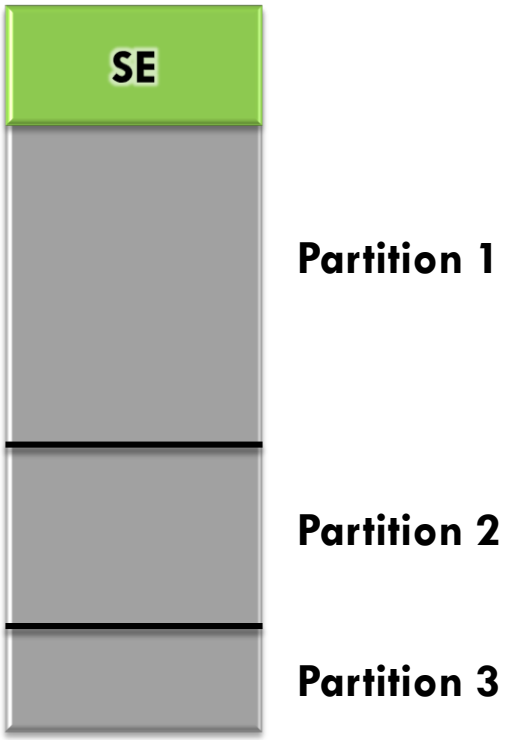
Gestion de la mémoire uniforme

2. Cas de la multiprogrammation

a) Partions fixes ou statiques

- Une file d'attente pour toutes les partitions

- Dès qu'une partition est libre, on lui affecte le plus grand processus possible



Gestion de la mémoire uniforme

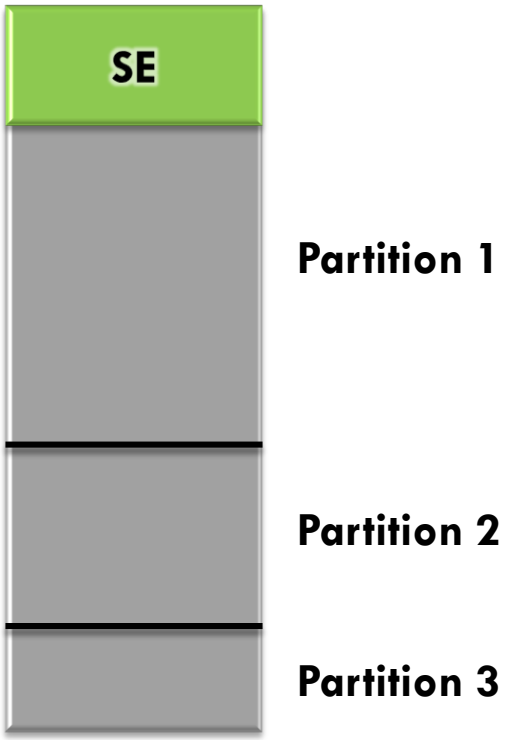
2. Cas de la multiprogrammation

a) Partions fixes ou statiques

❖ Une file d'attente pour toutes les partitions

➢ Dès qu'une partition est libre, on lui affecte le plus grand processus possible

- Risque de famine pour les processus de petites tailles



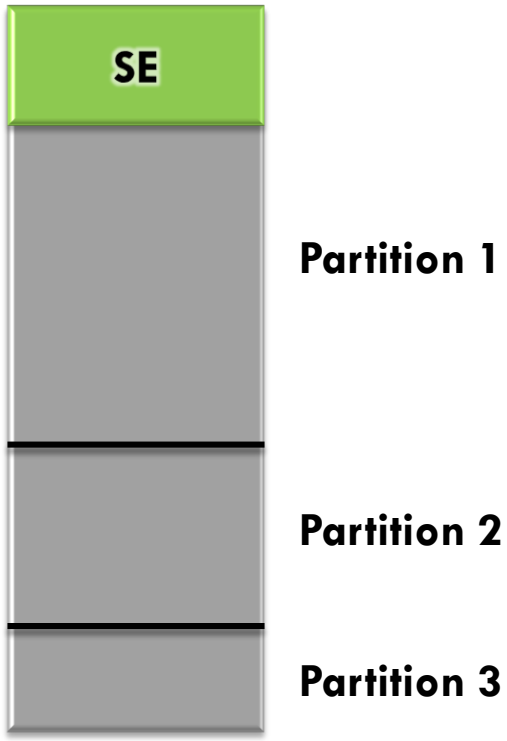
Gestion de la mémoire uniforme

2. Cas de la multiprogrammation

a) Partions fixes ou statiques

Si la taille d'un processus dépasse la taille de toutes les partitions

Alors on applique le recouvrement



Gestion de la mémoire uniforme

2. Cas de la multiprogrammation

b) Partions variables ou dynamiques

Chaque processus aura un espace mémoire égal à son besoin dit partition.

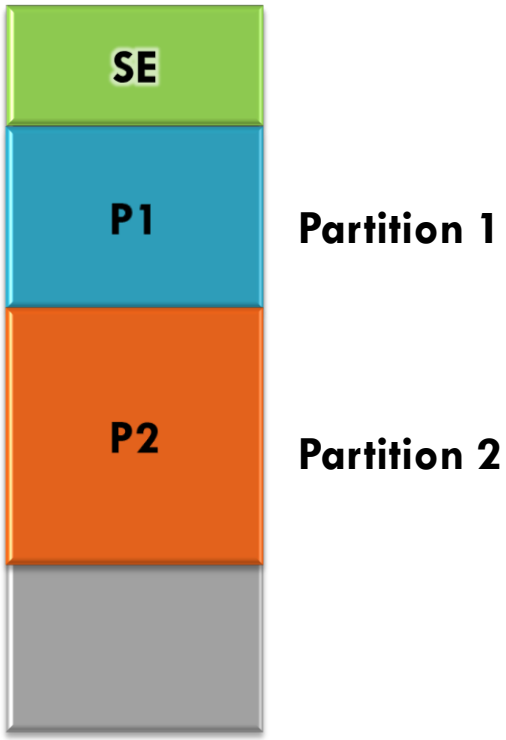
Gestion de la mémoire uniforme

2. Cas de la multiprogrammation

b) Partions variables ou dynamiques

Chaque processus aura un espace mémoire égal à son besoin dit partition.

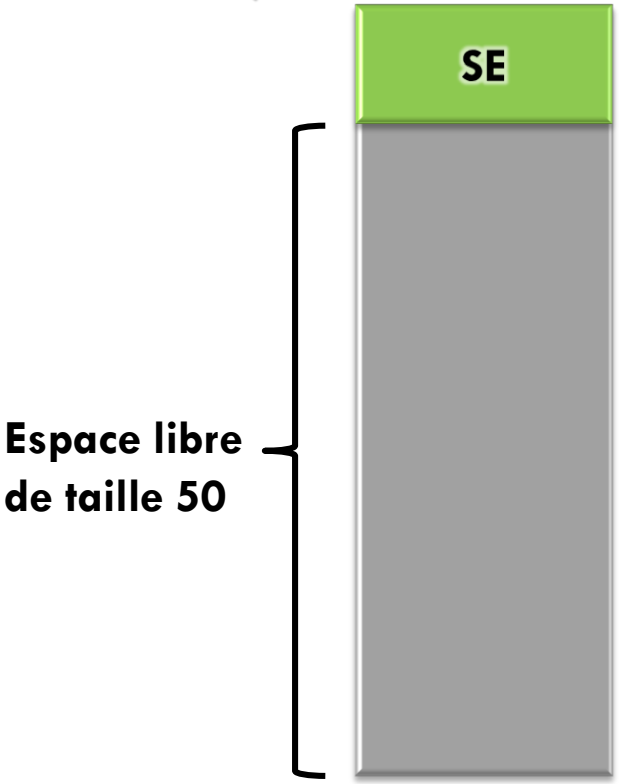
Pas de fragmentation interne



Gestion de la mémoire uniforme

2. Cas de la multiprogrammation

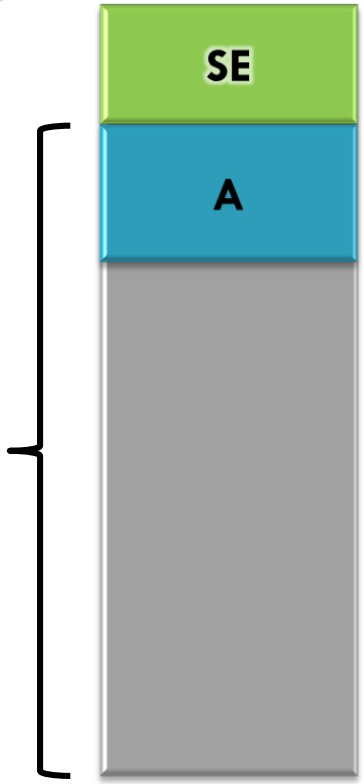
b) Partions variables ou dynamiques (exemple)



Gestion de la mémoire uniforme

2. Cas de la multiprogrammation

b) Partions variables ou dynamiques (exemple)



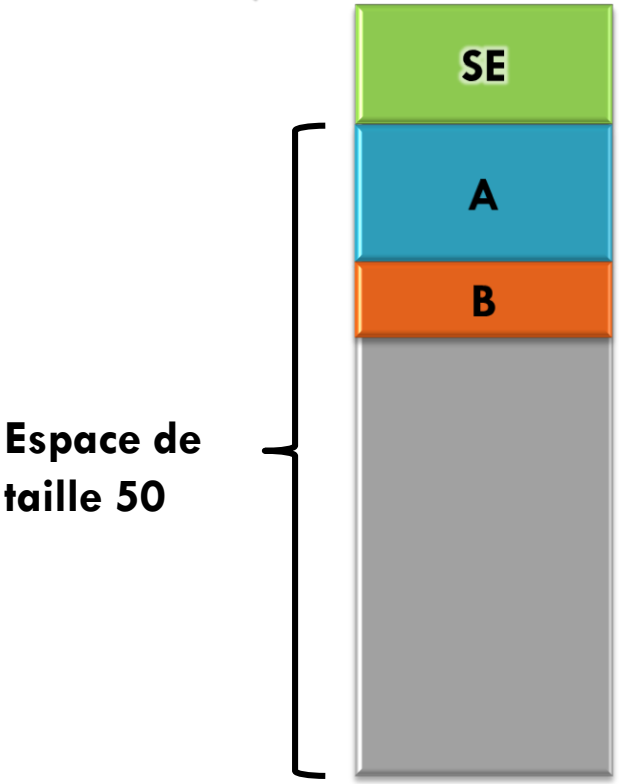
Arrivée de A de taille 10

Espace de taille 50

Gestion de la mémoire uniforme

2. Cas de la multiprogrammation

b) Partions variables ou dynamiques (exemple)

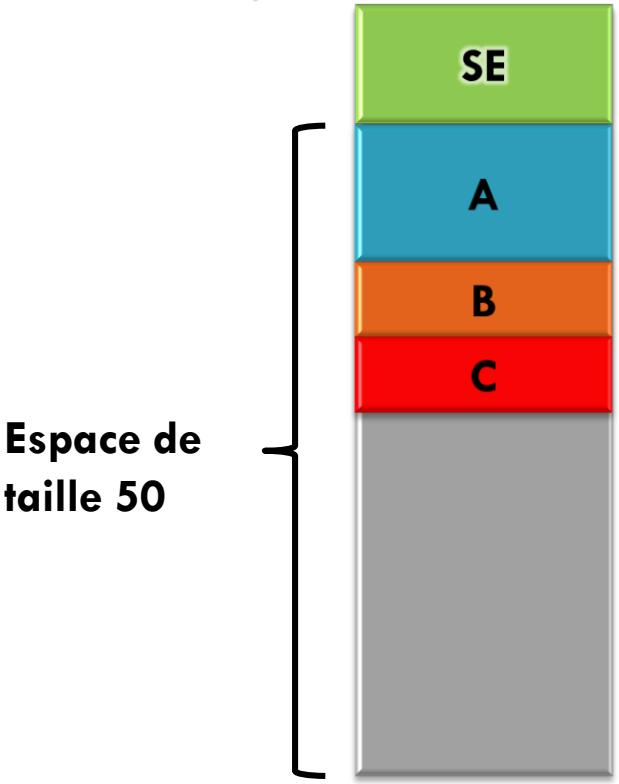


Arrivée de B de taille 5

Gestion de la mémoire uniforme

2. Cas de la multiprogrammation

b) Partions variables ou dynamiques (exemple)

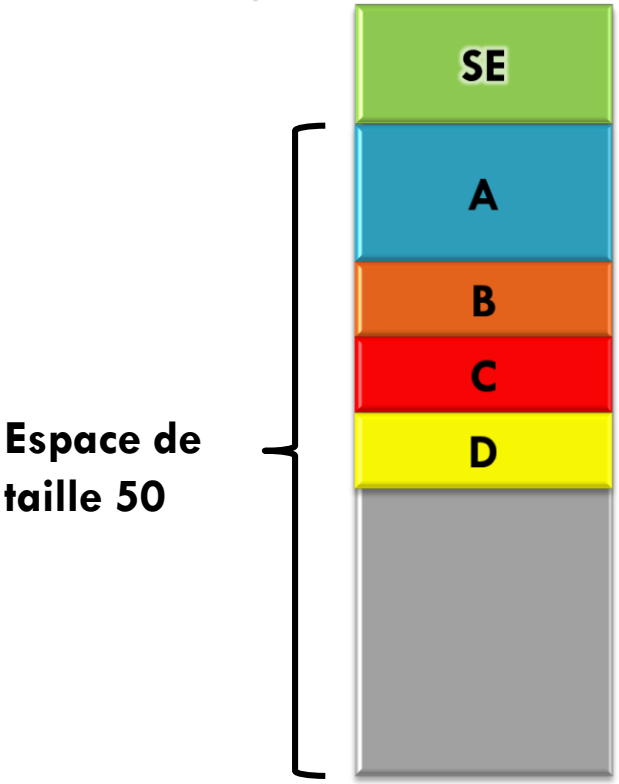


Arrivée de C de taille 5

Gestion de la mémoire uniforme

2. Cas de la multiprogrammation

b) Partions variables ou dynamiques (exemple)

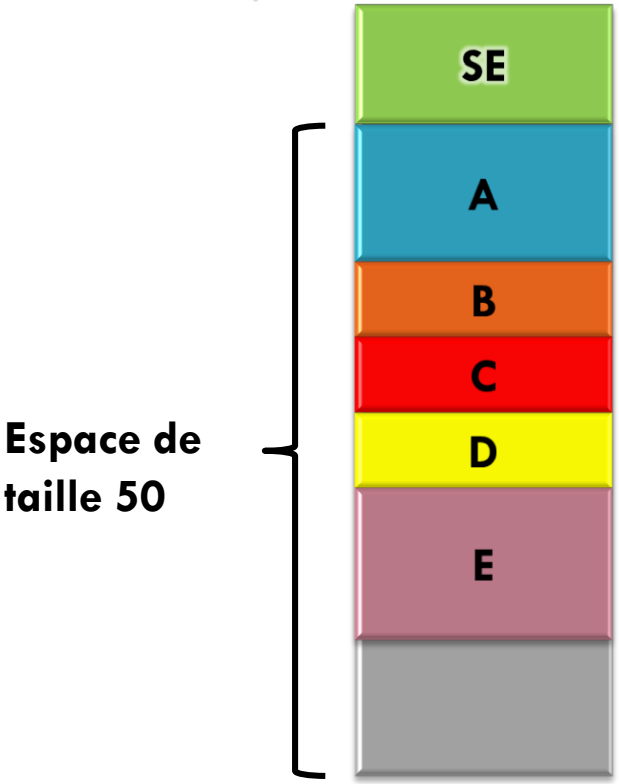


Arrivée de D de taille 5

Gestion de la mémoire uniforme

2. Cas de la multiprogrammation

b) Partions variables ou dynamiques (exemple)

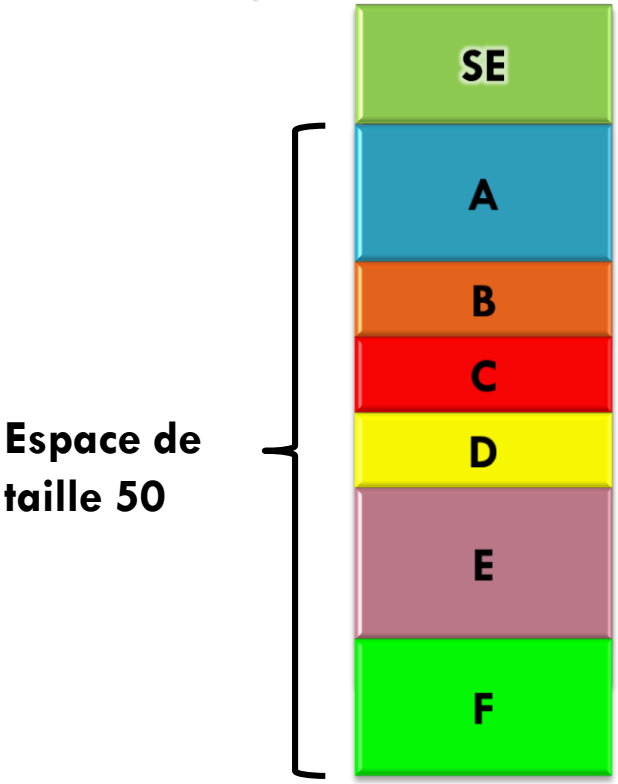


Arrivée de E de taille 15

Gestion de la mémoire uniforme

2. Cas de la multiprogrammation

b) Partions variables ou dynamiques (exemple)

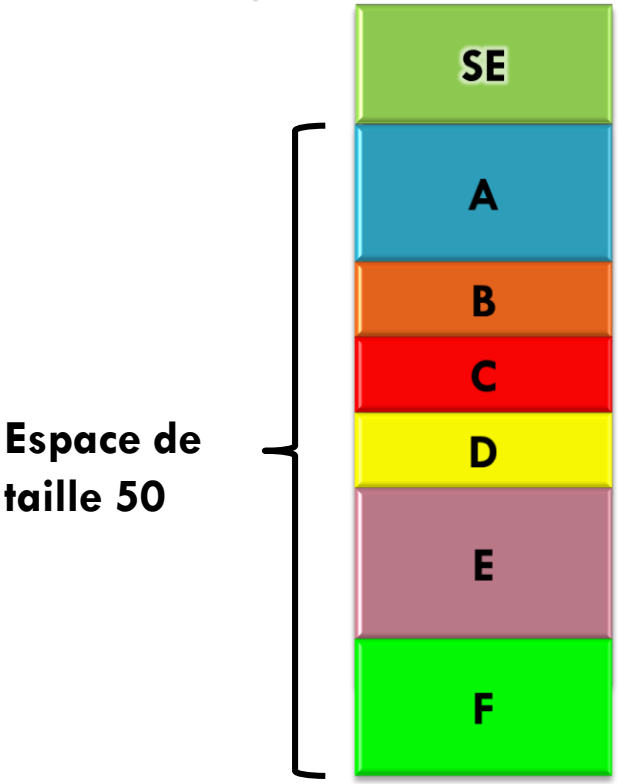


Arrivée de F de taille 10

Gestion de la mémoire uniforme

2. Cas de la multiprogrammation

b) Partions variables ou dynamiques (exemple)

