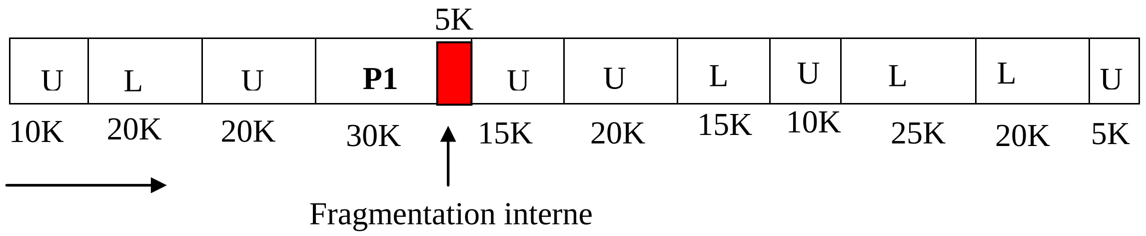
Corrigé de la série de TD3- Gestion de la mémoire centrale

<u>Exercice 1</u> : (Partition multiple fixe)

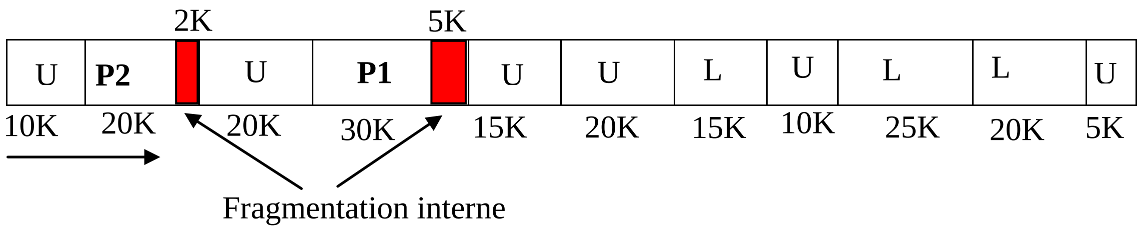
1) **Illustration de l’état de la mémoire centrale après chaque requête**

a) ***<u>Requête 1</u>* <u>de 25 KØ</u> :** la seconde partition libre (30K) peut loger la requête. L’état de la mémoire après chargement de cette requête sera comme suivant :



La requête 1 crée une fragmentation interne de 5 K.

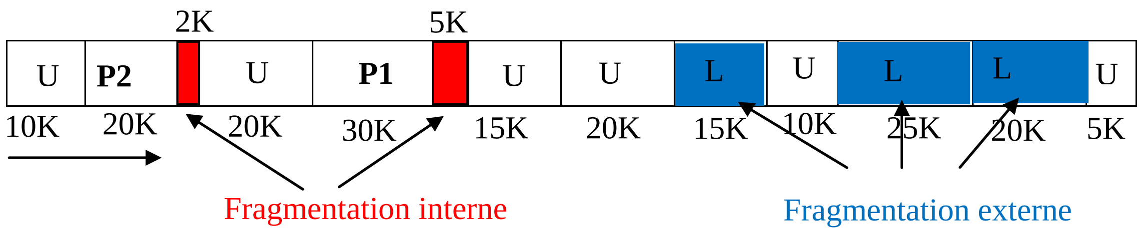
b) ***<u>Requête 21</u>* <u>de 18 KØ</u> :** la première partition libre (20K) peut loger la requête. L’état de la mémoire après chargement de cette requête sera comme suivant :



La requête2 crée une fragmentation interne de 2 K.

2) Existe-il des requêtes non exécutées ? Si oui, comment le système va procéder dans ce cas?

<u>Réponse :</u> Oui, la **requête 3** ne pourra pas être chargée, car aucune partition libre ne peut la contenir (supérieure ou égale à la taille de la requête). Le système met en attente cette requête jusqu’à ce qu’une partition pouvant la contenir soit libérée (dans ce cas la partition n° 4 de 30K) pour qu’il puisse la charger et l’exécuter. Dans ce cas, on a le problème de la **fragmentation externe**.



