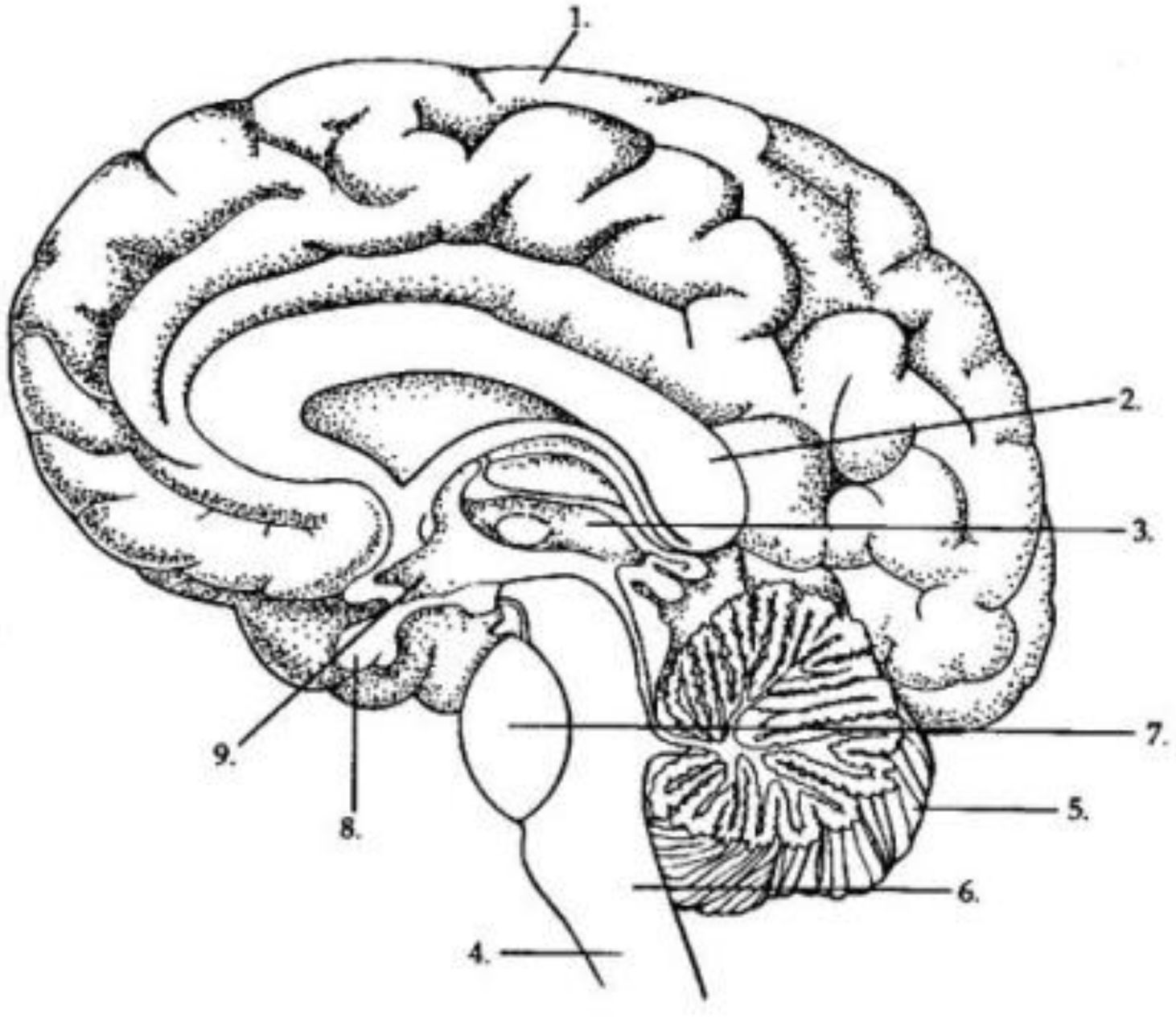
Allocation mémoire contigüe

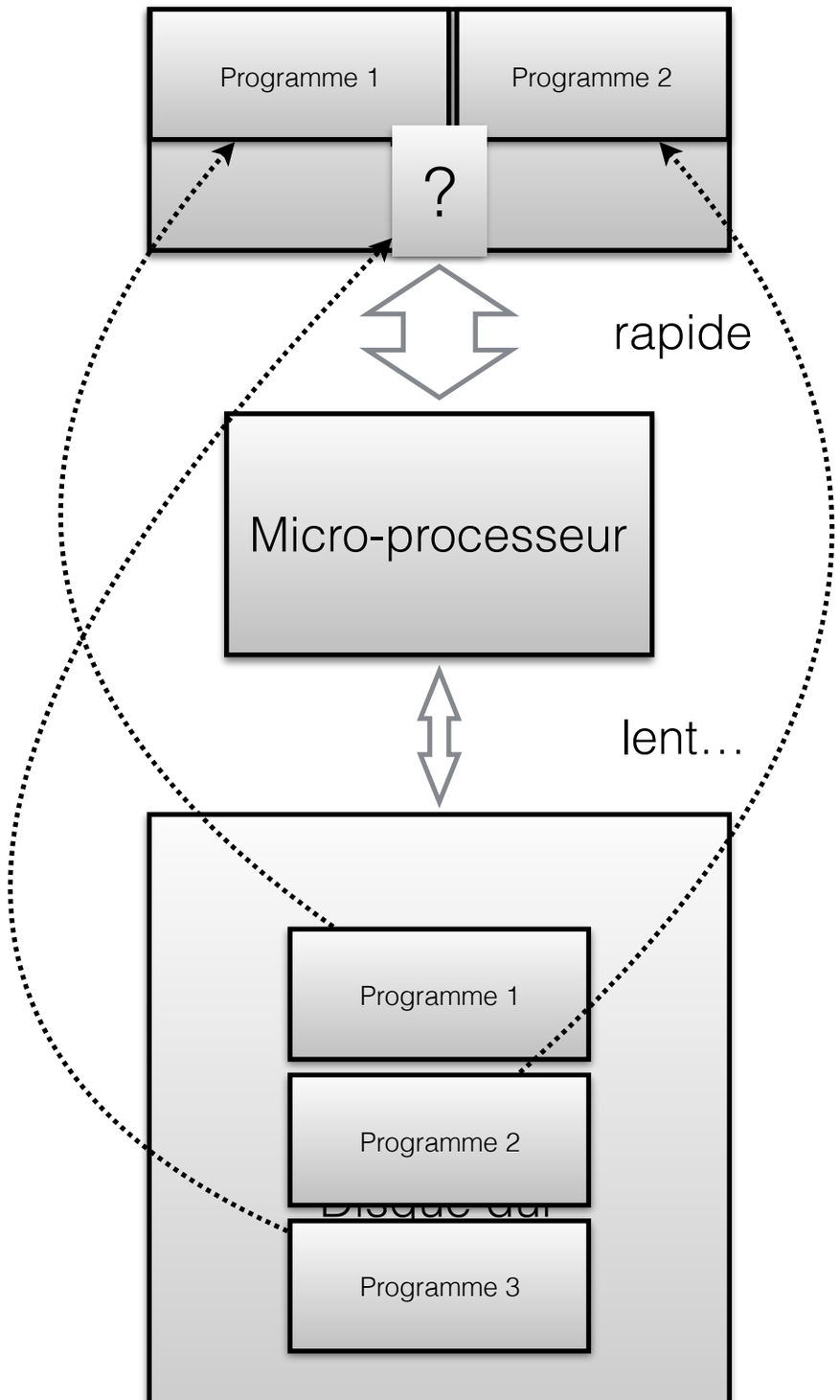


GIF-1001 Ordinateurs: Structure et Applications

Jean-François Lalonde

Allocation mémoire: objectif

- Un ordinateur possède *plusieurs* programmes
- Où ces programmes sont-ils situés?
  - Sur le disque dur
- Peut-on les exécuter s'ils sont sur le disque dur?
  - Non, le disque dur est un périphérique de stockage *lent*
- Donc, il nous faut les transférer dans la mémoire principale (RAM)
- L'objectif de la gestion mémoire est de *partager la mémoire RAM entre les divers programmes*



Allocation mémoire

- Objectif principal:
  - partager la mémoire RAM entre les divers programmes
- Objectifs secondaires:
  - Utilisation simple pour un programme
  - Maximiser l'utilisation de la mémoire disponible

Allocation mémoire: 2 stratégies

- Allocation **contigüe**: le programme occupe un **espace contigu** en mémoire
  - partitions de taille **fixe**
  - partitions de taille **variable**
- Allocation **paginée**: le programme est divisé en petits morceaux (pages) qui peuvent être répartis à **différents endroits**.