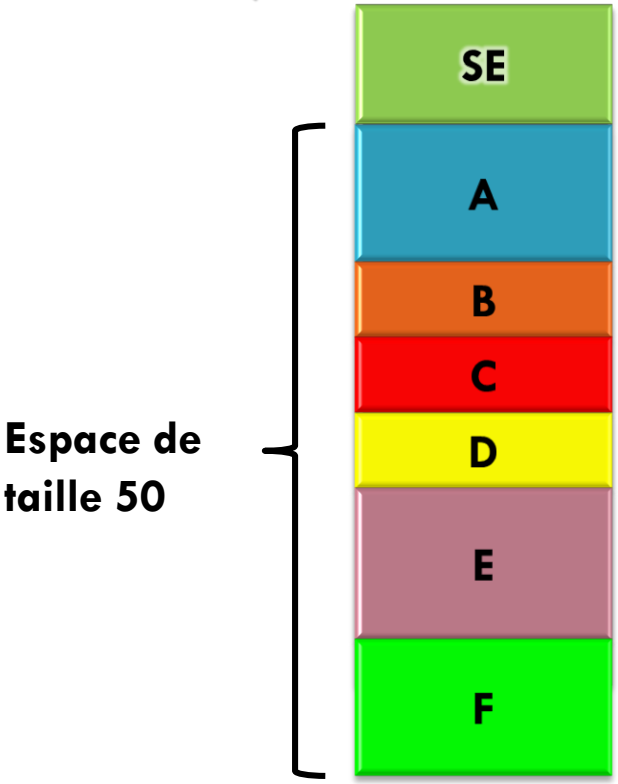


Arrivée de F de taille 10

Gestion de la mémoire uniforme

2. Cas de la multiprogrammation

b) Partions variables ou dynamiques (exemple)



La RAM contient des partitions

de tailles,

de nombres

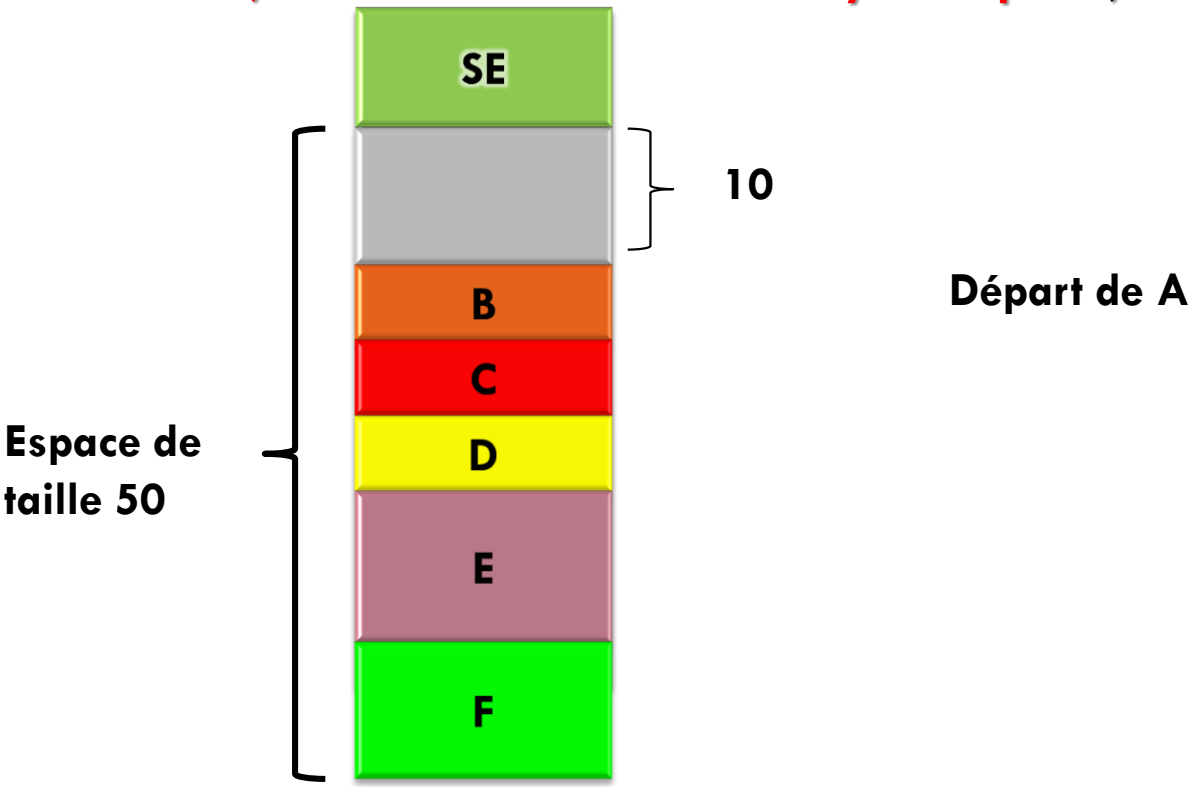
et de localisations

Variables.

Gestion de la mémoire uniforme

2. Cas de la multiprogrammation

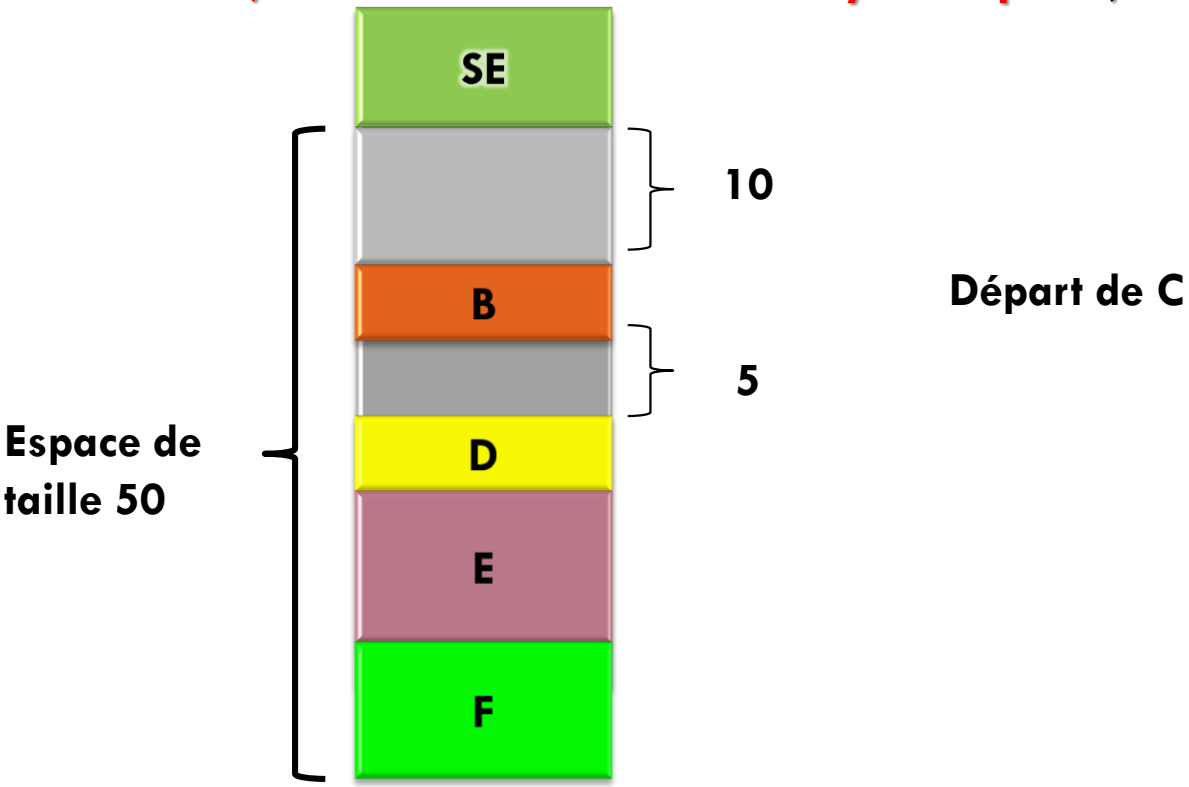
b) Partions variables ou dynamiques (exemple)



Gestion de la mémoire uniforme

2. Cas de la multiprogrammation

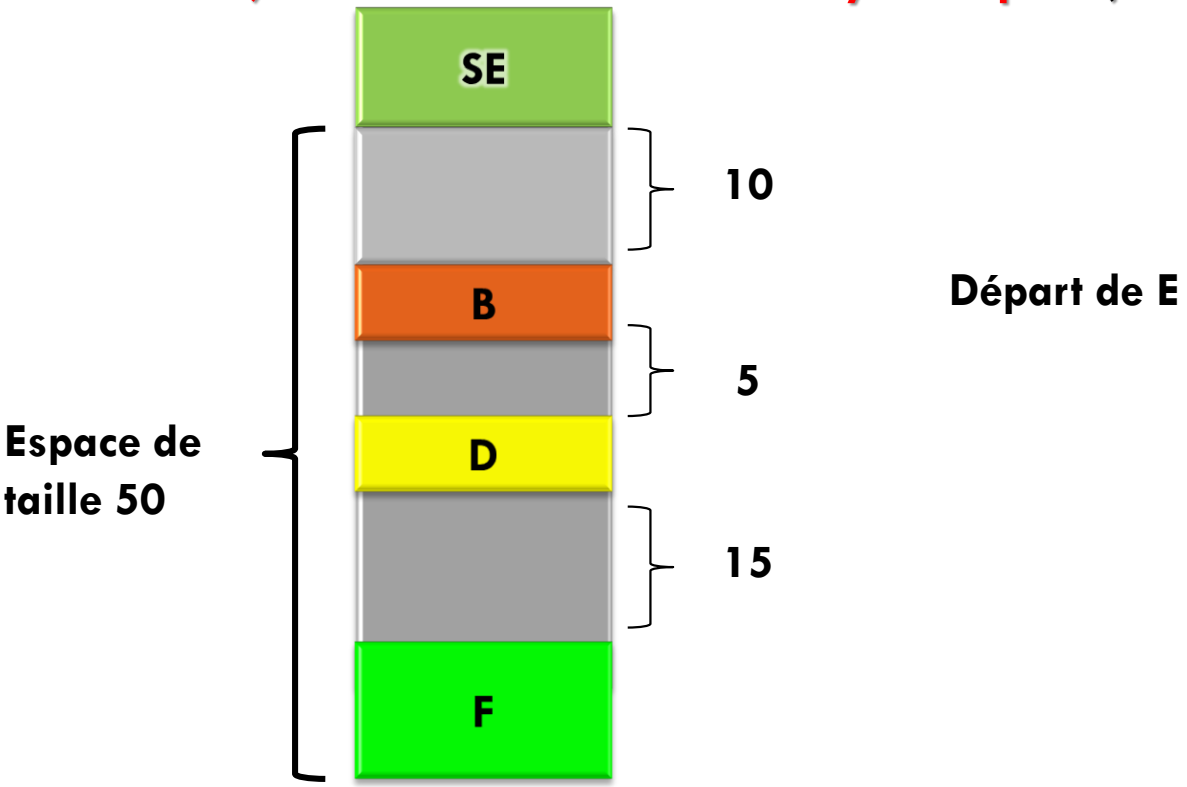
b) Partions variables ou dynamiques (exemple)



Gestion de la mémoire uniforme

2. Cas de la multiprogrammation

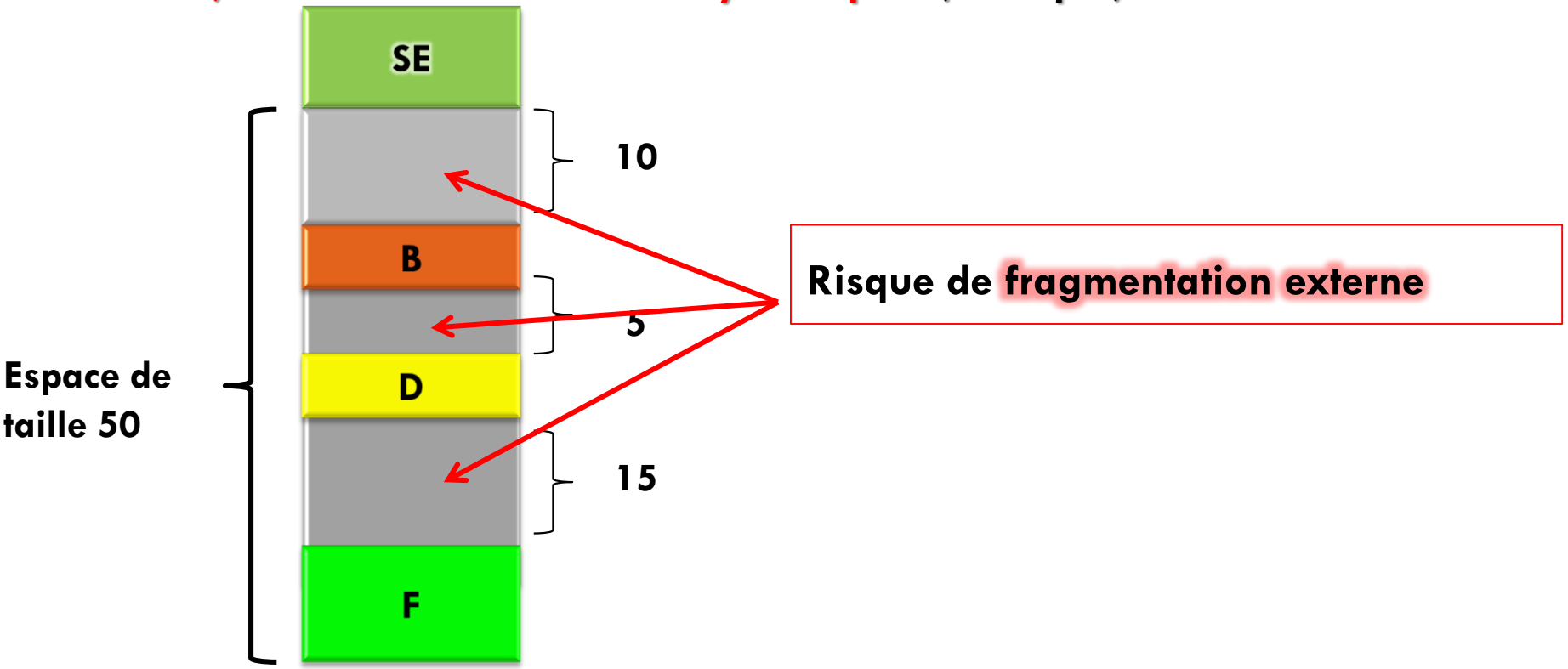
b) Partions variables ou dynamiques (exemple)



Gestion de la mémoire uniforme

2. Cas de la multiprogrammation

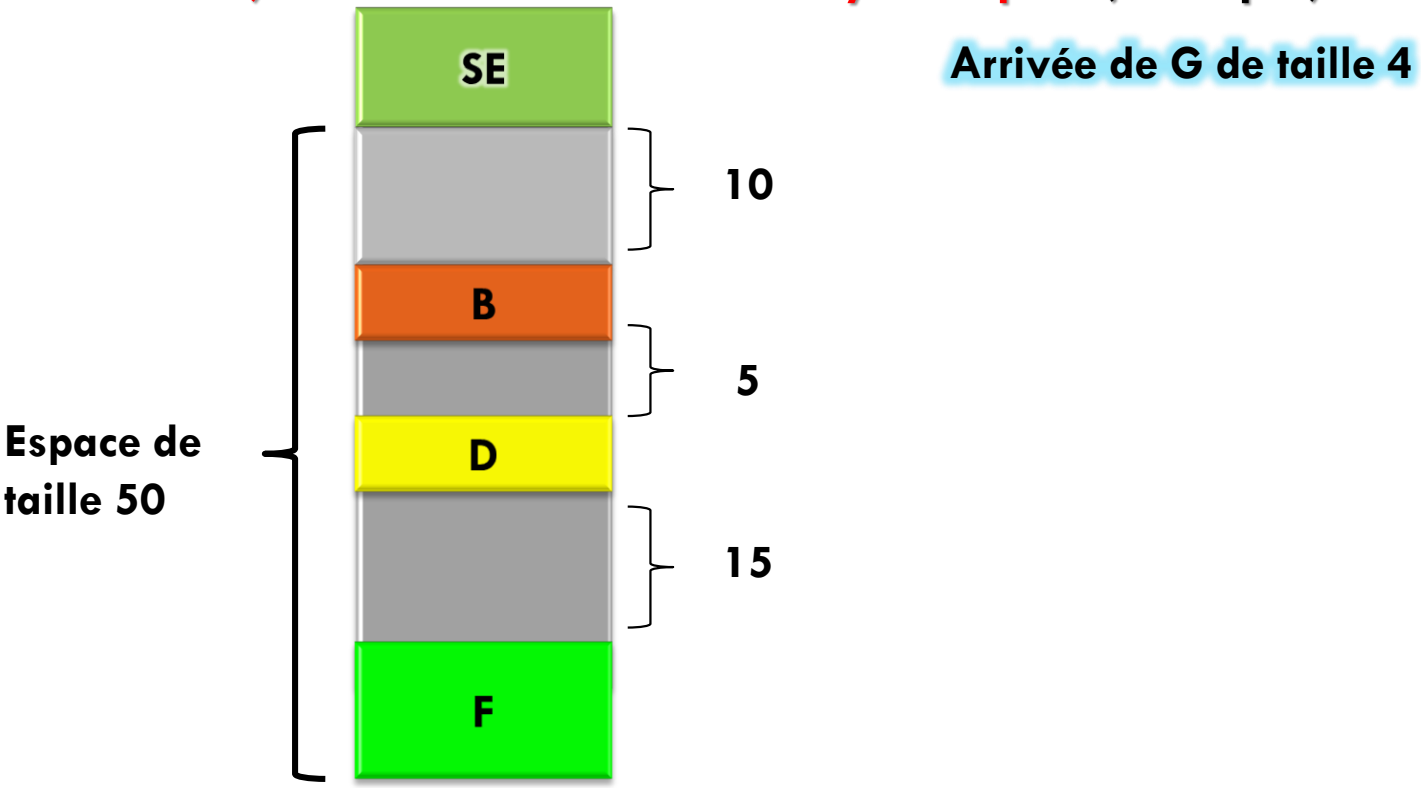
b) Partions variables ou dynamiques (exemple)



Gestion de la mémoire uniforme

2. Cas de la multiprogrammation

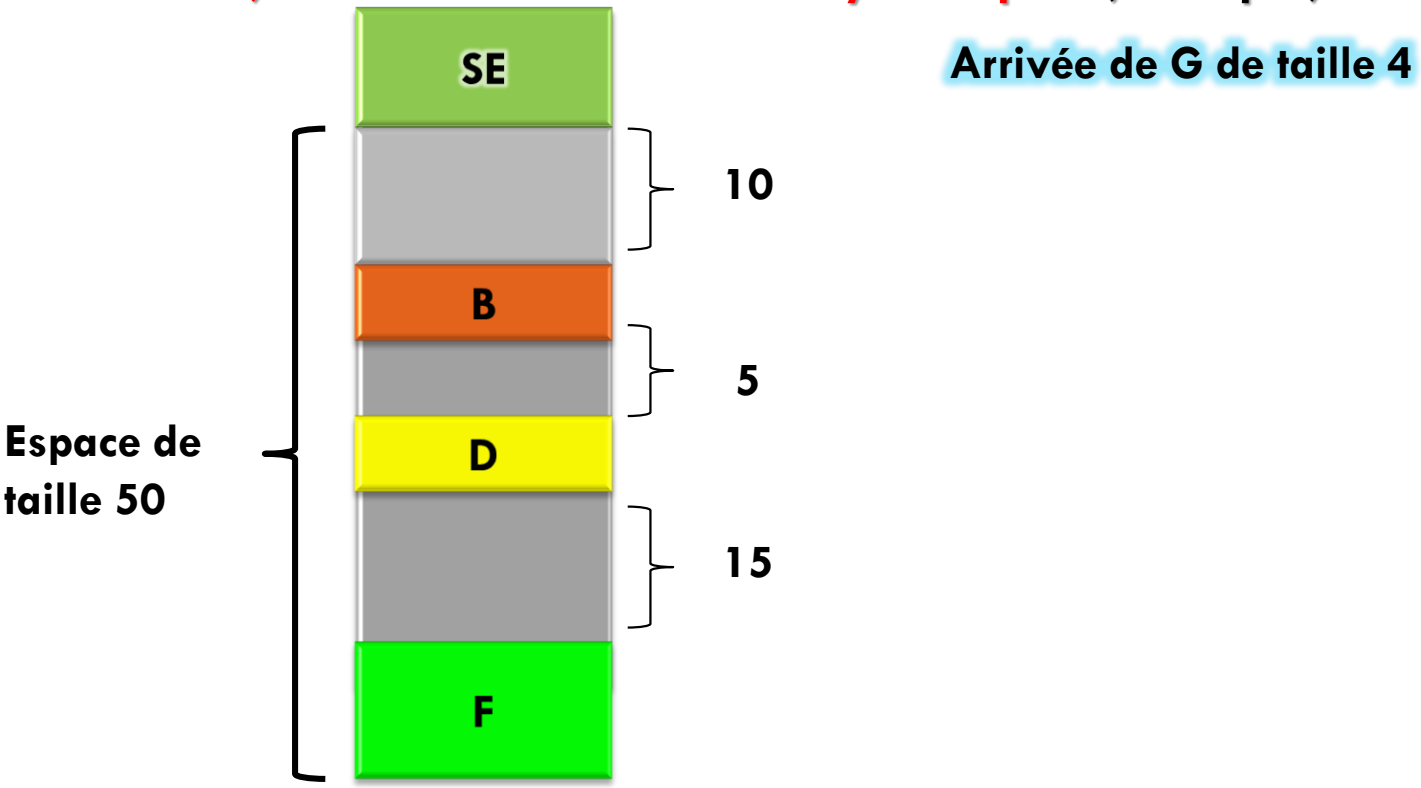
b) Partions variables ou dynamiques (exemple)