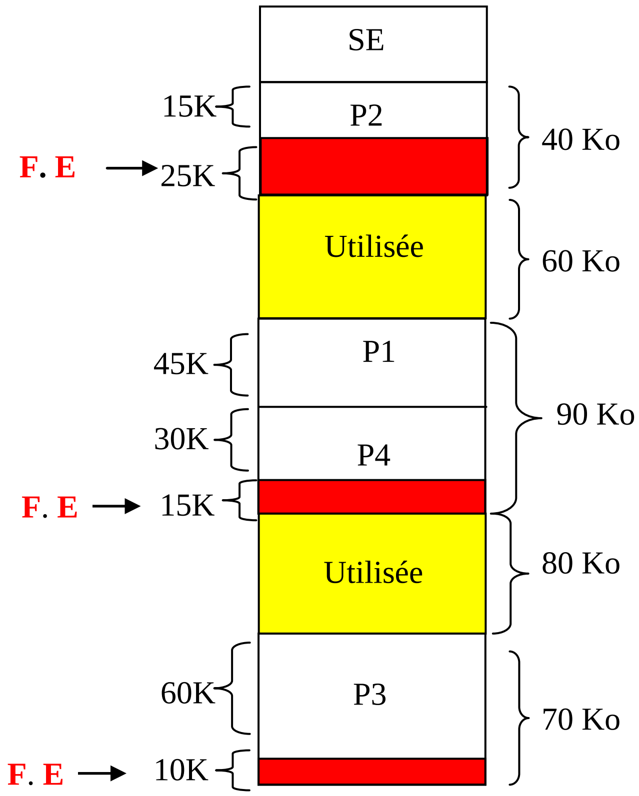
Exercice 2 :(Partitions multiples variables)

1) Placement des processus P1, P2, P3, P4 et P5, dans cet ordre, de tailles respectivement de 45 KØ, 15 KØ, 60 KØ, 30 KØ et 40 KØ, dans la mémoire centrale.

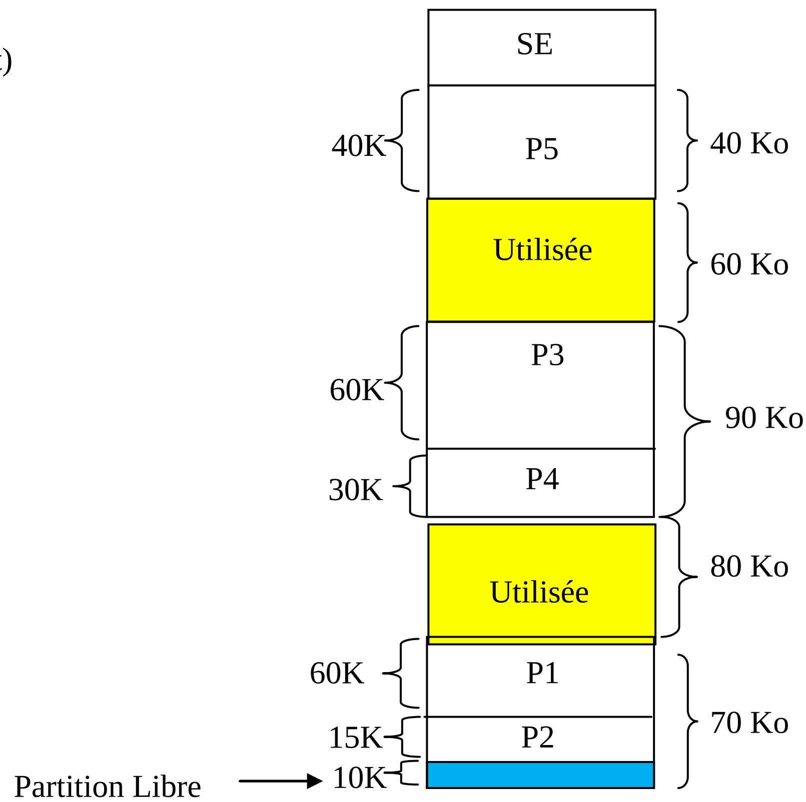
A) First fit (Algorithme du premier qui convient) :

Problème : P5 non chargé ?