Allocation mémoire contigüe

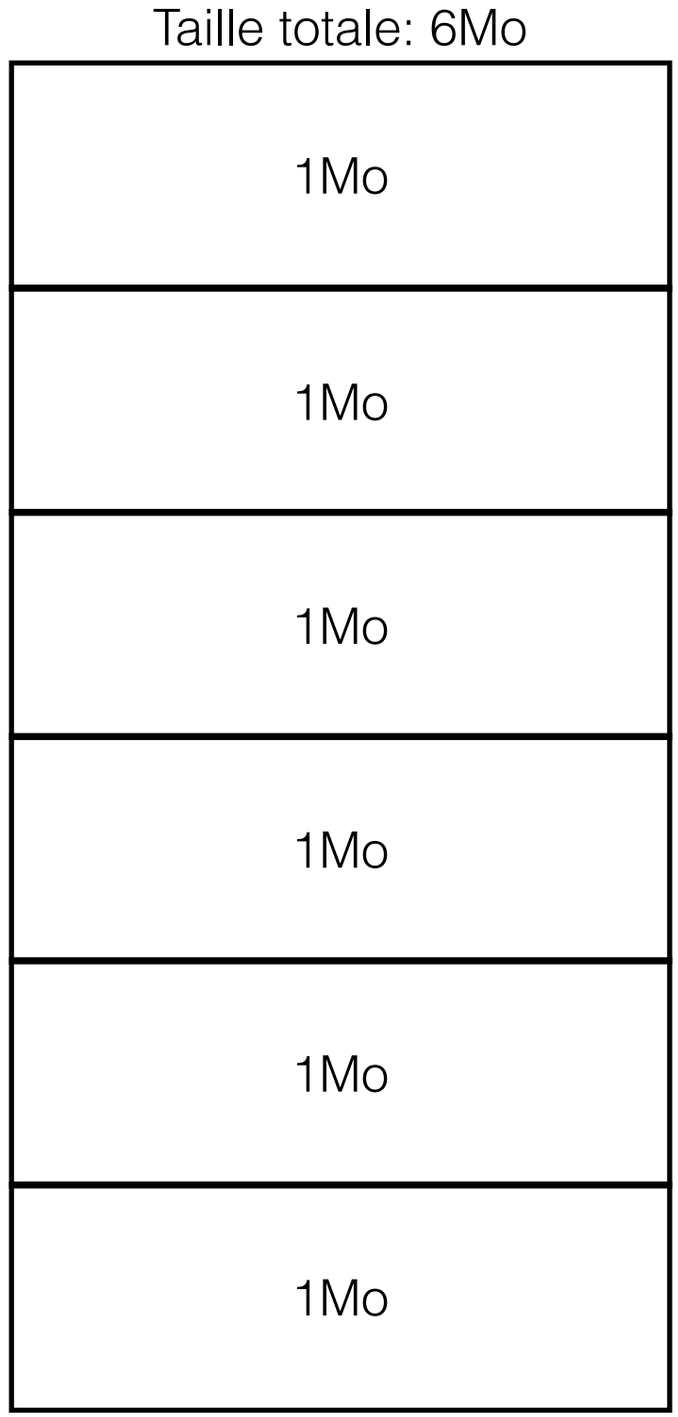
Allocation mémoire contigüe

- On réserve un «bloc» (ou «fragment») de mémoire contigu.
- Les blocs de mémoire peuvent avoir une taille:
  - **fixe**: la taille des blocs est prédéterminé, et est la même pour tous les processus.
  - **variable**: chaque processus se voit allouer un bloc de taille différente.

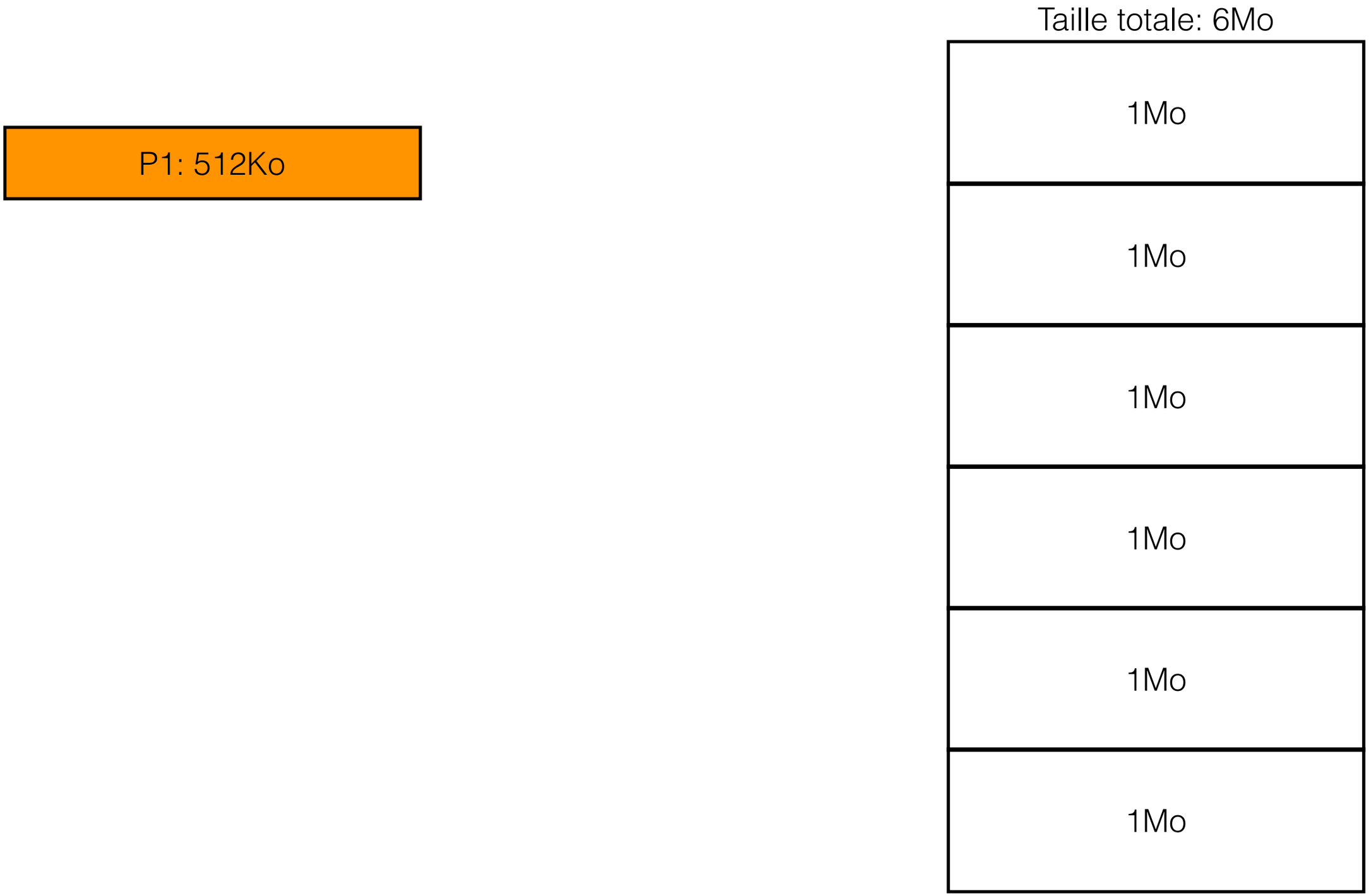
Allocation mémoire contigüe, taille **fixe**