Gestion de la mémoire uniforme

2. Cas de la multiprogrammation

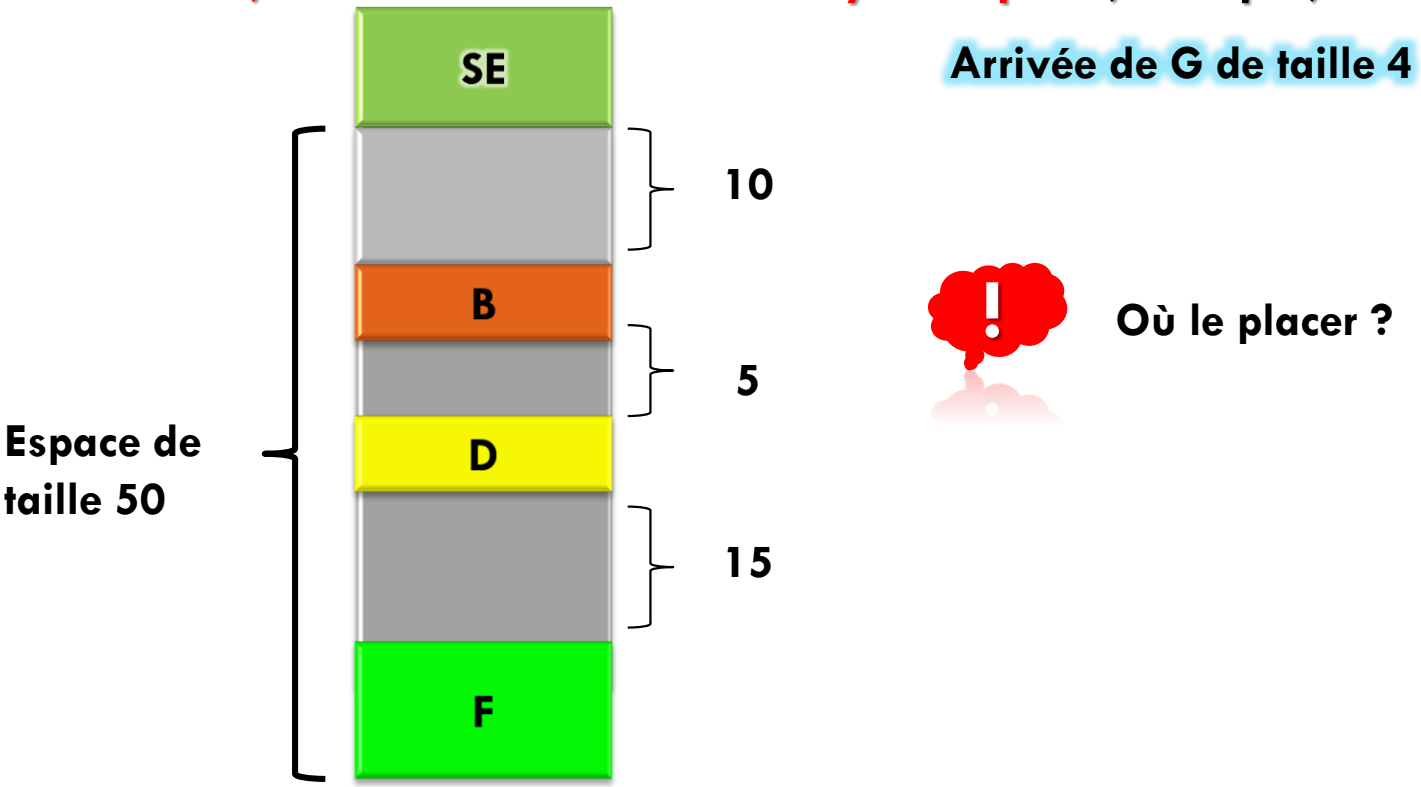
b) Partions variables ou dynamiques (exemple)



Gestion de la mémoire uniforme

2. Cas de la multiprogrammation

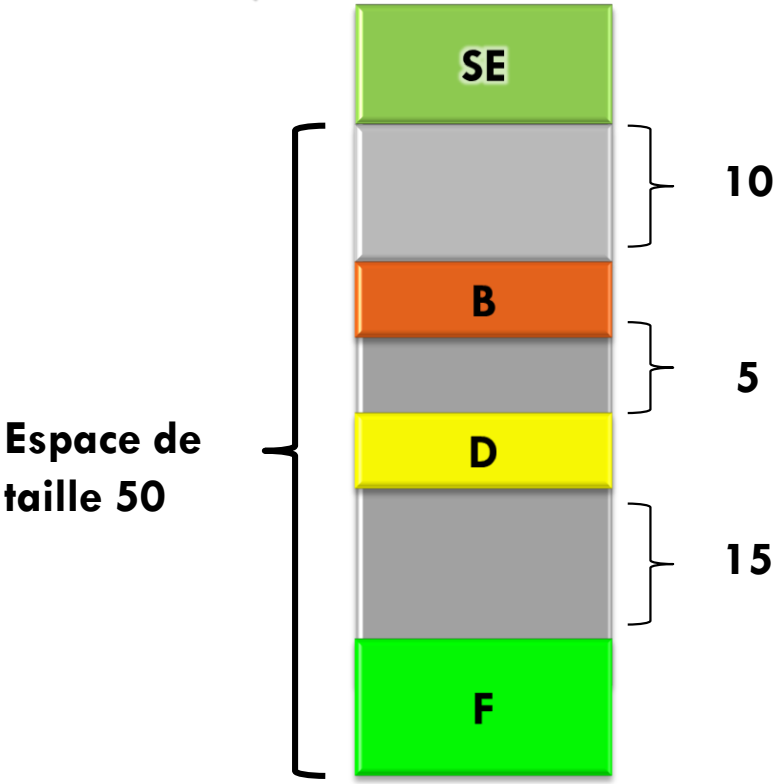
b) Partions variables ou dynamiques (exemple)



Gestion de la mémoire uniforme

2. Cas de la multiprogrammation

b) Partions variables ou dynamiques (exemple)



Arrivée de G de taille 4

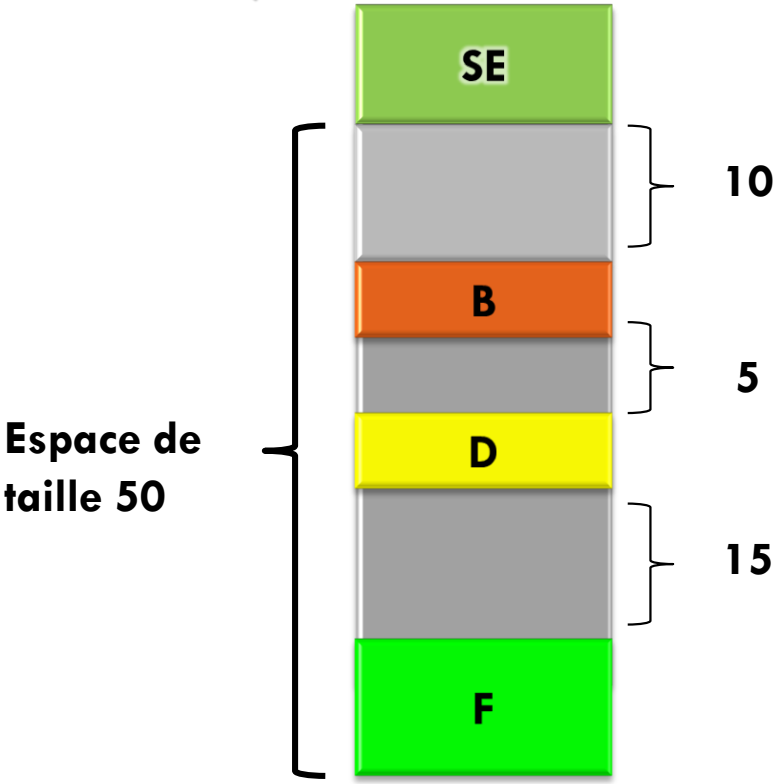
Algorithmes de placement :

- First Fit
- Best fit
- Worst Fit

Gestion de la mémoire uniforme

2. Cas de la multiprogrammation

b) Partions variables ou dynamiques (exemple)



Arrivée de G de taille 4

Algorithmes de placement :

- First Fit

Premier trou possible

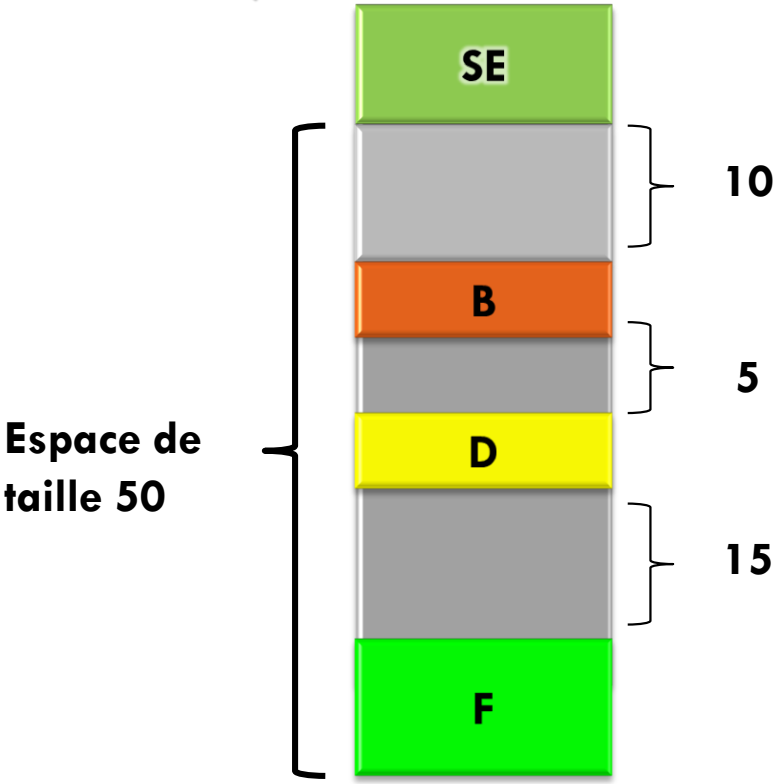
\+ Rapide

\- Risque d'augmenter la frag. externe

Gestion de la mémoire uniforme

2. Cas de la multiprogrammation

b) Partions variables ou dynamiques (exemple)



Arrivée de G de taille 4

Algorithmes de placement :

❖Best Fit

Plus petit trou possible

\+ Diminue la valeur de frag. externe

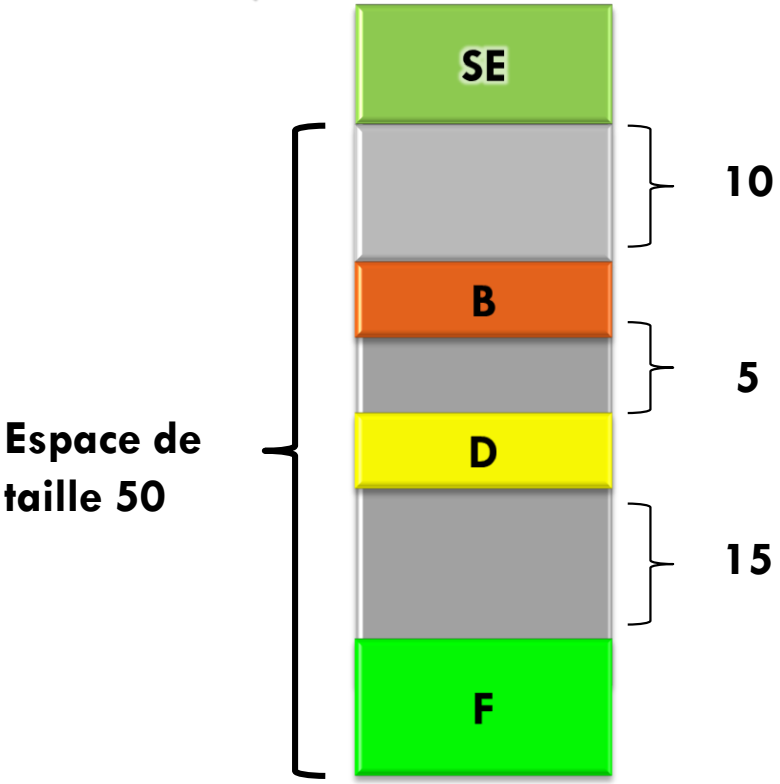
\- Lent

\- Les fragments sont de petites tailles et incapables de recevoir d'autres processus

Gestion de la mémoire uniforme

2. Cas de la multiprogrammation

b) Partions variables ou dynamiques (exemple)



Arrivée de G de taille 4

Algorithmes de placement :

❖Worst Fit

Plus grand trou possible

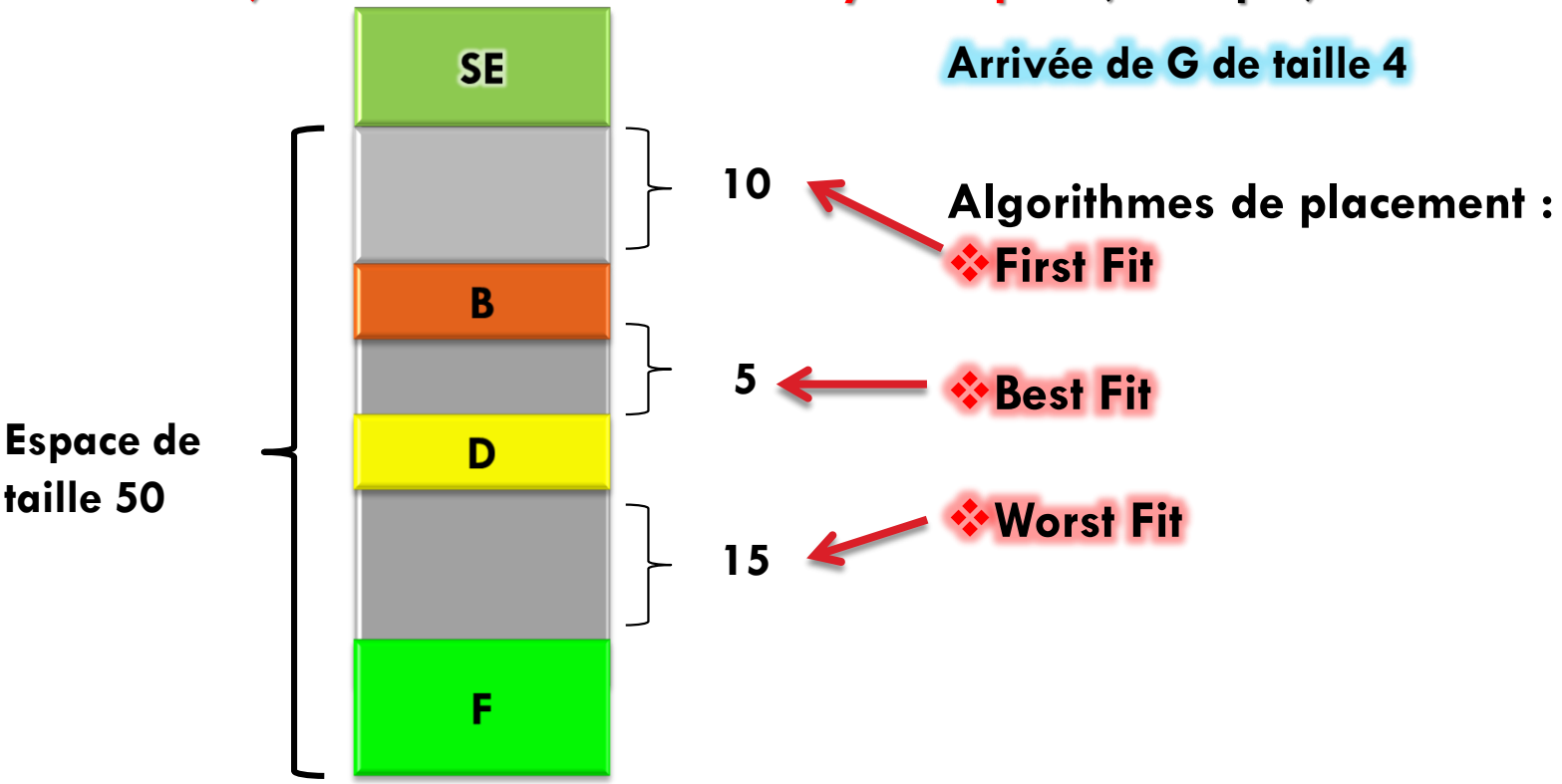
+ Les fragments (résidus) peuvent recevoir d'autres processus

- Lent
- Risque de famine pour les processus de grandes tailles

Gestion de la mémoire uniforme

2. Cas de la multiprogrammation

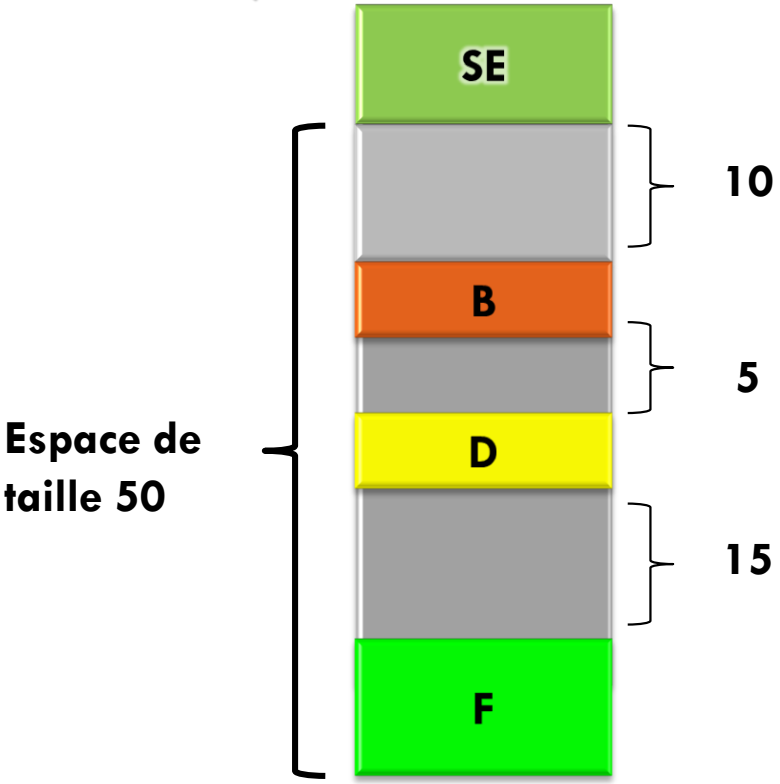
b) Partions variables ou dynamiques (exemple)



Gestion de la mémoire uniforme

2. Cas de la multiprogrammation

b) Partions variables ou dynamiques (exemple)



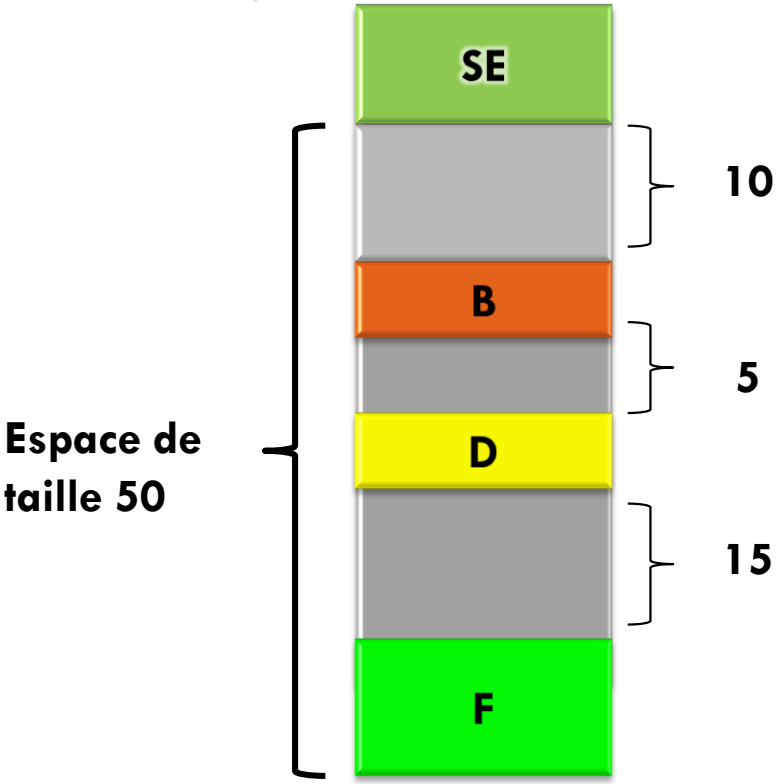
Si G est de taille 25

Où le placer ?