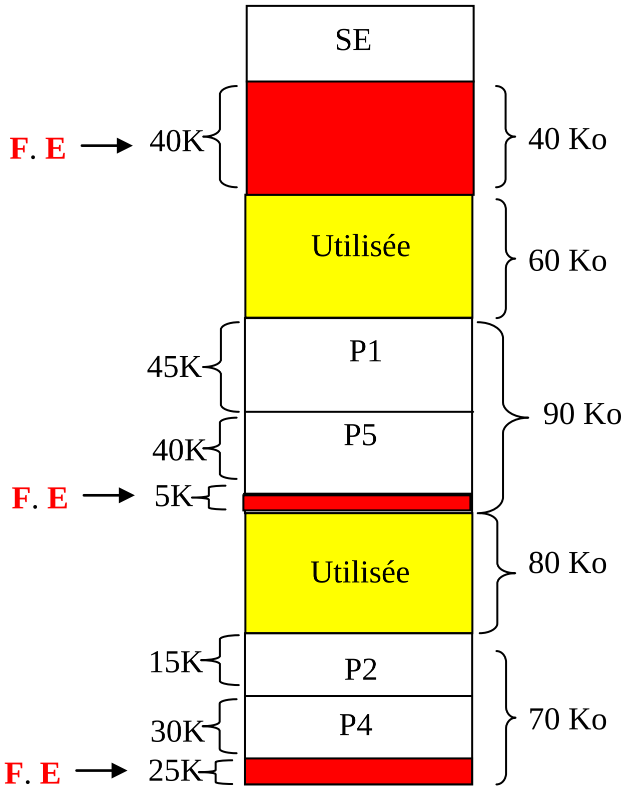
En utilisant l'algorithme first fit, le processus P5 dont la taille 40K <= l'espace total disponible de 50KØ (25+15+10), alors ces partitions éparpillées dans la mémoire forment de la fragmentation externe (**F.E**).



b) **Best fit (**Algorithme du meilleur qui convient)



c) **Worst fit** (Algorithme du pire qui convient)



Problème : P3 non chargé ?

De même, en utilisant l'algorithme Worst fit, le processus P3 est en attente d'exécution et sa taille est de 60K <= l'espace total disponible qui est de 70 **KØ** (40+5+25), alors ces partitions éparpillées dans la mémoire forment de la fragmentation externe (**F.E**).

Remarque : quand des processus sont en attente d'exécution :

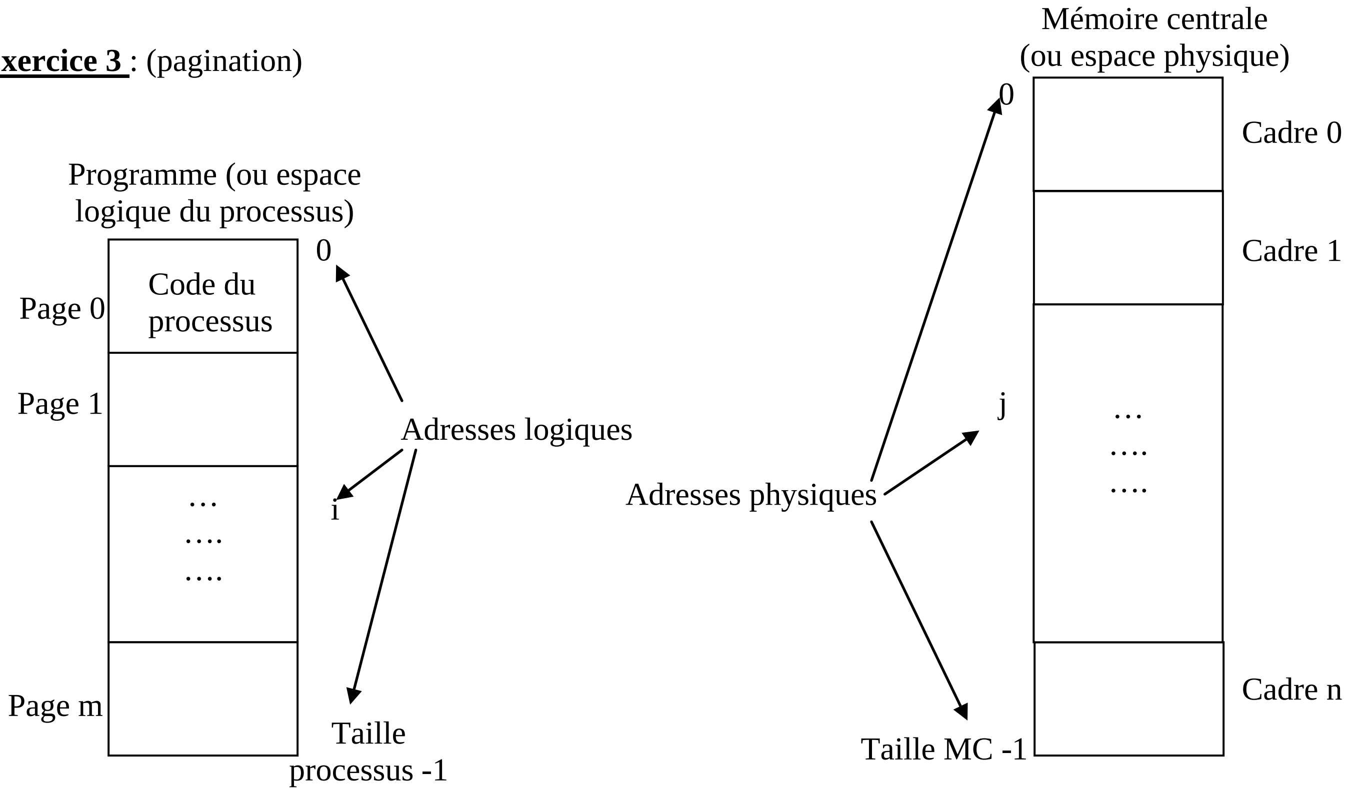
a) Si la taille de ces processus inferieur ou égal à l'espace total disponible, alors ces partitions non occupées forment la fragmentation externe (F.E), puisque la défragmentation permet à ces processus de quitter cet état d'attente.