- On divise la mémoire en blocs de taille **fixe**, par exemple 1Mo

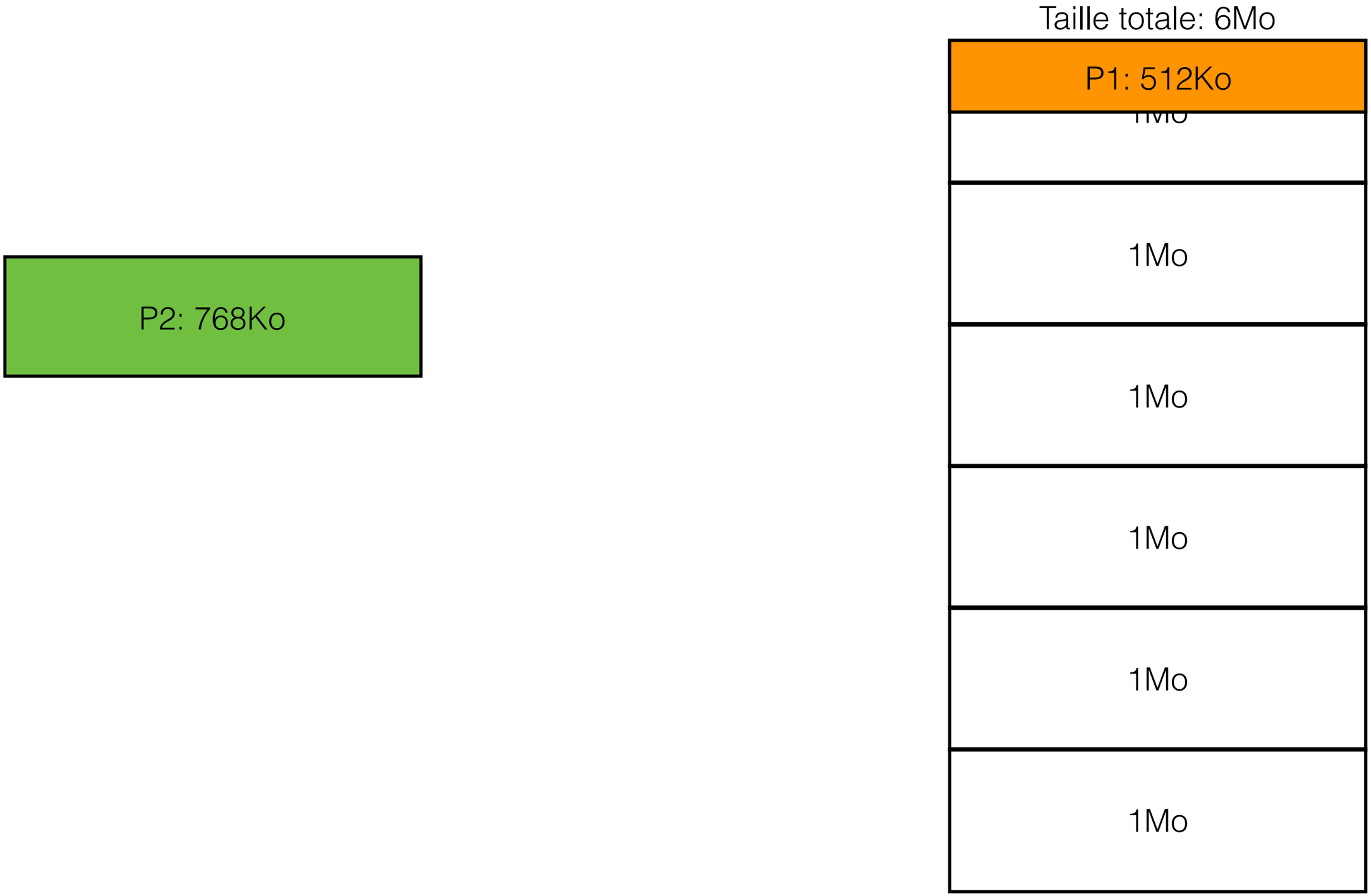
- Chaque processus est placé dans un bloc libre