Gestion de la mémoire uniforme

2. Cas de la multiprogrammation

b) Partions variables ou dynamiques (exemple)



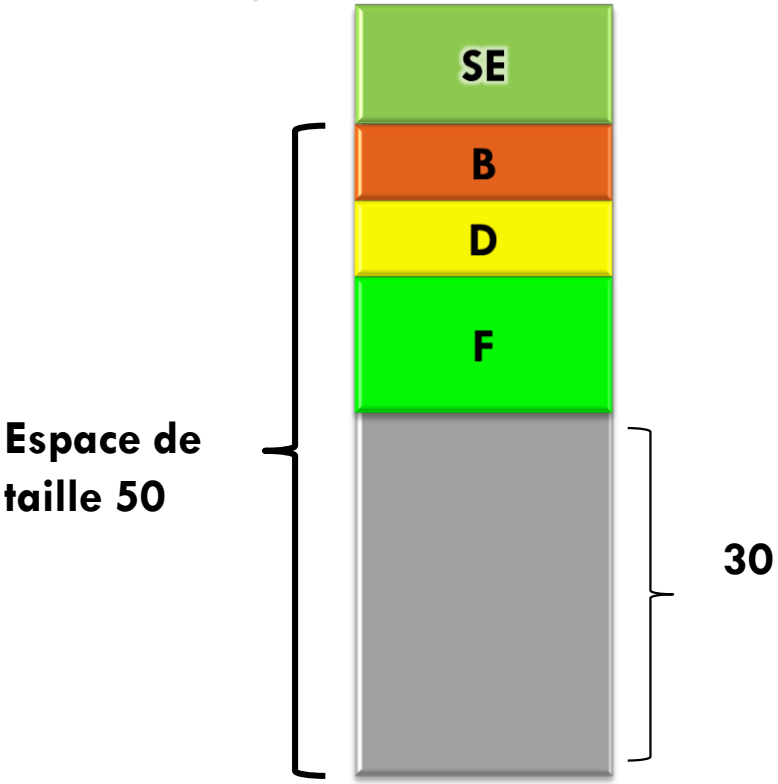
Si G est de taille 25

Solution :
Compactage = défragmentation

Gestion de la mémoire uniforme

2. Cas de la multiprogrammation

b) Partions variables ou dynamiques (exemple)



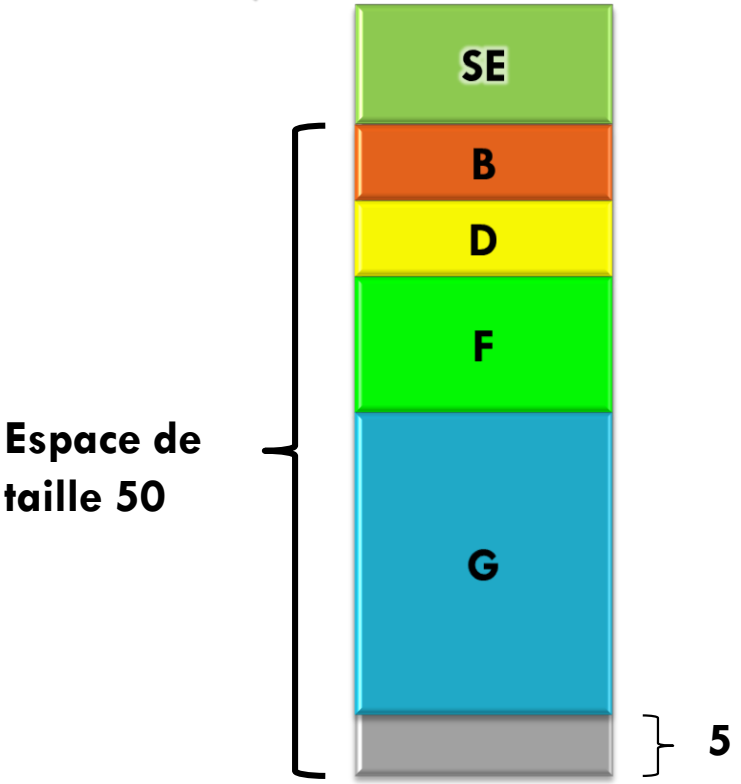
Si G est de taille 25

Solution :
Compactage = défragmentation

Gestion de la mémoire uniforme

2. Cas de la multiprogrammation

b) Partions variables ou dynamiques (exemple)



Si G est de taille 25

Solution :

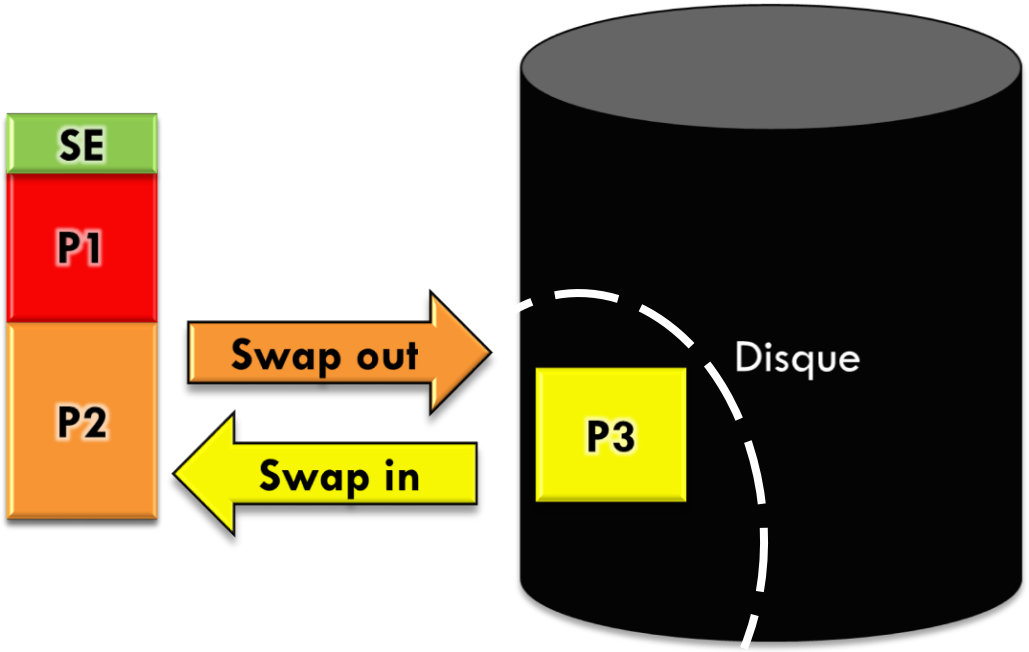
Compactage = défragmentation

Placement

Gestion de la mémoire uniforme

3. Swap (technique de va-et-vient)

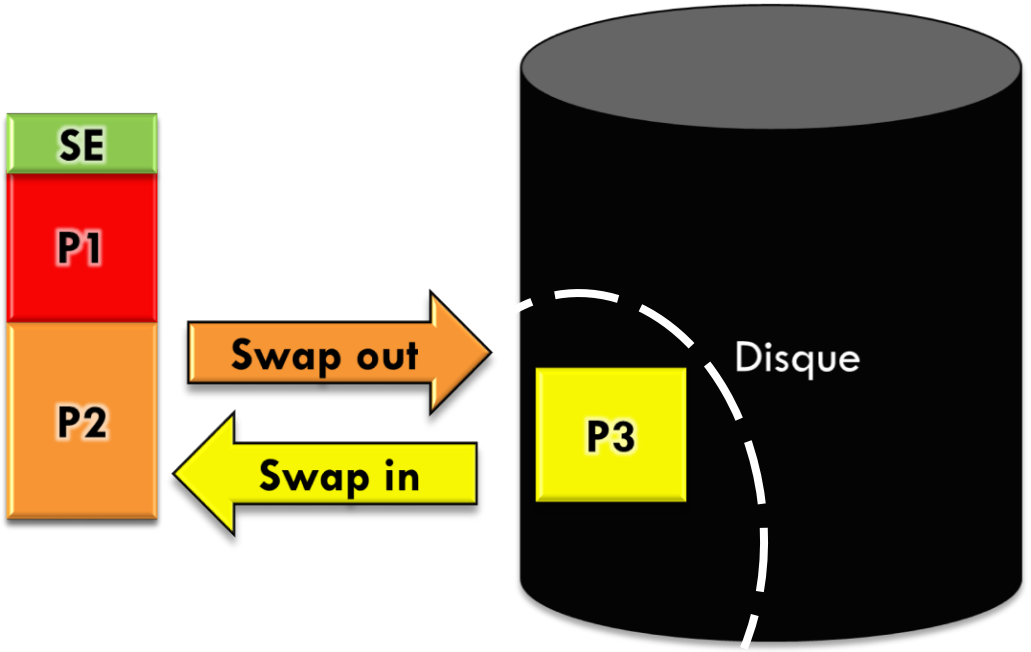
Quand la RAM devient insuffisante pour recevoir tous les processus, certains sont placés dans une zone du disque dur dite zone de swap. Ces processus seront ramenés en RAM à la demande



Gestion de la mémoire uniforme

3. Swap (technique de va-et-vient)

Cette technique est consommatrice du temps (nécessité des mise à jour des adresses de base des processus concernés par le swap)



Gestion de la mémoire uniforme

4. Évaluation de la mémoire uniforme

- **Il faut prendre en considération la limite imposée par la taille de la RAM**

Gestion de la mémoire uniforme

4. Évaluation de la mémoire uniforme

- **Il faut prendre en considération la limite imposée par la taille de la RAM**

  a) **Si la taille de processus dépasse celle de la RAM :**

  - **Overlays (à la charge du programmeur)**

Gestion de la mémoire uniforme

4. Évaluation de la mémoire uniforme

- **Il faut prendre en considération la limite imposée par la taille de la RAM**

  a) **Si la taille de processus dépasse celle de la RAM :**

     - **Overlays (à la charge du programmeur)**

  b) **Si le nombre de processus augmente :**

     - **Swap du processus entier (très coûteux)**

Gestion de la mémoire uniforme

4. Évaluation de la mémoire uniforme

- **Partitions fixes ou statiques : Placement via**
  - **Une file d'attente par partition**
  - **Une seule file d'attente pour toutes les partitions**
    - **Dès qu'une partition est libre on lui affecte le 1^{ier} processus possible**
    - **Dès qu'une partition est libre on lui affecte le plus grand processus possible**
- **Partitions variables ou dynamiques : Placement selon**
  - **First fit**
  - **Best fit**
  - **Worst fit**

Gestion de la mémoire uniforme

4. Évaluation de la mémoire uniforme

Critère	Partitions fixes	Partitions variables
Nombre de partitions connu	✓	✗
Tailles de partitions définies	✓	✗
Localisations de partitions définies	✓	✗
Risque de déséquilibre entre les partitions	✓	✗
Fragmentation interne	✓	✗
Fragmentation externe	✗	✓

Gestion de la mémoire virtuelle

1. Objectifs

- **Surmonter la limite imposée par la taille de la RAM**

a) **Si la taille de processus dépasse ou non celle de la RAM :**

- **Découpage automatique de processus par le SE en des segments ou des pages**

b) **Si le nombre de processus augmente :**

- **Swap de centaines parties des processus (les parties utilisées sont conservées en RAM et le reste est stocké sur le disque si nécessaire)**

Gestion de la mémoire virtuelle

2. Segmentation

Chaque processus est divisé en des modules ou segments

Chaque segment :

- **correspond à une entité logique (fonction, données, …)**
- **possède un numéro, une taille et une adresse de base s'il est chargé en RAM**

Gestion de la mémoire virtuelle

2. Segmentation

Chaque processus est divisé en des modules ou segments

Chaque segment :

- **correspond à une entité logique (fonction, données, …)**
- **possède un numéro, une taille et une adresse de base s'il est chargé en RAM**

L'adresse logique dans ce cas est :

(numéro de segment, déplacement à faire dans ce segment)

Gestion de la mémoire virtuelle

2. Segmentation

Chaque processus possède une table de segments (conservée dans le PCB du dit processus).

Gestion de la mémoire virtuelle

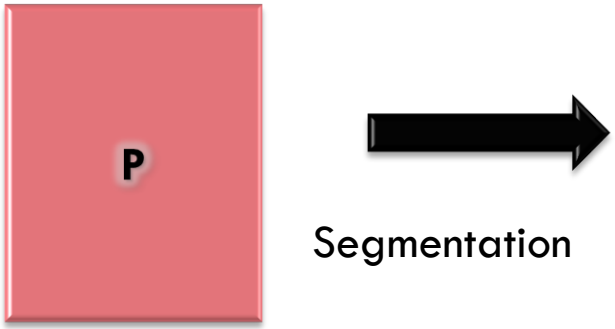
2. Segmentation

Chaque processus possède une table de segments (conservée dans le PCB du dit processus).

Cette table contient toutes les informations relatives aux segments du processus (elle indique pour chaque segment : sa taille et son adresse de base).