b) Dans le cas contraire, ces partitions ne forment pas la fragmentation externe, puisque la défragmentation ne permet pas à ces processus de quitter l'état d'attente.

2) Le meilleur algorithme est Best fit, car **aucun processus n'est resté** en attente contrairement à first-fit et worst-fit.

3) Le problème constaté est que des processus ne peuvent pas s'exécuter, malgré que l'espace total disponible soit supérieur ou égal à leurs tailles. Ce problème est dû au fait que cet espace mémoire est dispersé (non contigu). Pour résoudre ce problème, il faut faire le **Compactage (**c.-à-d., **défragmentation de la mémoire)**.

Exercice 3 : (pagination)



Données : taille de l'espace physique = 64 M Ø,

Nombre de pages de l'espace logique du processus = 32 pages,

Taille d'une page = 2 KØ.

1 KØ = 1024 = 2^{10} Ø (Octets).

1MØ = 1024 KØ = 1024*1024 Ø = $2^{10}*2^{10}=2^{20}$ Ø.

1) Le nombre de bits nécessaires pour coder le numéro de page (*p*).

On a 32 pages. 32 = 2^5, donc il faut **5 bits pour représenter les 32 pages**

2) Le nombre de bits nécessaires pour le déplacement dans la page (*d*).

On a la taille d'une page = 2 KØ = $2*2^{10}$ Ø = 2^{11} Ø. Donc, il faut **11 bits pour représenter les octets d'une page.**

3) Déduction de la taille de l'adresse logique en bits.

Une adresse logique paginée (p, d) est composée des bits pour coder le numéro de la page (p) et des bits pour coder le déplacement dans la page (d). Donc,

Nombre_bits_adresse_logique =Nbre_bits_page (p)+ Nbre_bits_deplacement (d) = 5 + 11

= **16 bits**

4) Taille d'un cadre (de pages) en octets.

La taille d'un cadre de page = la taille d'une page =2 KØ =2* 1024= **2048 Ø**.

5) Le nombre de bits nécessaires pour coder un cadre.

Pour déterminer le nombre de bits nécessaires pour coder un cadre, il faut d'abord déterminer le nombre de cadres dans l'espace physique (ou mémoire physique).

Pour cela, il faut diviser la taille de l'espace physique sur la taille d'un cadre pour trouver combien de cadres sont dans la MC.

Le nombre de cadres =taille de la mémoire (espace physique) / taille d'un cadre.

= 64 MØ /2 KØ

$=2^6 * 2^{20} /2*2^{10}$

$= 2^{15}$.

Donc, il faut **15 bits**.

6) Adresse logique paginée associée à l'adresse logique en binaire suivante :

$(\underbrace{1100\ 1}_{p}\ \underbrace{010\ \ 1101\ \ 0101}_{d})$.