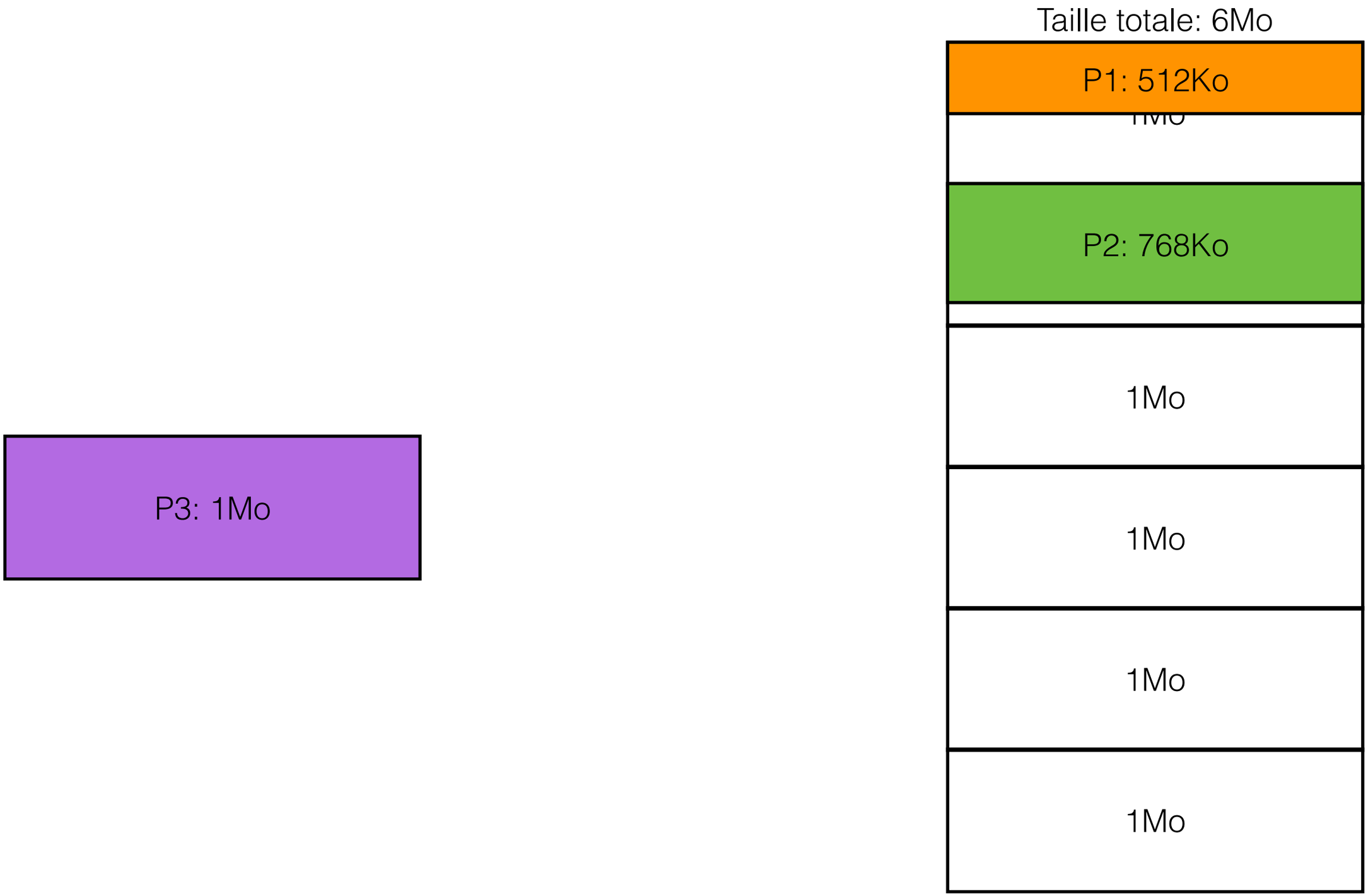
Allocation mémoire contigüe, taille **fixe**



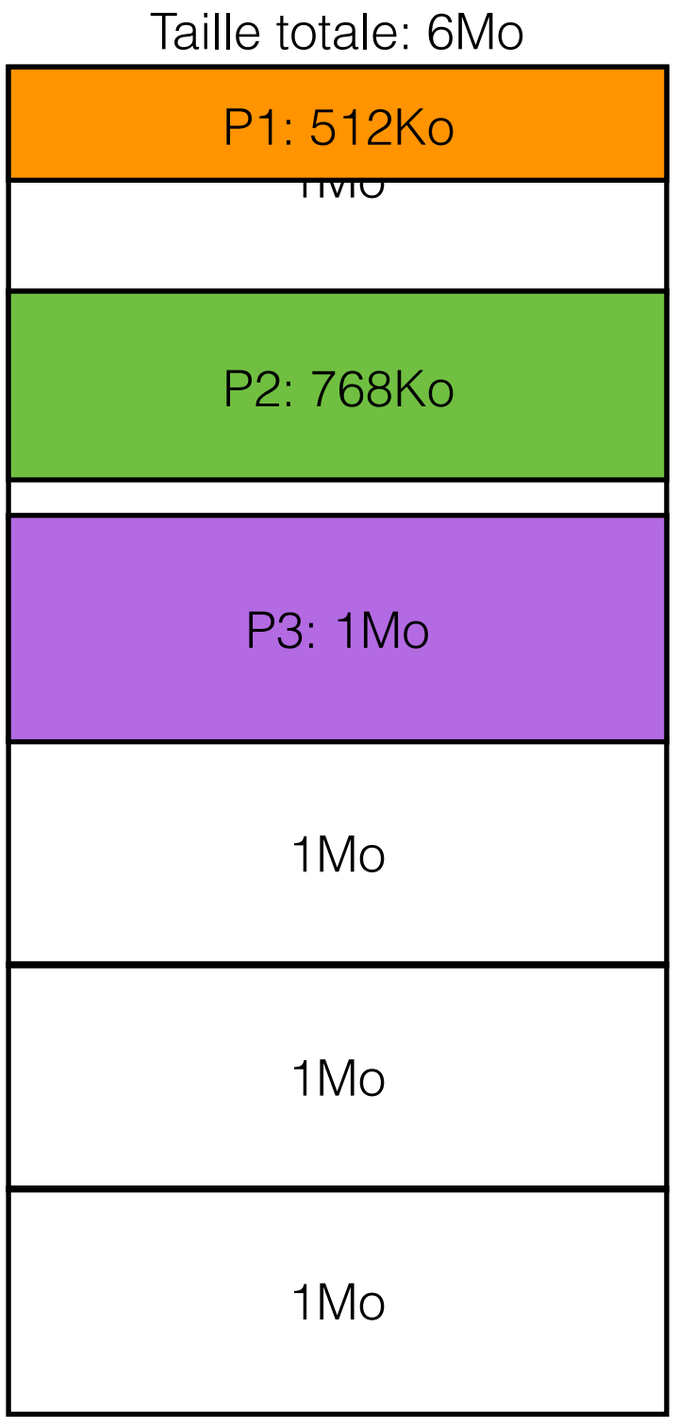
Allocation mémoire contigüe, taille **fixe**