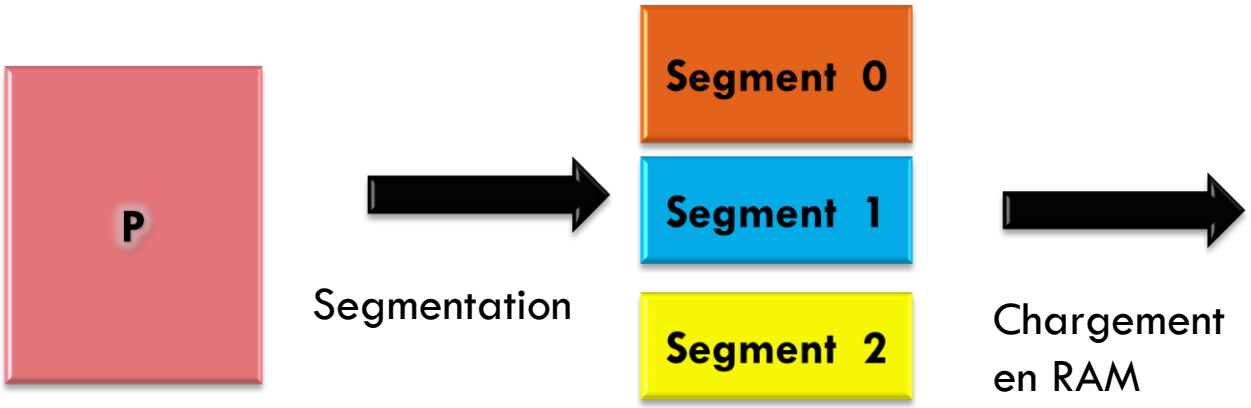
Gestion de la mémoire virtuelle

2. Segmentation



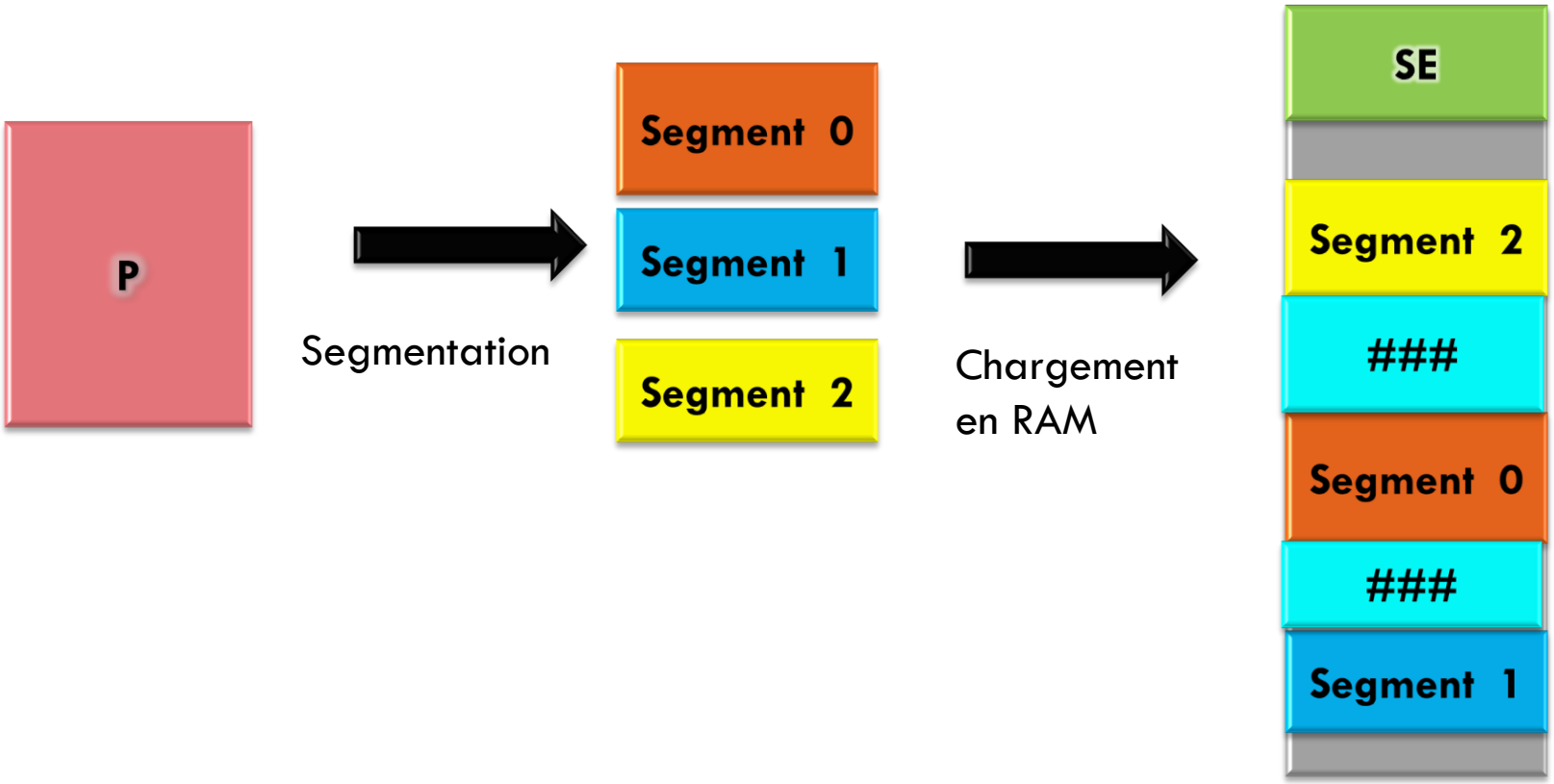
Gestion de la mémoire virtuelle

2. Segmentation



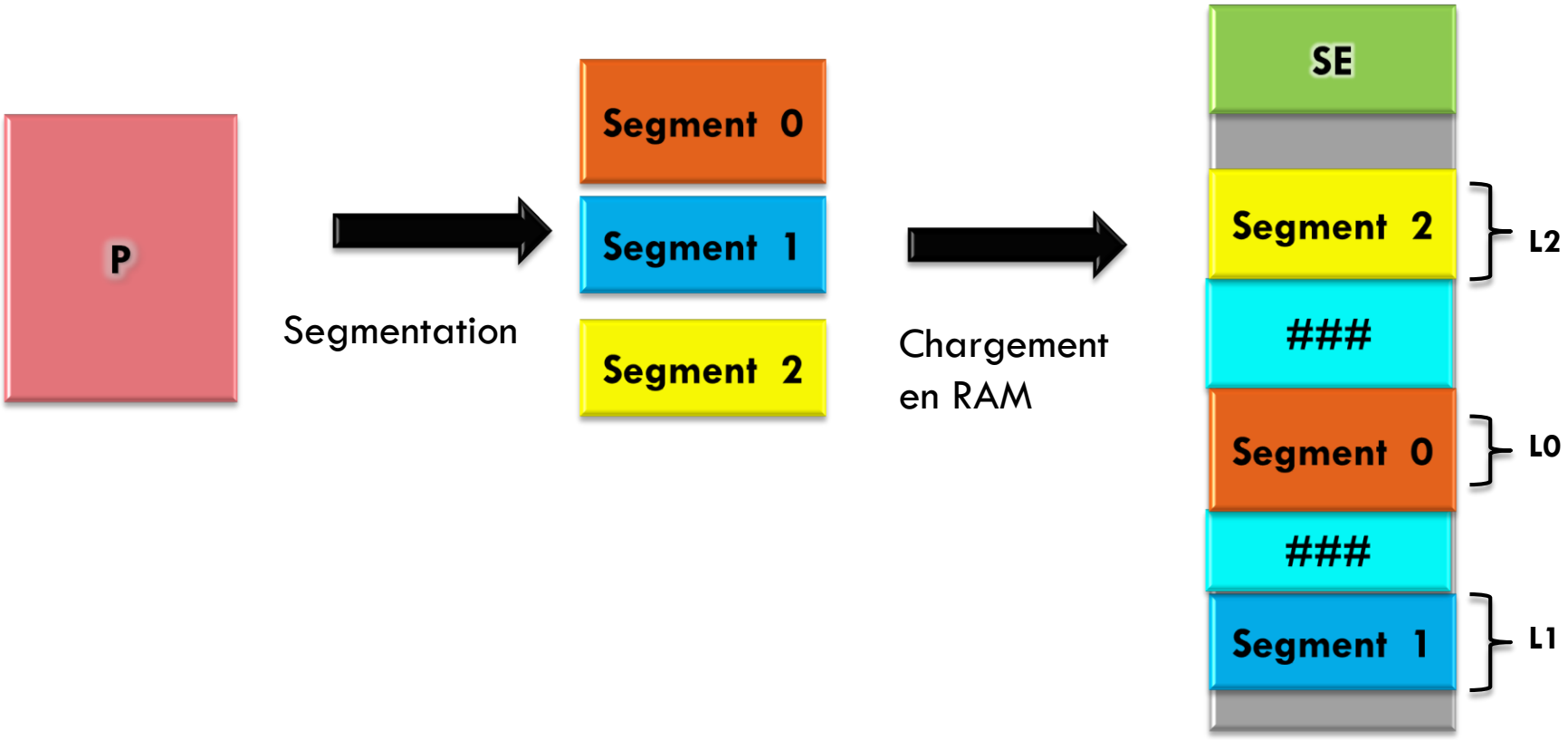
Gestion de la mémoire virtuelle

2. Segmentation



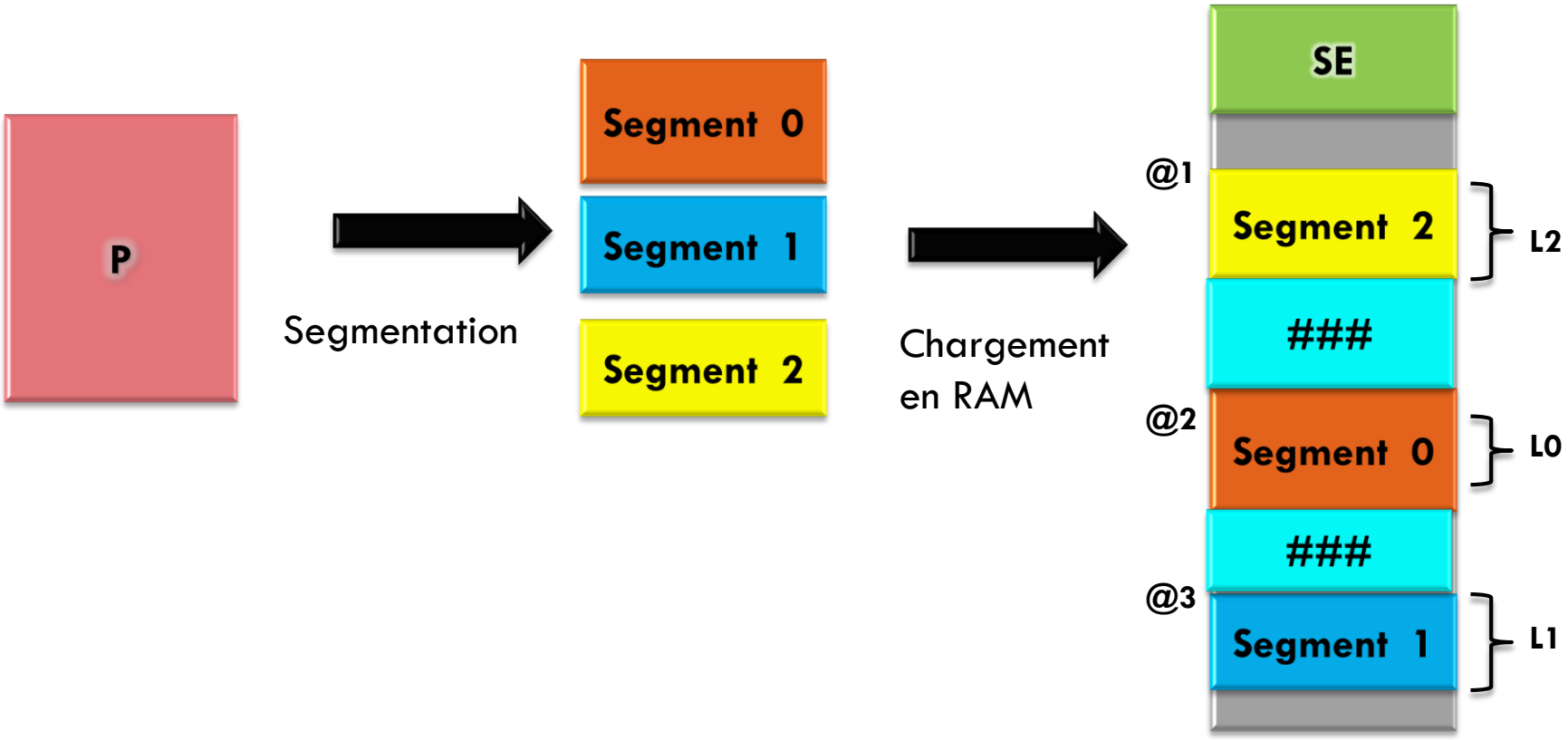
Gestion de la mémoire virtuelle

2. Segmentation



Gestion de la mémoire virtuelle

2. Segmentation



Gestion de la mémoire virtuelle

2. Segmentation

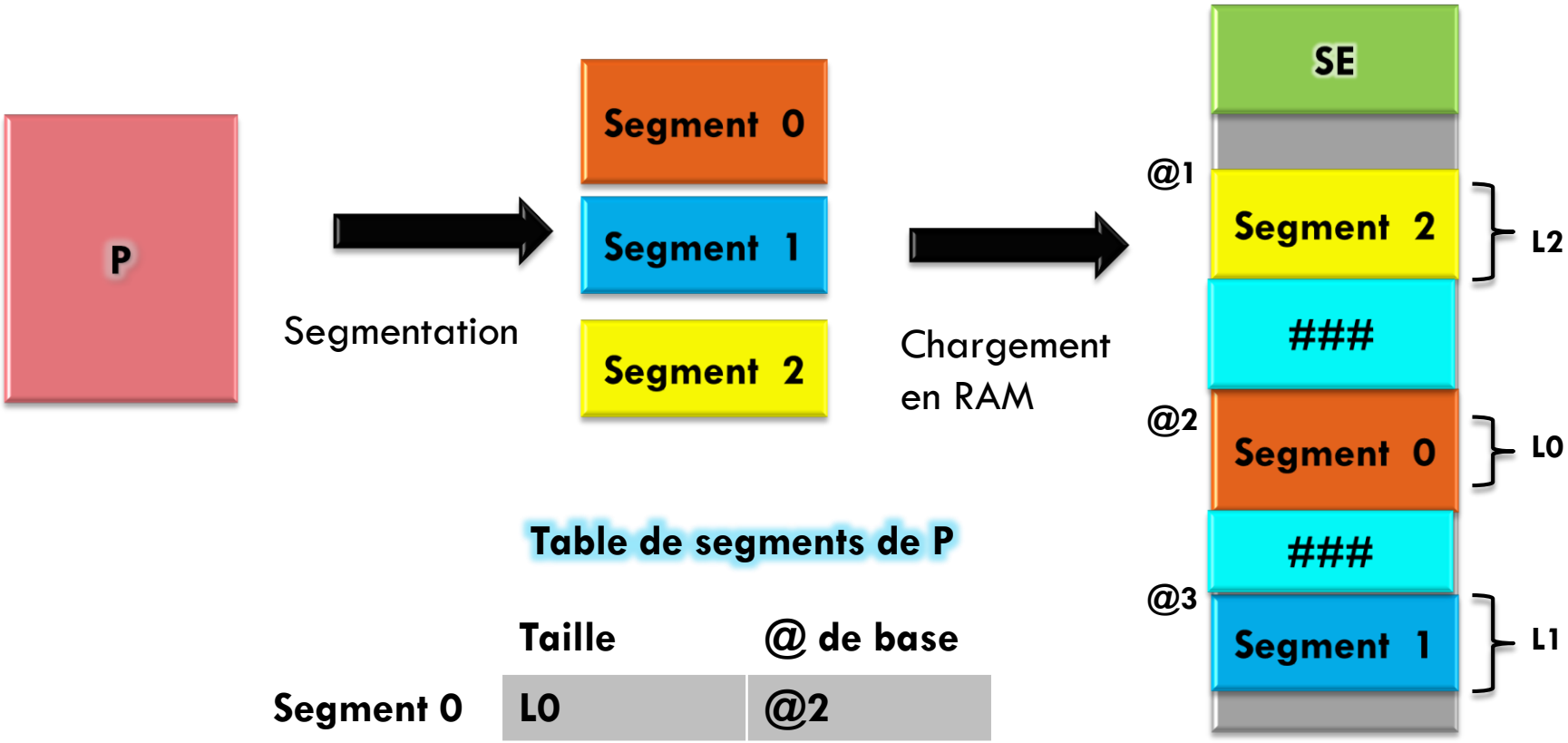
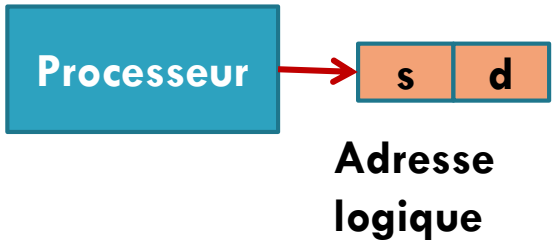