On a, $P = (11001)_2 = 2^4+2^3+2^0 =16+8+1= 25$, et

$d= (010\ 1101\ 0101)_2 = 2^9+2^7+2^6+2^4+2^2+2^0= 512+128+64+16+4+1= 721$

Donc, (p, d) = **(25, 721)**

7) Adresse paginée correspondant à l'adresse décimale A =5240.

P= A/ T = 5240/ 2048 = 2 (T : taille d'une page)

d= A % T = 5240 % 2048 = 1144

⇨ (p, d) = **(2, 1144)**

Exercice 3 : (Segmentation paginée)

1) **Segmentation**

Calcul des adresses physiques (@f) associées aux dresses logiques segmentées (S, d) suivantes:

- **@L (0,85)** : il s'agit du segment S=0 et du déplacement d= 85.

  0<=S <=2 et 0<= d <longueur (L) =90, donc, l'@p= Base +d =150+85= **235.**

- **@L (0, 95)** : il s'agit du segment S=0 et du déplacement d= 95.

  0<=S <=2, mais L=90 $\nless$ L= 90, donc **erreur d'adressages**.

- **@L (1, 290)** : il s'agit du segment S=1 et du déplacement d= 290.

  0<=S <=2, mais L=290 $\nless$ L=2 90, donc **erreur d'adressages.**

- **@L (1,220)** : il s'agit du segment S=1 et du déplacement d= 220.

  0<=S <=2 et 0<= d <290, donc, l'@p= Base +d =850+220= **1070**

- **@L (2,30)** : il s'agit du segment S=2 et du déplacement d= 30.
  0<=S <=2 et 0<= d <200, donc, l'@p= Base +d =1500+30= **1530.**

- **@L (3, 80)** : le segment S=3 n'existe pas, donc **erreur d'adressages**.

2) **Segmentation paginée**

Calcul des adresses physiques (S, f, d) correspondants aux adresses logiques segmentés données en 1).
On a, la mémoire centrale de 16 cadres et la taille d'un cadre = taille d'une page =100 octets.

Pour implémenter les segments par des pages, il faut transformer le déplacement dans le segment (d) en un numéro de page (**p**) et un déplacement dans la page (**d'**) comme suit :
P = d / T (T : taille d'un cadre ou d'une page), **d'** = d % T.

- **@L (S, d) = (0, 85)**: on a d=85 => p= 85/ 100 = 0 et d' = 85%100 = 85. Donc, l'adresse segmentée (S, d) = (0, 85) devient adresse segmentée paginée (S, p, d') = (0, 0, 85). D'après la table des pages des segments, la page 0 du segment 0 est dans le cadre f=11. Donc l'adresse physique segmentée paginée est (S, f, d') = **(0, 11, 85)** et l'adresse physique linéaire est =11*100+85 = **1185**.

- **@L (S, d) = (0, 95)** : le déplacement $d= 95 \nless L=90$, donc **erreur d'adressages**.

- **@L (S, d) = (1, 290)** : le déplacement $d=290 \nless L=290$, donc **erreur d'adressages.**

- **@L (S, d) = (1, 220)**: on a d=220 => p= 220/ 100= 2 et d' = 220%100 = 20. Donc, l'adresse segmentée (S, d) = (1, 220) devient adresse segmentée paginée (S, p, d') = (1, 2, 20). D'après la table des pages des segments, la page 2 du segment 1 est dans le cadre f=10. Donc l'adresse physique segmentée paginée est (S, f, d') = **(1,10, 20)** et l'adresse physique linéaire est =10*100+20 = **1020**.

- **@L (S, d) = (2, 30):** on a d=30 => p= 30/ 100= 0 et d' = 30%100 = 30. Donc, l'adresse segmentée (S, d) = (2, 30) devient adresse segmentée paginée (S, p, d') = (2, 0, 30). D'après la table des pages des segments, la page 0 du segment 2 est dans le cadre f=8. Donc l'adresse physique segmentée paginée est (S, f, d') = **(2, 8, 30)** et l'adresse physique linéaire est =80*100+30 = **830**.
- **@L (S, d) = (3, 80)** : le segment S=3 n'existe pas, donc **erreur d'adressages**.