Allocation mémoire contigüe, taille **fixe**



Allocation mémoire contigüe, taille **fixe**

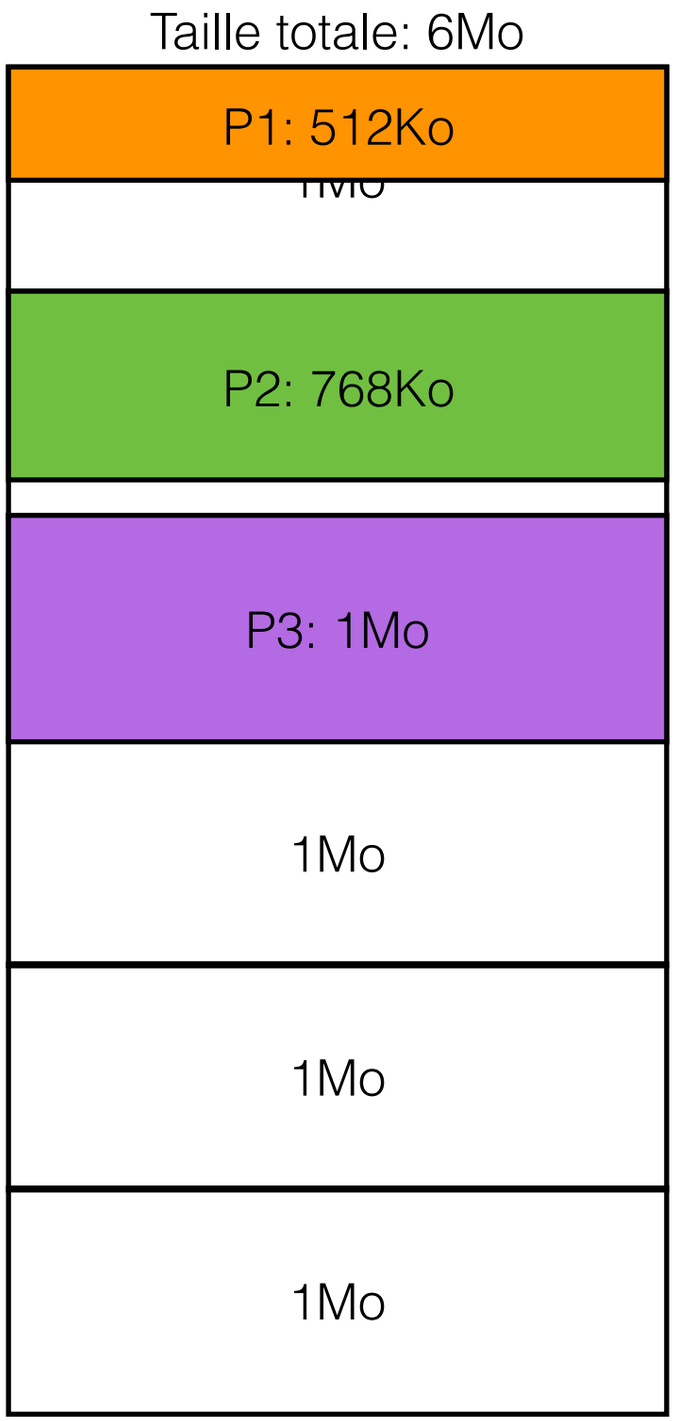


P4: 1.5Mo

Ne peut être admis en mémoire!