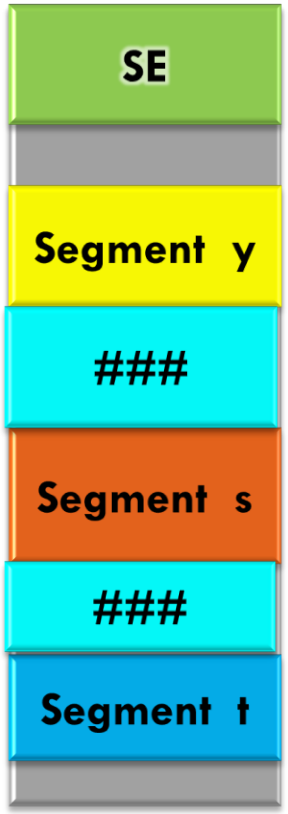
Table de segments de P

	Taille	@ de base
Segment 0	L0	@2
Segment 1	L1	@3
Segment 2	L2	@1

Gestion de la mémoire virtuelle

2. Segmentation

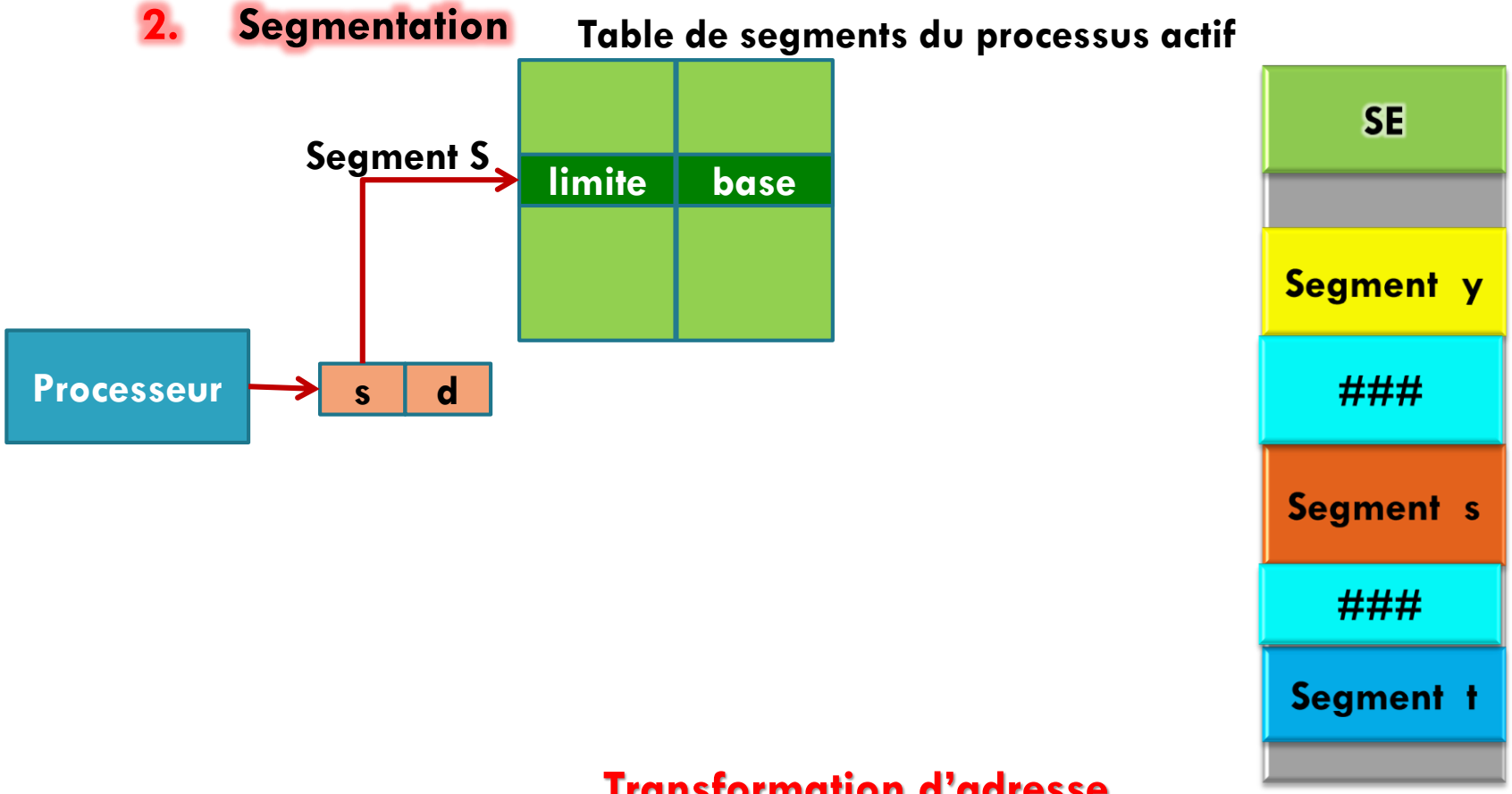




Transformation d'adresse

Gestion de la mémoire virtuelle

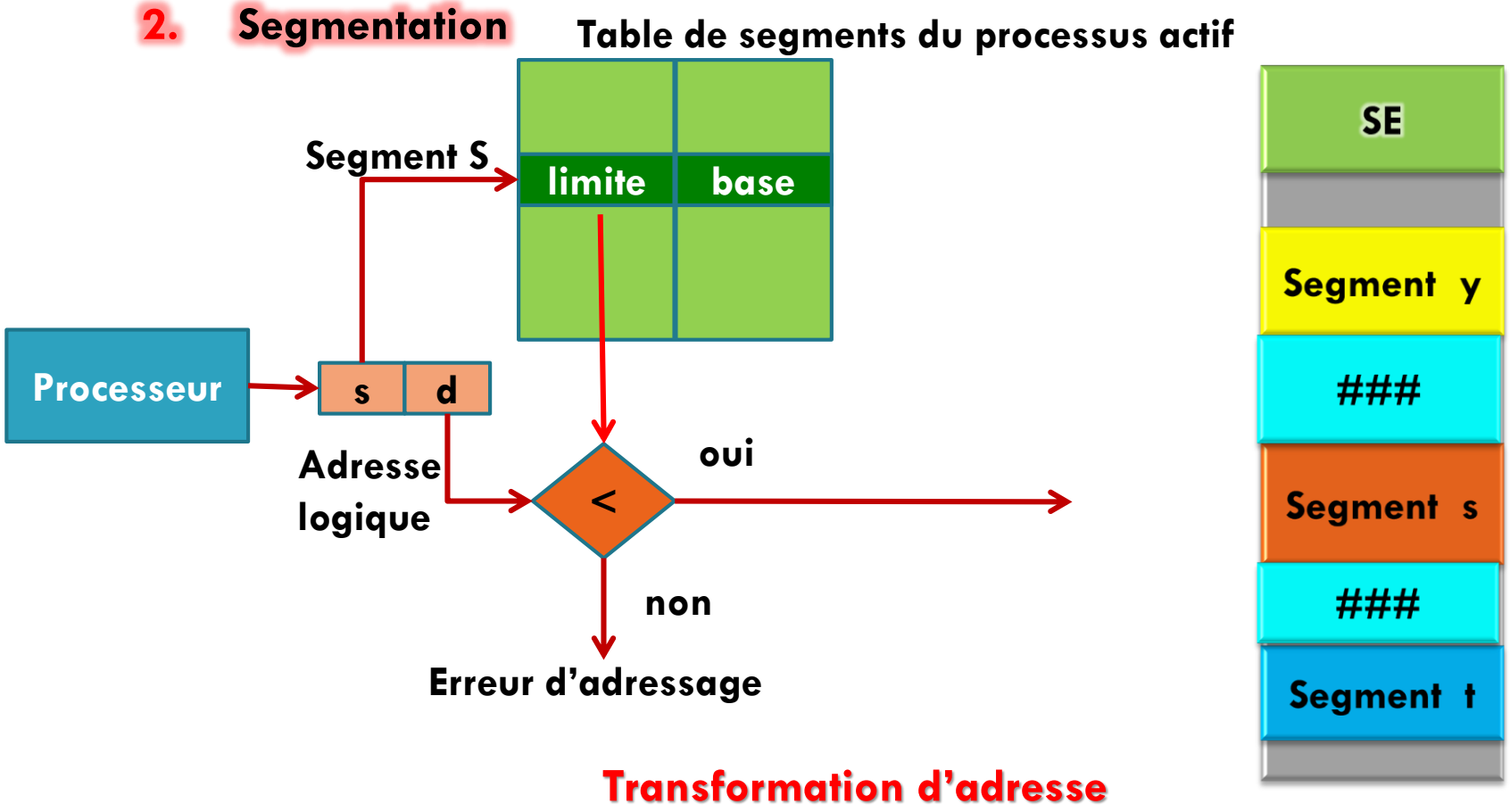
2. Segmentation



Transformation d'adresse

Gestion de la mémoire virtuelle

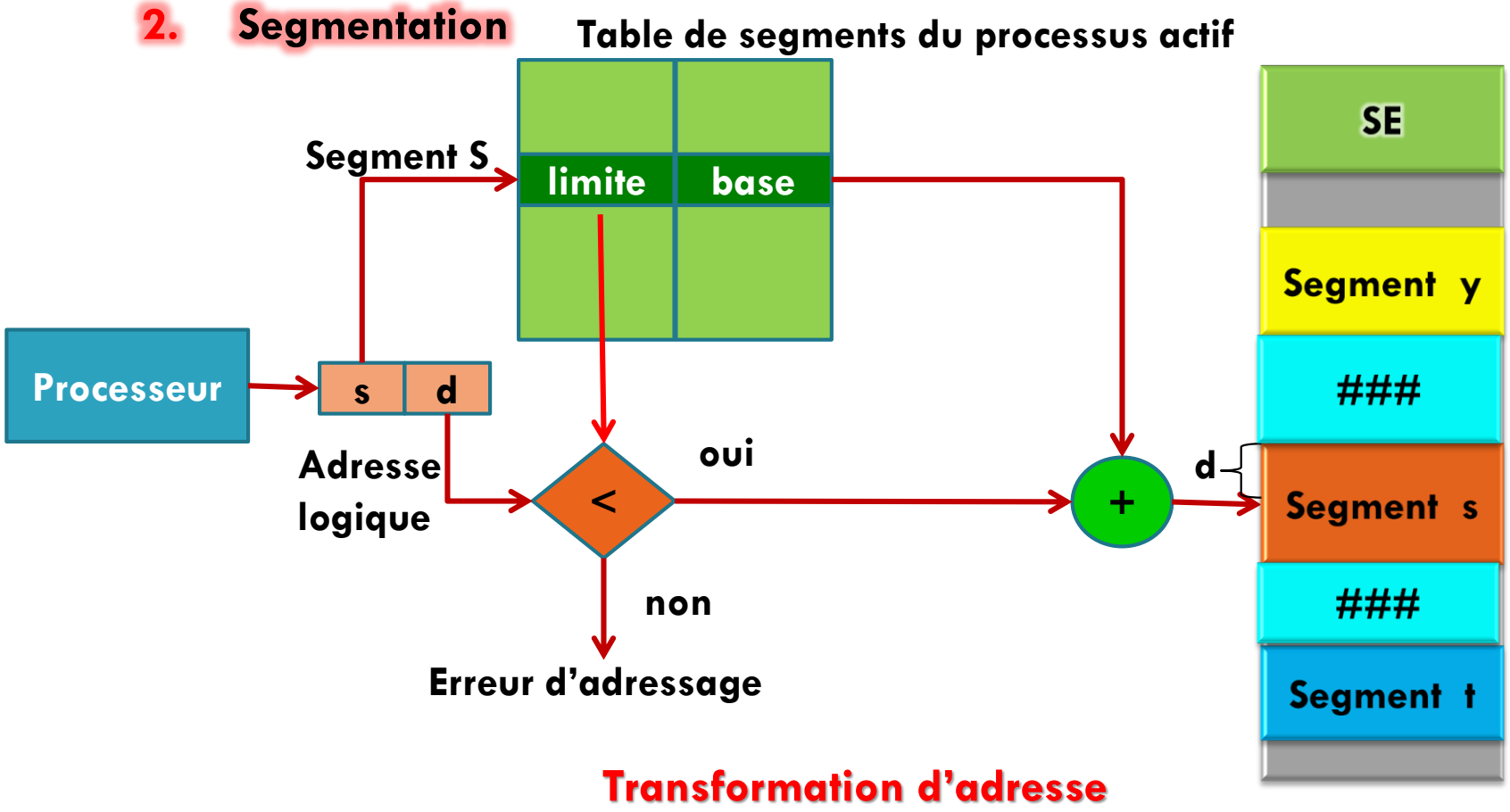
2. Segmentation



Transformation d'adresse

Gestion de la mémoire virtuelle

2. Segmentation



Transformation d'adresse

Gestion de la mémoire virtuelle

2. Segmentation

Le placement des segments se base sur le First Fit ou le Best Fit.

Si le segment demandé n'existe pas en RAM alors c'est un défaut de segment.

Gestion de la mémoire virtuelle

3. Pagination

- La pagination consiste à diviser le processus en des parties de même taille dites **pages**
- La RAM est également partitionnée en des zones de même taille dites **cadres** ou **cases** ou **frames**
- La taille d'une page est égale la taille d'un cadre (puissance de 2)
- Les différentes pages sont conservées sur le disque et chargées dynamiquement à la demande en RAM

Gestion de la mémoire virtuelle

3. Pagination

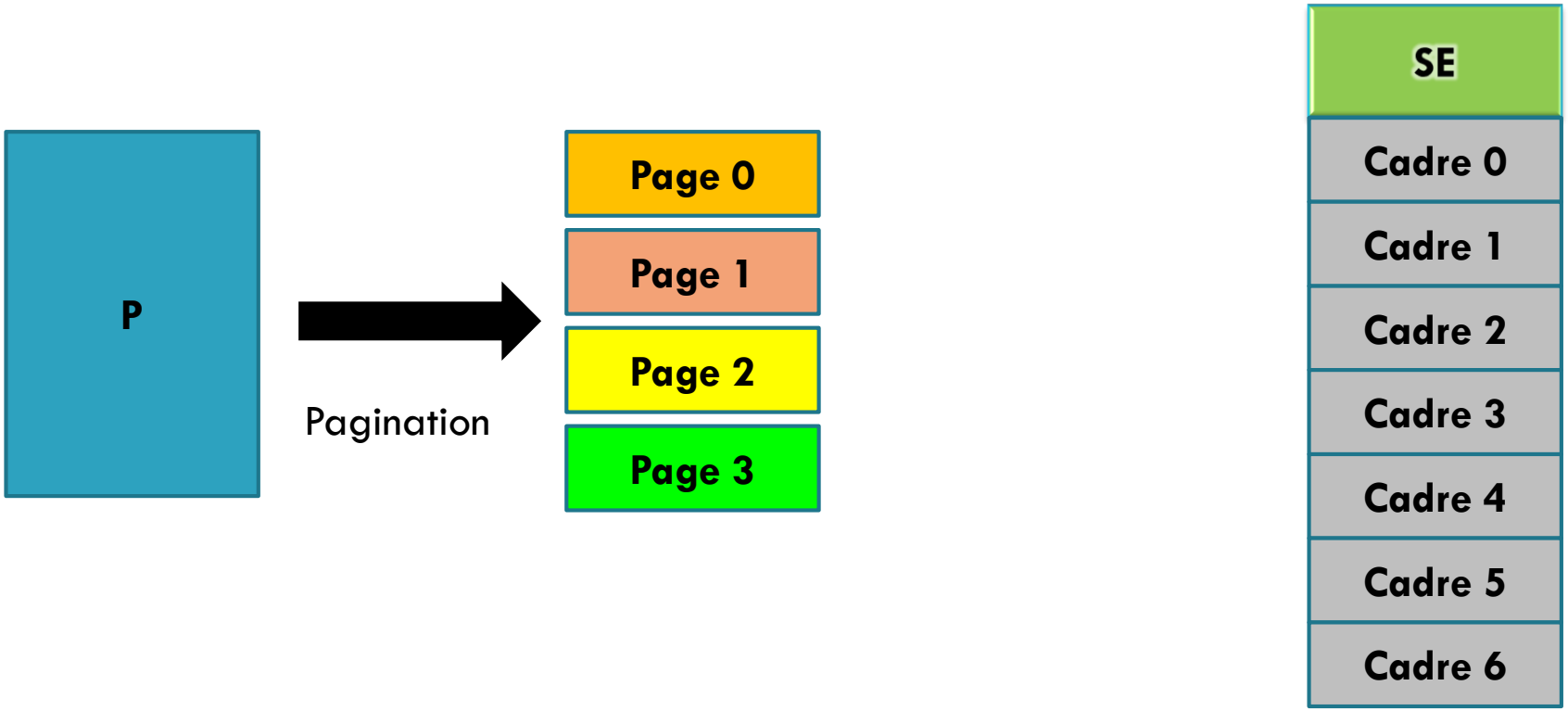
L'adresse logique dans ce cas est :

(Numéro de page, Déplacement à faire dans cette page)

Chaque processus admet une table de pages qui indique pour chaque page le numéro de cadre où est chargée la page.

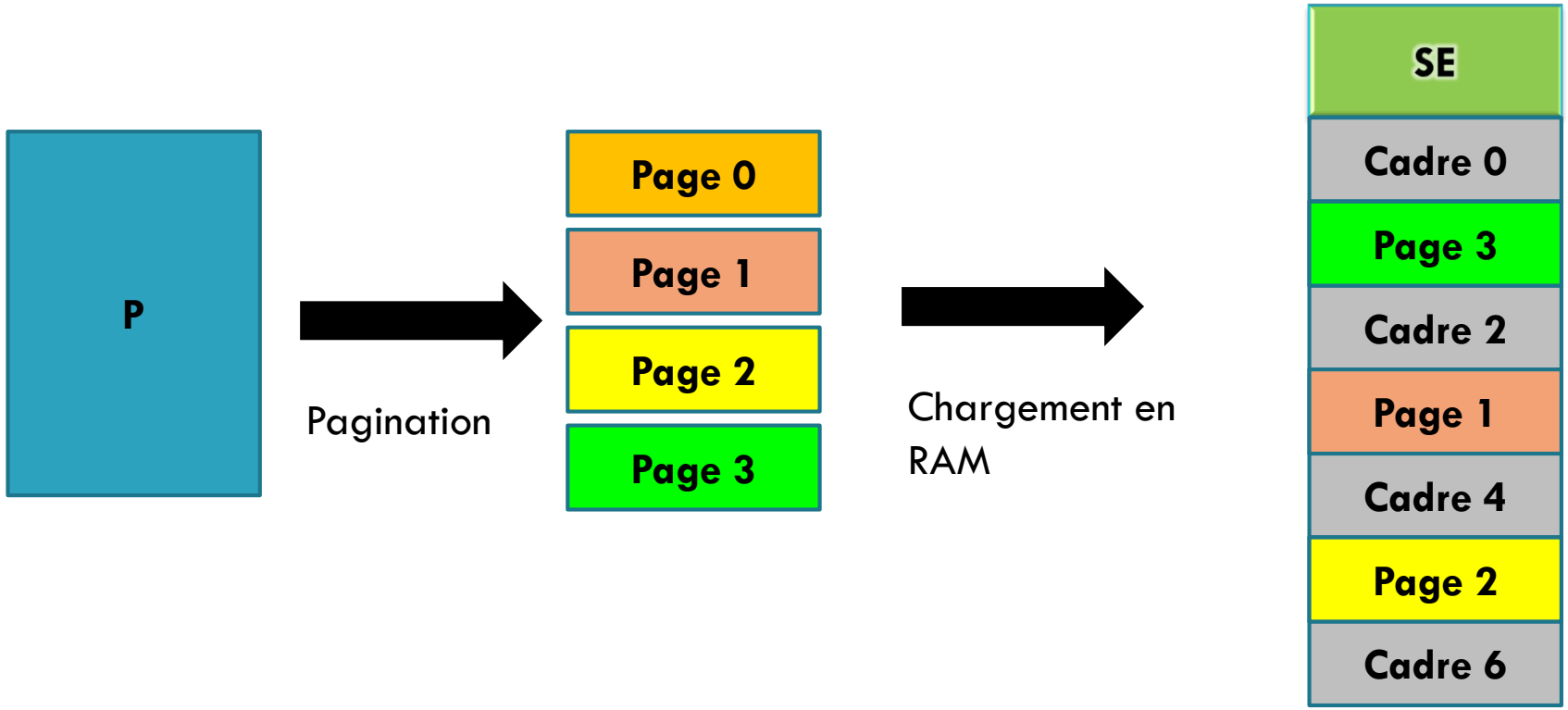
Gestion de la mémoire virtuelle

3. Pagination



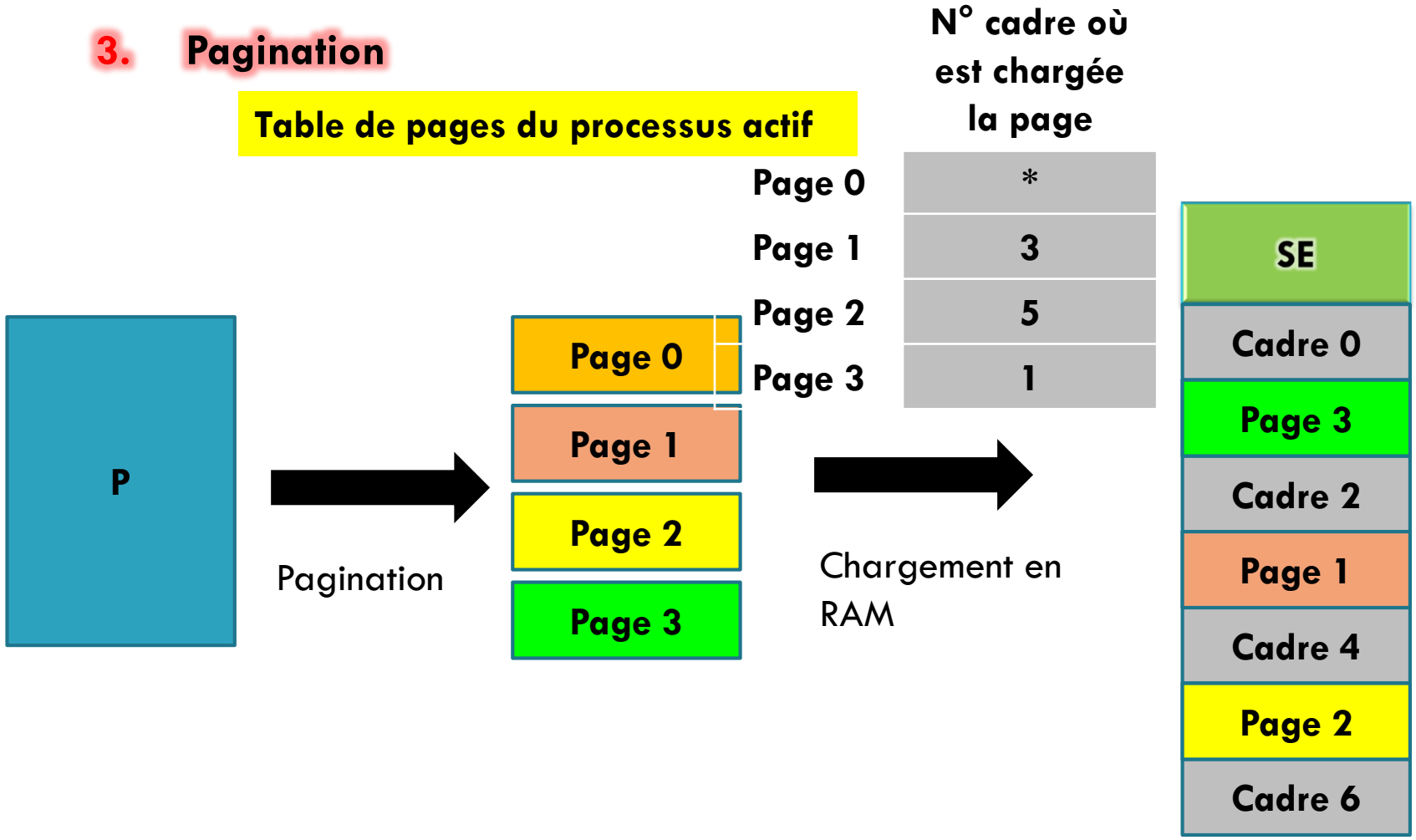
Gestion de la mémoire virtuelle

3. Pagination



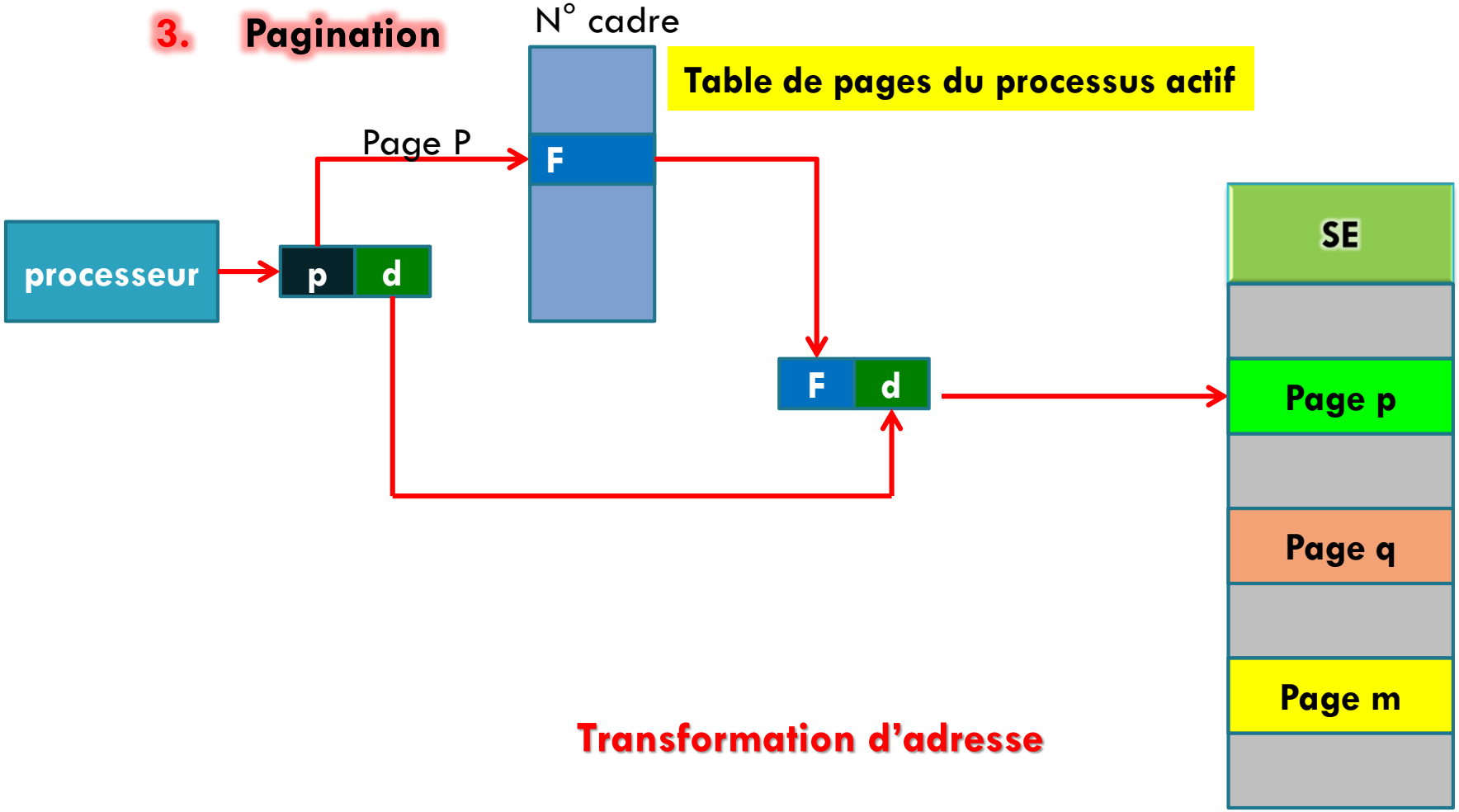
Gestion de la mémoire virtuelle

3. Pagination



Gestion de la mémoire virtuelle

3. Pagination



Gestion de la mémoire virtuelle

3. Pagination

Si la page demandée n'est pas chargée en RAM alors on a un défaut de page

Gestion de la mémoire virtuelle

3. Pagination

Si la page demandée n'est pas chargée en RAM alors on a

un défaut de page

Le traitement d'un défaut de page :