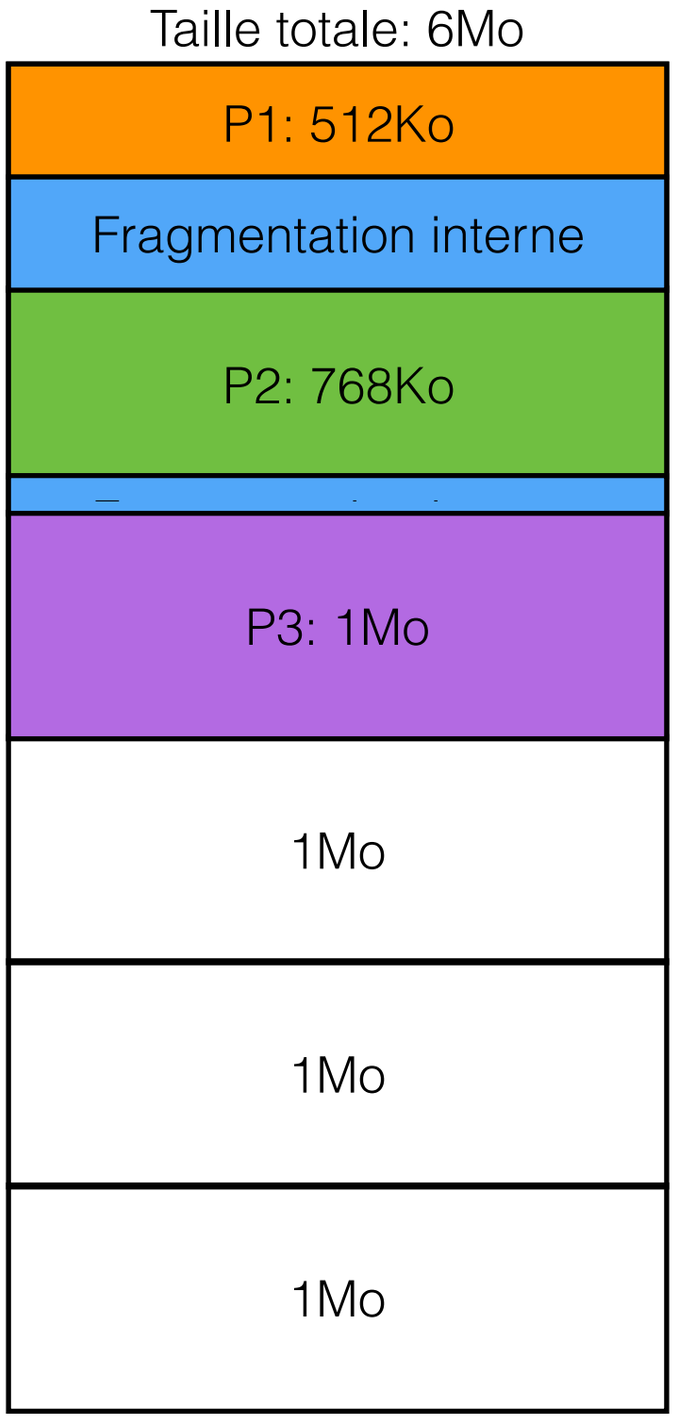
Allocation mémoire contigüe, taille **fixe**

- Quels sont les problèmes avec l'allocation contigüe avec partitions de taille fixe?
  - Ne peut allouer un programme si sa taille dépasse celle d'une partition
  - Beaucoup d'espace mémoire perdu: **fragmentation**!



Fragmentation

- Il en existe 2 types:
  - Fragmentation **interne**: espace perdu **à l'intérieur** une partition
  - Fragmentation **externe**: espace perdu **à l'extérieur** d'une partition
- Avec des partitions de taille **fixe**, seule la fragmentation **interne** est possible
  - aucun espace à l'extérieur d'une partition n'est perdu—il peut toujours être alloué à un autre processus peu importe son emplacement