- **Si la RAM n'est pas pleine alors placement selon First Fit**
- **Sinon il faut choisir une page victime qui va être remplacée par la page demandée ➡ algorithme de remplacement de page**

Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page

OPT : OPTimal

FIFO : First In First Out

LRU : Least Recently Used

NRU : Not Recently Used

Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page

Liste de références aux pages

		0	1	0	2	0	1	3	0	2	3
Cadre 0											
Cadre 1											
Cadre 2											

D.P

Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page

Etat initial de la mémoire centrale

		0	1	0	2	0	1	3	0	2	3
Cadre 0											
Cadre 1											
Cadre 2											

D.P

Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page

		0	1	0	2	0	1	3	0	2	3
Cadre 0											
Cadre 1											
Cadre 2											
D.P											

Défaut de pages : présent (x) ou non (-)

Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page

		0	1	0	2	0	1	3	0	2	3
Cadre 0											
Cadre 1											
Cadre 2											
D.P		x									

- Défaut de page
- Mémoire centrale non pleine

Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page

		0	1	0	2	0	1	3	0	2	3
Cadre 0		0									
Cadre 1											
Cadre 2											
D.P		**x**									

- Défaut de page
- Mémoire centrale non pleine

Placement selon First Fit

Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page

		0	1	0	2	0	1	3	0	2	3
Cadre 0		0	0								
Cadre 1			1								
Cadre 2											
D.P		x	x								

- Défaut de page
- Mémoire centrale non pleine

Placement selon First Fit

Gestion de la mémoire virtuelle

3. Pagination

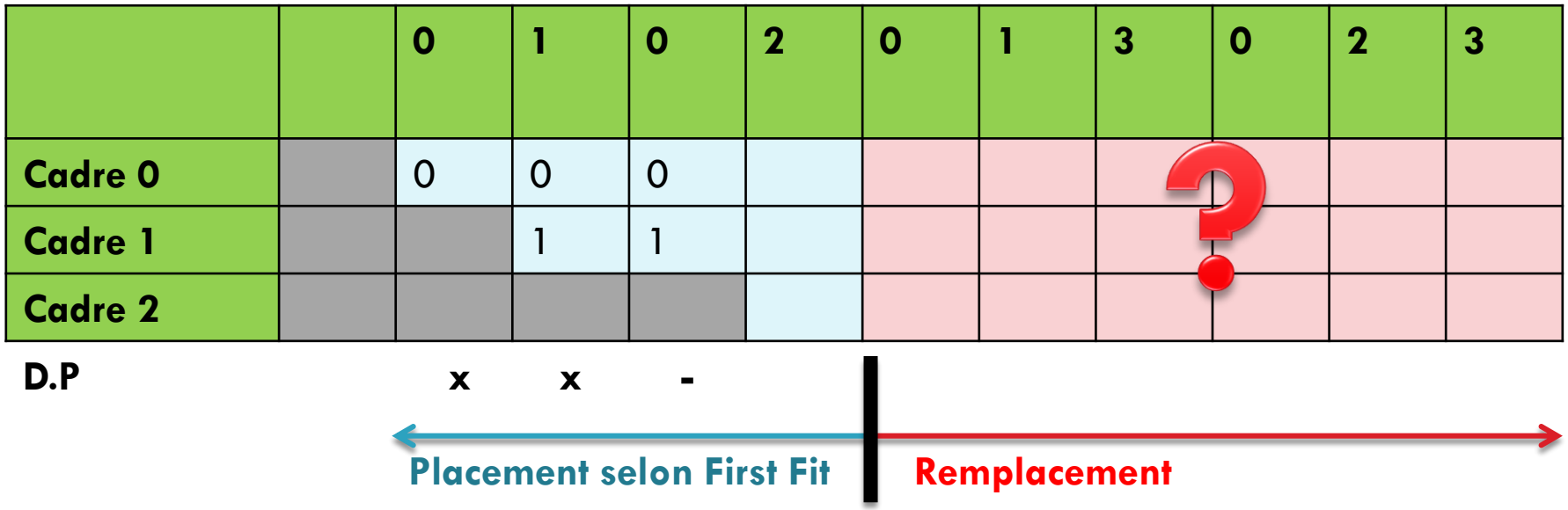
Algorithmes de remplacement de page

		0	1	0	2	0	1	3	0	2	3
Cadre 0		0	0	0							
Cadre 1			1	1							
Cadre 2											
D.P			x	x	-						

Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page



Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page

OPT (OPTimal)

Principe

- Chaque page est étiquetée par le nombre d'instructions qui seront exécutées avant que la page ne soit référencée
- La page victime est celle dont la valeur de l'étiquette est la plus grande: c'est la page qui sera référencée le plus tard possible

Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page

OPT (OPTimal)

		0	1	0	2	0	1	3	0	2	3
Cadre 0											
Cadre 1											
Cadre 2											

D.P

Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page

OPT (OPTimal)

		0	1	0	2	0	1	3	0	2	3
Cadre 0		0									
Cadre 1											
Cadre 2											
D.P		x									

Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page

OPT (OPTimal)

		0	1	0	2	0	1	3	0	2	3
Cadre 0		0	0								
Cadre 1			1								
Cadre 2											
D.P		x	x								

Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page

OPT (OPTimal)

		0	1	0	2	0	1	3	0	2	3
Cadre 0		0	0	0							
Cadre 1			1	1							
Cadre 2											
D.P		x	x	-							

Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page

OPT (OPTimal)

		0	1	0	2	0	1	3	0	2	3
Cadre 0		0	0	0	0						
Cadre 1			1	1	1						
Cadre 2					2						
D.P		x	x	-	x						

Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page

OPT (OPTimal)

		0	1	0	2	0	1	3	0	2	3
Cadre 0		0	0	0	0	0					
Cadre 1			1	1	1	1					
Cadre 2					2	2					
D.P		x	x	-	x	-					

Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page

OPT (OPTimal)

		0	1	0	2	0	1	3	0	2	3
Cadre 0		0	0	0	0	0	0				
Cadre 1			1	1	1	1	1				
Cadre 2					2	2	2				
D.P		x	x	-	x	-	-				

Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page

OPT (OPTimal)

		0	1	0	2	0	1	3	0	2	3
Cadre 0		0	0	0	0	0	0_1				
Cadre 1			1	1	1	1	1				
Cadre 2					2	2	2				
D.P		x	x	-	x	-	-				

Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page

OPT (OPTimal)

		0	1	0	2	0	1	3	0	2	3
Cadre 0		0	0	0	0	0	0_1				
Cadre 1			1	1	1	1	1_∞				
Cadre 2					2	2	2				
D.P		x	x	-	x	-	-				

Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page

OPT (OPTimal)

		0	1	0	2	0	1	3	0	2	3
Cadre 0		0	0	0	0	0	0_1				
Cadre 1			1	1	1	1	1_∞				
Cadre 2					2	2	2_2				
D.P		x	x	-	x	-	-				

Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page

OPT (OPTimal)

		0	1	0	2	0	1	3	0	2	3
Cadre 0		0	0	0	0	0	0	0			
Cadre 1			1	1	1	1	1	3			
Cadre 2					2	2	2	2			
D.P		x	x	-	x	-	-	x			

Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page

OPT (OPTimal)

		0	1	0	2	0	1	3	0	2	3
Cadre 0		0	0	0	0	0	0	0	0		
Cadre 1			1	1	1	1	1	3	3		
Cadre 2					2	2	2	2	2		
D.P		x	x	-	x	-	-	x	-		

Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page

OPT (OPTimal)

		0	1	0	2	0	1	3	0	2	3
Cadre 0		0	0	0	0	0	0	0	0	0	
Cadre 1			1	1	1	1	1	3	3	3	
Cadre 2					2	2	2	2	2	2	
D.P		x	x	-	x	-	-	x	-	-	

Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page

OPT (OPTimal)

		0	1	0	2	0	1	3	0	2	3
Cadre 0		0	0	0	0	0	0	0	0	0	0
Cadre 1			1	1	1	1	1	3	3	3	3
Cadre 2					2	2	2	2	2	2	2
D.P		x	x	-	x	-	-	x	-	-	-

4 D.P

Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page

FIFO (First in, First Out)

Principe

- La page victime est la plus anciennement chargée en RAM ⇒ la plus vieille parmi celles qui existent en RAM

Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page

FIFO (First in, First Out)

		0	1	0	2	0	1	3	0	2	3
Cadre 0		0	0	0	0						
Cadre 1			1	1	1						
Cadre 2					2						
D.P		x	x	-	x						

Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page

FIFO (First in, First Out)

		0	1	0	2	0	1	3	0	2	3
Cadre 0		0	0	0	0	0					
Cadre 1			1	1	1	1					
Cadre 2					2	2					
D.P		x	x	-	x	-					

Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page

FIFO (First in, First Out)

		0	1	0	2	0	1	3	0	2	3
Cadre 0		0	0	0	0	0	0				
Cadre 1			1	1	1	1	1				
Cadre 2					2	2	2				
D.P		x	x	-	x	-	-				

Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page

FIFO (First in, First Out)

		0	1	0	2	0	1	3	0	2	3
Cadre 0		0	0	0	0	0	0	3			
Cadre 1			1	1	1	1	1	1			
Cadre 2					2	2	2	2			
D.P		x	x	-	x	-	-	x			

Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page

FIFO (First in, First Out)

		0	1	0	2	0	1	3	0	2	3
Cadre 0		0	0	0	0	0	0	3	3		
Cadre 1			1	1	1	1	1	1	0		
Cadre 2					2	2	2	2	2		
D.P		x	x	-	x	-	-	x	x		

Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page

FIFO (First in, First Out)

		0	1	0	2	0	1	3	0	2	3
Cadre 0		0	0	0	0	0	0	3	3	3	
Cadre 1			1	1	1	1	1	1	0	0	
Cadre 2					2	2	2	2	2	2	
D.P		x	x	-	x	-	-	x	x	-	

Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page

FIFO (First in, First Out)

		0	1	0	2	0	1	3	0	2	3
Cadre 0		0	0	0	0	0	0	3	3	3	3
Cadre 1			1	1	1	1	1	1	0	0	0
Cadre 2					2	2	2	2	2	2	2
D.P		x	x	-	x	-	-	x	x	-	-

Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page

FIFO (First in, First Out)

		0	1	0	2	0	1	3	0	2	3
Cadre 0		0	0	0	0	0	0	3	3	3	3
Cadre 1			1	1	1	1	1	1	0	0	0
Cadre 2					2	2	2	2	2	2	2
D.P		x	x	-	x	-	-	x	x	-	-

5 D.P

Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page

LRU (Least Recently Used) = moins récemment utilisée

Principe

- **Choisir la page la moins récemment référencée c.à.d. celle qui a restée non utilisée le plus de temps**
  - **Chaque page est alors indexée par la date du dernier accès**

Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page

LRU (Least Recently Used) = moins récemment utilisée

Temps		0	1	2	3	4	5	6	7	8	9
		0	1	0	2	0	1	3	0	2	3
Cadre 0		0_0									
Cadre 1											
Cadre 2											
D.P		x									

Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page

LRU (Least Recently Used) = moins récemment utilisée

Temps		0	1	2	3	4	5	6	7	8	9
		0	1	0	2	0	1	3	0	2	3
Cadre 0		0_0	0_0								
Cadre 1			1_1								
Cadre 2											
D.P		x	x								

Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page

LRU (Least Recently Used) = moins récemment utilisée

Temps		0	1	2	3	4	5	6	7	8	9
		0	1	0	2	0	1	3	0	2	3
Cadre 0		0_0	0_0	0_2							
Cadre 1			1_1	1_1							
Cadre 2											
D.P		x	x	-							

Mise à jour de la date du dernier accès

Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page

LRU (Least Recently Used) = moins récemment utilisée

Temps		0	1	2	3	4	5	6	7	8	9
		0	**1**	**0**	**2**	**0**	**1**	**3**	**0**	**2**	**3**
Cadre 0		0_0	0_0	0_2	0_2						
Cadre 1			1_1	1_1	1_1						
Cadre 2					2_3						
D.P			x	x	-	x					

Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page

LRU (Least Recently Used) = moins récemment utilisée

Temps		0	1	2	3	4	5	6	7	8	9
		0	**1**	**0**	**2**	**0**	**1**	**3**	**0**	**2**	**3**
Cadre 0		0_0	0_0	0_2	0_2	0_4					
Cadre 1			1_1	1_1	1_1	1_1					
Cadre 2					2_3	2_3					
D.P		x	x	-	x	-					

Mise à jour de la date du dernier accès

Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page

LRU (Least Recently Used) = moins récemment utilisée

Temps		0	1	2	3	4	5	6	7	8	9
		0	**1**	**0**	**2**	**0**	**1**	**3**	**0**	**2**	**3**
Cadre 0		0_0	0_0	0_2	0_2	0_4	0_4				
Cadre 1			1_1	1_1	1_1	1_1	1_5				
Cadre 2					2_3	2_3	2_3				
D.P		x	x	-	x	-	-				

Mise à jour de la date du dernier accès

Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page

LRU (Least Recently Used) = moins récemment utilisée

Temps		0	1	2	3	4	5	6	7	8	9
		0	**1**	**0**	**2**	**0**	**1**	**3**	**0**	**2**	**3**
Cadre 0		0_0	0_0	0_2	0_2	0_4	0_4	0_4			
Cadre 1			1_1	1_1	1_1	1_1	1_5	1_5			
Cadre 2					2_3	2_3	2_3	3_6			
D.P		x	x	-	x	-	-	x			

Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page

LRU (Least Recently Used) = moins récemment utilisée

Temps		0	1	2	3	4	5	6	7	8	9
		0	**1**	**0**	**2**	**0**	**1**	**3**	**0**	**2**	**3**
Cadre 0		0_0	0_0	0_2	0_2	0_4	0_4	0_4	0_7		
Cadre 1			1_1	1_1	1_1	1_1	1_5	1_5	1_5		
Cadre 2					2_3	2_3	2_3	3_6	3_6		
D.P		x	x	-	x	-	-	x	-		

Mise à jour de la date du dernier accès

Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page

LRU (Least Recently Used) = moins récemment utilisée

Temps		0	1	2	3	4	5	6	7	8	9
		0	**1**	**0**	**2**	**0**	**1**	**3**	**0**	**2**	**3**
Cadre 0		0_0	0_0	0_2	0_2	0_4	0_4	0_4	0_7	0_7	
Cadre 1			1_1	1_1	1_1	1_1	1_5	1_5	1_5	2_8	
Cadre 2					2_3	2_3	2_3	3_6	3_6	3_6	
D.P		x	x	-	x	-	-	x	-	x	

Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page

LRU (Least Recently Used) = moins récemment utilisée

Temps		0	1	2	3	4	5	6	7	8	9
		0	**1**	**0**	**2**	**0**	**1**	**3**	**0**	**2**	**3**
Cadre 0		0_0	0_0	0_2	0_2	0_4	0_4	0_4	0_7	0_7	0_7
Cadre 1			1_1	1_1	1_1	1_1	1_5	1_5	1_5	2_8	2_8
Cadre 2					2_3	2_3	2_3	3_6	3_6	3_6	3_9
D.P		x	x	-	x	-	-	x	-	x	-

Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page

LRU (Least Recently Used) = moins récemment utilisée

Temps		0	1	2	3	4	5	6	7	8	9
		0	**1**	**0**	**2**	**0**	**1**	**3**	**0**	**2**	**3**
Cadre 0		0_0	0_0	0_2	0_2	0_4	0_4	0_4	0_7	0_7	0_7
Cadre 1			1_1	1_1	1_1	1_1	1_5	1_5	1_5	2_8	2_8
Cadre 2					2_3	2_3	2_3	3_6	3_6	3_6	3_9
D.P		x	x	-	x	-	-	x	-	x	-

5 D.P

Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page

NRU (Not Recently Used) = non récemment utilisée

Principe

- Utilise les bits R et M
- R mis à 1 chaque fois que la page est référencée
- M mis à 1 chaque fois que la page est modifiée
- Quand un processus démarre les 2 bits de toutes ses pages sont à 0
- À chaque quantum, R est remis à 0
- La page victime est celle qui a la valeur la plus petite de (R, M)

Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page

NRU (Not Recently Used) = non récemment utilisée

		0	1	0*	2	0	1*	3	0*	2	3
Cadre 0											
Cadre 1											
Cadre 2											
D.P						RAZ				RAZ	

- * est utilisée pour dire que la page a été modifiée
- Les bits R de toutes les pages sont remis à zéro après chaque quantum (Le quantum = 4 UT)

Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page

NRU (Not Recently Used) = non récemment utilisée

		0	1	0*	2	0	1*	3	0*	2	3
Cadre 0		0 (1,0)	0 (1,0)	0 (1,1)	0 (1,1)	0 (0,1)	0 (1,0)	3 (1,0)	3 (1,0)	3 (1,0)	3 (1,0)
Cadre 1			1 (1,0)	1 (1,0)	1 (1,0)	1 (0,0)	1 (1,0)	1 (1,0)	1 (1,0)	1 (1,0)	1 (1,0)
Cadre 2					2 (1,0)	2 (0,0)	2 (1,0)	2 (1,0)	2 (1,0)	2 (1,0)	2 (1,0)
D.P		x	x	-	x	-	-	x	x	-	-

RAZ **RAZ**

Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page

NRU (Not Recently Used) = non récemment utilisée

		0	1	0*	2	0	1*	3	0*	2	3
Cadre 0		0 (1,0)	0 (1,0)								
Cadre 1			1 (1,0)								
Cadre 2											
D.P		x	x								
						RAZ				RAZ	

Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page

NRU (Not Recently Used) = non récemment utilisée

		0	1	0*	2	0	1*	3	0*	2	3
Cadre 0		0 (1,0)	0 (1,0)	0 (1,1)							
Cadre 1			1 (1,0)	1 (1,0)							
Cadre 2											
D.P		x	x	-							

RAZ **RAZ**

Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page

NRU (Not Recently Used) = non récemment utilisée

		0	1	0*	2	0	1*	3	0*	2	3
Cadre 0		0 (1,0)	0 (1,0)	0 (1,1)	0 (1,1)						
Cadre 1			1 (1,0)	1 (1,0)	1 (1,0)						
Cadre 2					2 (1,0)						
D.P		x	x	-	x						
						RAZ				RAZ	

Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page

NRU (Not Recently Used) = non récemment utilisée

		0	1	0*	2	0	1*	3	0*	2	3
Cadre 0		0 (1,0)	0 (1,0)	0 (1,1)	0 (1,1)	0 (0,1)					
Cadre 1			1 (1,0)	1 (1,0)	1 (1,0)	1 (0,0)					
Cadre 2					2 (1,0)	2 (0,0)					
D.P		x	x	-	x	-					

RAZ **RAZ**

Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page

NRU (Not Recently Used) = non récemment utilisée

		0	1	0*	2	0	1*	3	0*	2	3
Cadre 0		0 (1,0)	0 (1,0)	0 (1,1)	0 (1,1)	0 (1,1)					
Cadre 1			1 (1,0)	1 (1,0)	1 (1,0)	1 (0,0)					
Cadre 2					2 (1,0)	2 (0,0)					
D.P		x	x	-	x	-					

RAZ **RAZ**

Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page

NRU (Not Recently Used) = non récemment utilisée

		0	1	0*	2	0	1*	3	0*	2	3
Cadre 0		0 (1,0)	0 (1,0)	0 (1,1)	0 (1,1)	0 (1,1)	0 (1,1)				
Cadre 1			1 (1,0)	1 (1,0)	1 (1,0)	1 (0,0)	1 (1,1)				
Cadre 2					2 (1,0)	2 (0,0)	2 (0,0)				
D.P		x	x	-	x	-	-				
					RAZ				RAZ		

Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page

NRU (Not Recently Used) = non récemment utilisée

		0	1	0*	2	0	1*	3	0*	2	3
Cadre 0		0 (1,0)	0 (1,0)	0 (1,1)	0 (1,1)	0 (1,1)	0 (1,1)	0 (1,1)			
Cadre 1			1 (1,0)	1 (1,0)	1 (1,0)	1 (0,0)	1 (1,1)	1 (1,1)			
Cadre 2					2 (1,0)	2 (0,0)	2 (0,0)	3 (1,0)			
D.P		x	x	-	x	-	-	x			
					RAZ				RAZ		

Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page

NRU (Not Recently Used) = non récemment utilisée

		0	1	0*	2	0	1*	3	0*	2	3
Cadre 0		0 (1,0)	0 (1,0)	0 (1,1)	0 (1,1)	0 (1,1)	0 (1,1)	0 (1,1)	0 (1,1)		
Cadre 1			1 (1,0)	1 (1,0)	1 (1,0)	1 (0,0)	1 (1,1)	1 (1,1)	1 (1,1)		
Cadre 2					2 (1,0)	2 (0,0)	2 (0,0)	3 (1,0)	3 (1,0)		
D.P		x	x	-	x	-	-	x	-		

RAZ **RAZ**

Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page

NRU (Not Recently Used) = non récemment utilisée

		0	1	0*	2	0	1*	3	0*	2	3
Cadre 0		0 (1,0)	0 (1,0)	0 (1,1)	0 (1,1)	0 (1,1)	0 (1,1)	0 (1,1)	0 (1,1)	0 (0,1)	
Cadre 1			1 (1,0)	1 (1,0)	1 (1,0)	1 (0,0)	1 (1,1)	1 (1,1)	1 (1,1)	1 (0,1)	
Cadre 2					2 (1,0)	2 (0,0)	2 (0,0)	3 (1,0)	3 (1,0)	3 (0,0)	
D.P		x	x	-	x	-	-	x	-	x	
						RAZ				RAZ	

Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page

NRU (Not Recently Used) = non récemment utilisée

		0	1	0*	2	0	1*	3	0*	2	3
Cadre 0		0 (1,0)	0 (1,0)	0 (1,1)	0 (1,1)	0 (1,1)	0 (1,1)	0 (1,1)	0 (1,1)	0 (0,1)	
Cadre 1			1 (1,0)	1 (1,0)	1 (1,0)	1 (0,0)	1 (1,1)	1 (1,1)	1 (1,1)	1 (0,1)	
Cadre 2					2 (1,0)	2 (0,0)	2 (0,0)	3 (1,0)	3 (1,0)	2 (1,0)	
D.P		x	x	-	x	-	-	x	-	x	

RAZ **RAZ**

Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page

NRU (Not Recently Used) = non récemment utilisée

		0	1	0*	2	0	1*	3	0*	2	3
Cadre 0		0 (1,0)	0 (1,0)	0 (1,1)	0 (1,1)	0 (1,1)	0 (1,1)	0 (1,1)	0 (1,1)	0 (0,1)	3 (1,0)
Cadre 1			1 (1,0)	1 (1,0)	1 (1,0)	1 (0,0)	1 (1,1)	1 (1,1)	1 (1,1)	1 (0,1)	1 (0,1)
Cadre 2					2 (1,0)	2 (0,0)	2 (0,0)	3 (1,0)	3 (1,0)	2 (1,0)	2 (1,0)
D.P		x	x	-	x	-	-	x	-	x	x

RAZ **RAZ**

6 D.P

Gestion de la mémoire virtuelle

3. Pagination

Algorithmes de remplacement de page

		0	1	0	2	0	1	3	0	2	3
Cadre 0		0	0	0	0						
Cadre 1			1	1	1						
Cadre 2					2						
D.P		x	x	-	x						

Gestion de la mémoire virtuelle

4. Segmentation VS Pagination

Critère	Pagination	Segmentation
Les procédures et les données sont séparées et protégées séparément	✘	✓
Partage des fonctions entre utilisateurs	✘	✓
Fragmentation interne	✓	✘
Fragmentation externe	✘	✓
Visible au programmeur	✘	✓

Gestion de la mémoire virtuelle

5. Segmentation paginée

- Les segments sont divisés en des pages
- Enlève la restriction de la continuité des segments
- La table de segments indique pour chaque segment l'adresse de sa table de pages

- L'adresse logique est :

(Numéro de segment, Numéro de page, Déplacement à faire dans cette page)

Etude de cas

1. Cas de Linux

32 bits => 2^{32} adresses absolues différentes➔ 4 Go de mémoire adressable

Chaque processus a un espace d'adressage virtuel de 4 GØ (1GØ est réservé aux tables du processus)

- **L'espace d'adressage est composé d'un ensemble de segments (maximum 16) paginés (pages de 4Kø)**
- **Le SE se charge de maintenir un certain nombre de pages libres en mémoire**
  - **Il vérifie périodiquement ou après une forte allocation d'espace, l'espace disponible**
  - **Si l'espace disponible devient insuffisant, il libère certains cadres**

Etude de cas

1. Cas de Linux

Algorithmes de remplacement de page

Clock = Horloge

Principe

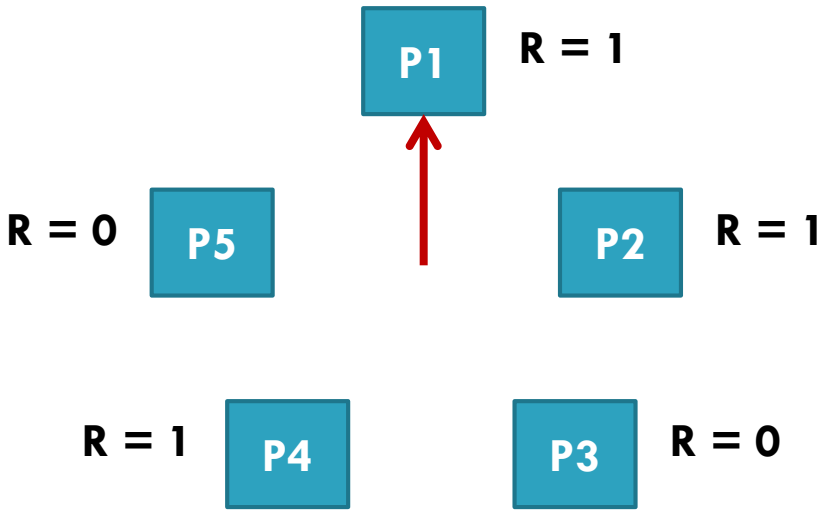
- Les cadres forment conceptuellement une liste circulaire
- Un pointeur est initialement sur la page la plus ancienne
- La première page rencontrée ayant son bit de référence R à 0 est remplacée. Le bit R de la page ajoutée est à 1 et le pointeur est avancé d'une position
- Si le bit R d'une page examinée est 1, il est mis à 0 et le pointeur est avancé d'une position

Etude de cas

1. Cas de Linux

Algorithmes de remplacement de page

Clock = Horloge



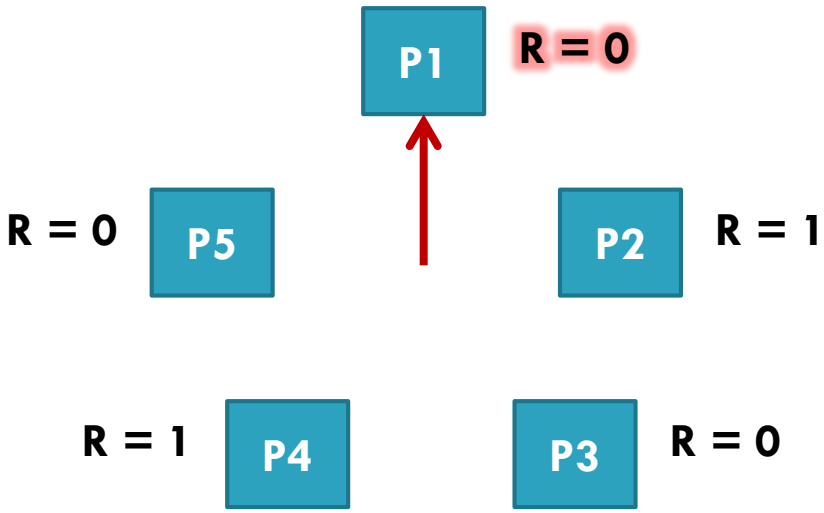
P6 La page 6 est demandée

Etude de cas

1. Cas de Linux

Algorithmes de remplacement de page

Clock = Horloge



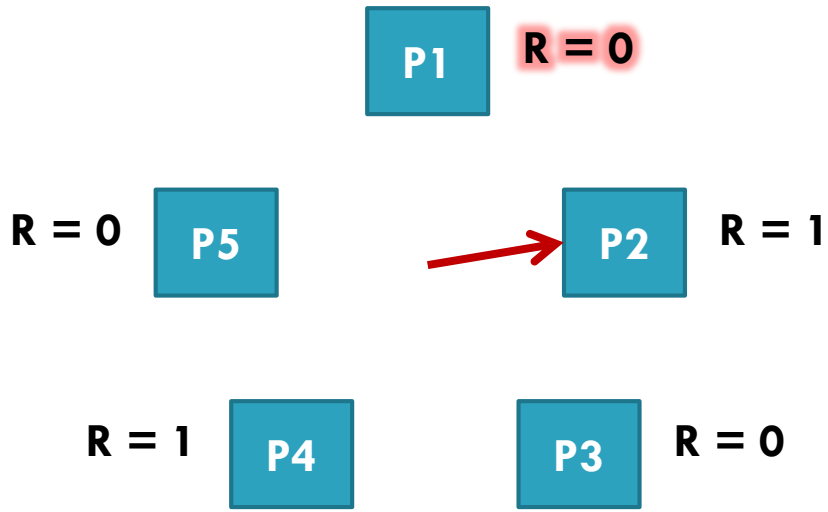
P6 La page 6 est demandée

Etude de cas

1. Cas de Linux

Algorithmes de remplacement de page

Clock = Horloge



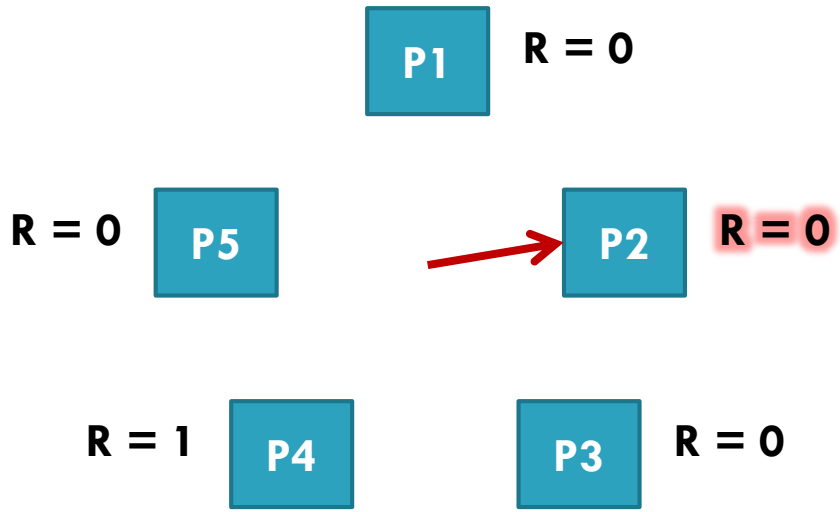
P6 La page 6 est demandée

Etude de cas

1. Cas de Linux

Algorithmes de remplacement de page

Clock = Horloge



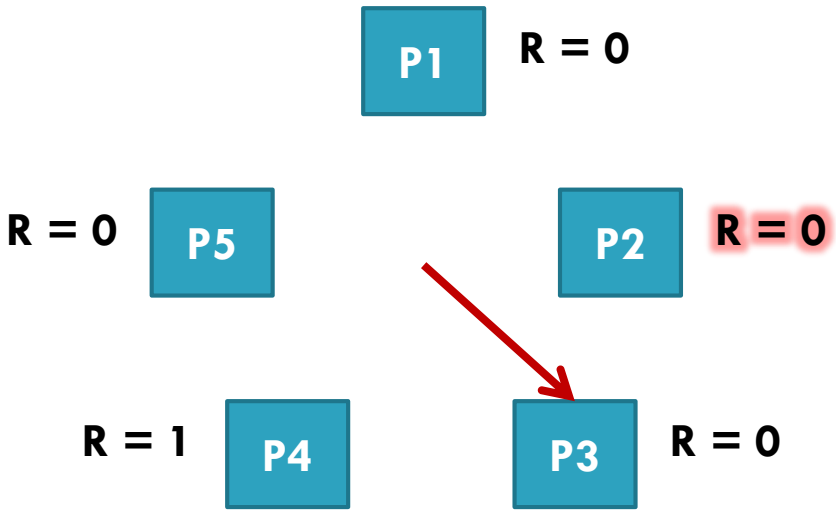
P6 La page 6 est demandée

Etude de cas

1. Cas de Linux

Algorithmes de remplacement de page

Clock = Horloge



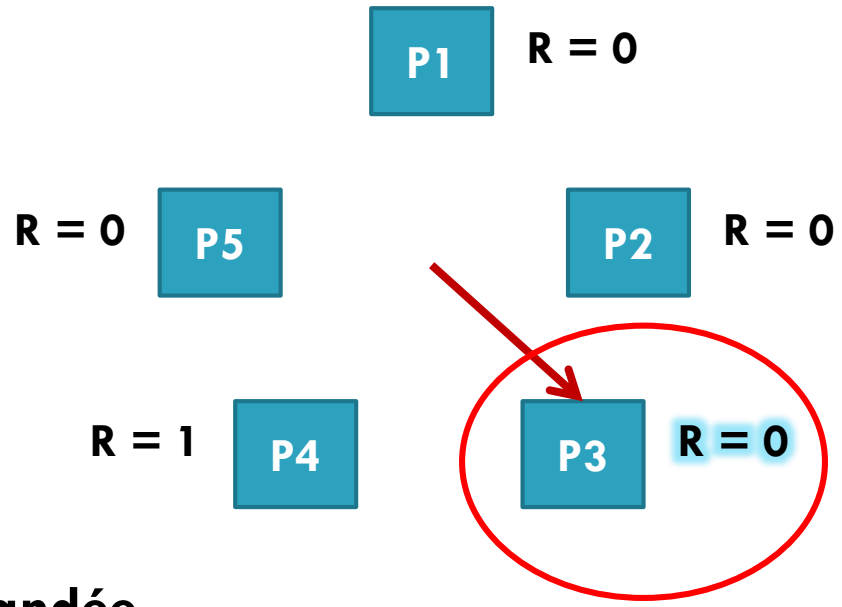
P6 La page 6 est demandée

Etude de cas

1. Cas de Linux

Algorithmes de remplacement de page

Clock = Horloge



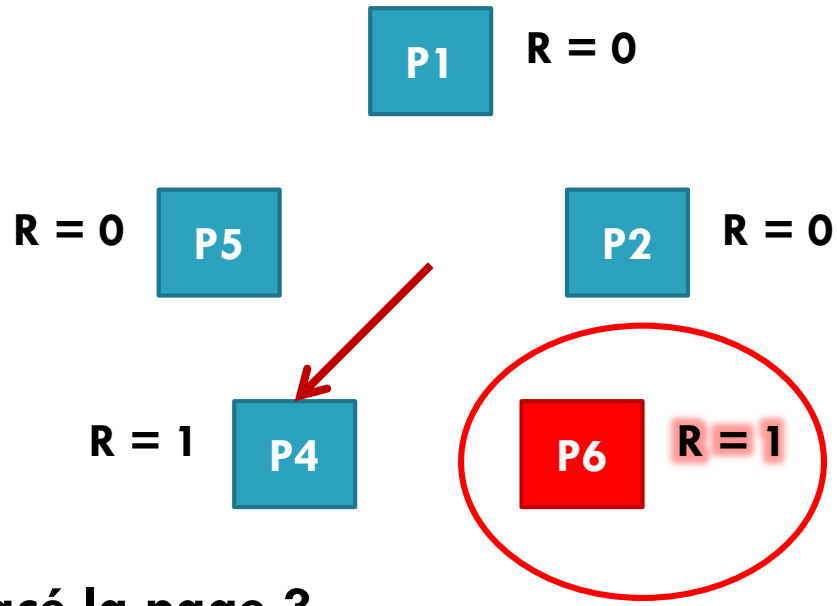
P6 La page 6 est demandée

Etude de cas

1. Cas de Linux

Algorithmes de remplacement de page

Clock = Horloge



P6 La page 6 a remplacé la page 3

Etude de cas

2. Cas de Windows

- Chaque processus a un espace d'adressage virtuel (4 GB).

  L'espace d'adressage virtuel est paginé (pas de segmentation).
- La taille d'une page est de 4 Ko
- Chaque processus a un espace de travail (l'ensemble des pages qu'un processus se sert couramment)
- Le SE effectue le chargement préalable (Préchargement: *Prepaging*) de l'ensemble de travail lorsqu'un processus est lancé et ce pour minimiser les défauts de pages

Etude de cas

2. Cas de Windows

Working Set = Ensemble de travail

Principe

- **Si un défaut de page se produit et l'espace de travail du processus qui a provoqué ce défaut de page est inférieur à une certaine limite, la page est chargée en mémoire. Elle est donc ajoutée à l'espace de travail du processus. Sinon on élimine une page qui n'est pas dans l'ensemble de travail du processus actif :**
  - **Si R = 0 et le temps de résidence de la page en mémoire $\geq \tau$**
  - **Si toutes les pages ont leur R = 0 et leur temps de résidence $< \tau$ alors choisir celle qui a le plus grand temps de résidence**

FIN

Madame Khaoula ElBedoui-Maktouf

2ème année Ingénieur Informatique