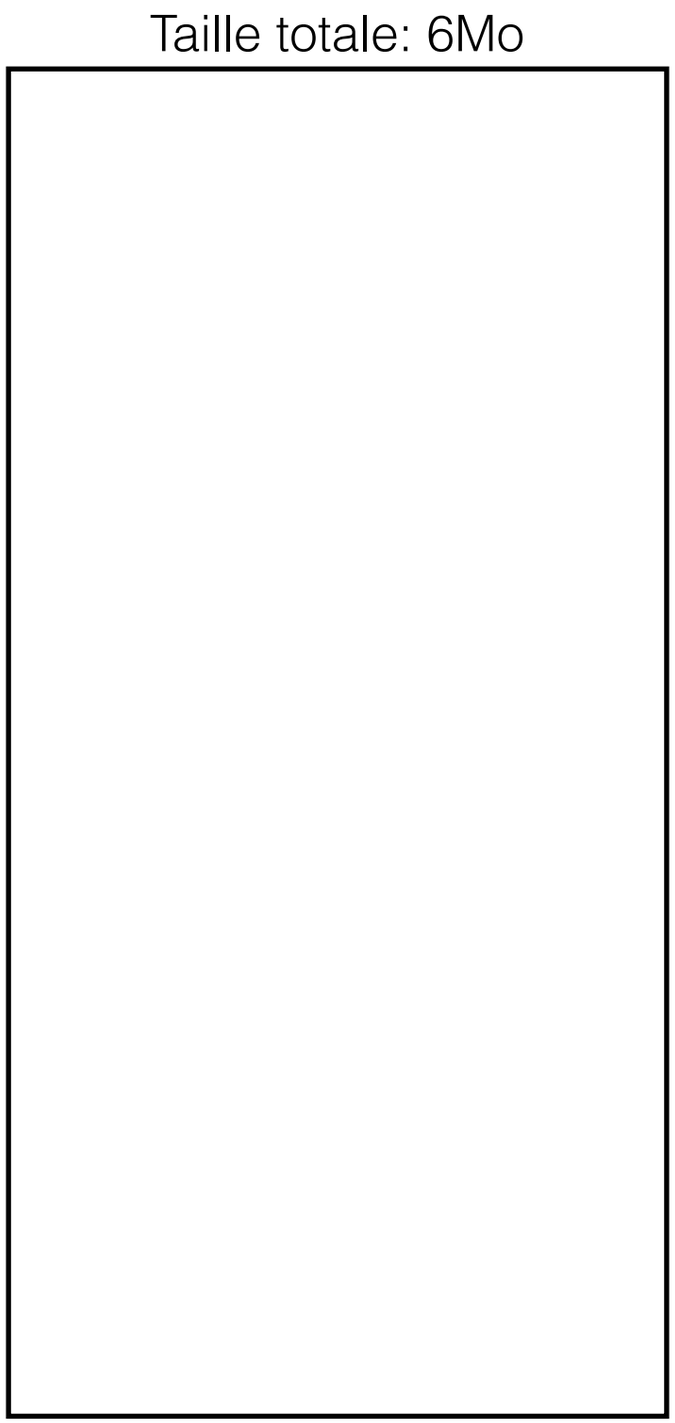
Allocation mémoire contigüe

- On réserve un «bloc» (ou «fragment») de mémoire contigu.
- Les blocs de mémoire peuvent avoir une taille:
  - **fixe**: la taille des blocs est prédéterminé, et est la même pour tous les processus.
  - **variable**: chaque processus se voit allouer un bloc de taille différente.

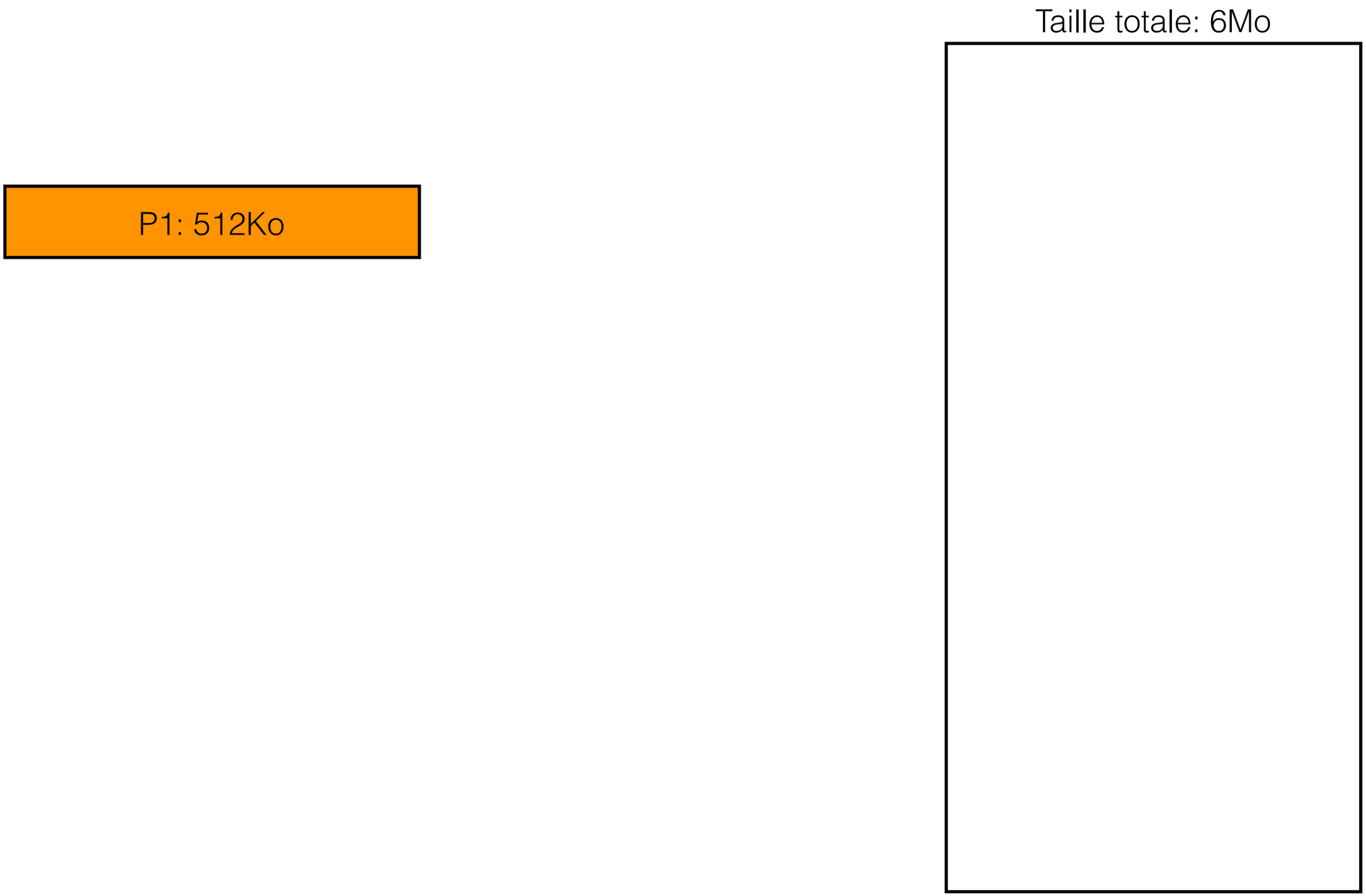
Allocation mémoire contigüe, taille **variable**

- On crée une partition de la bonne taille pour chaque processus.

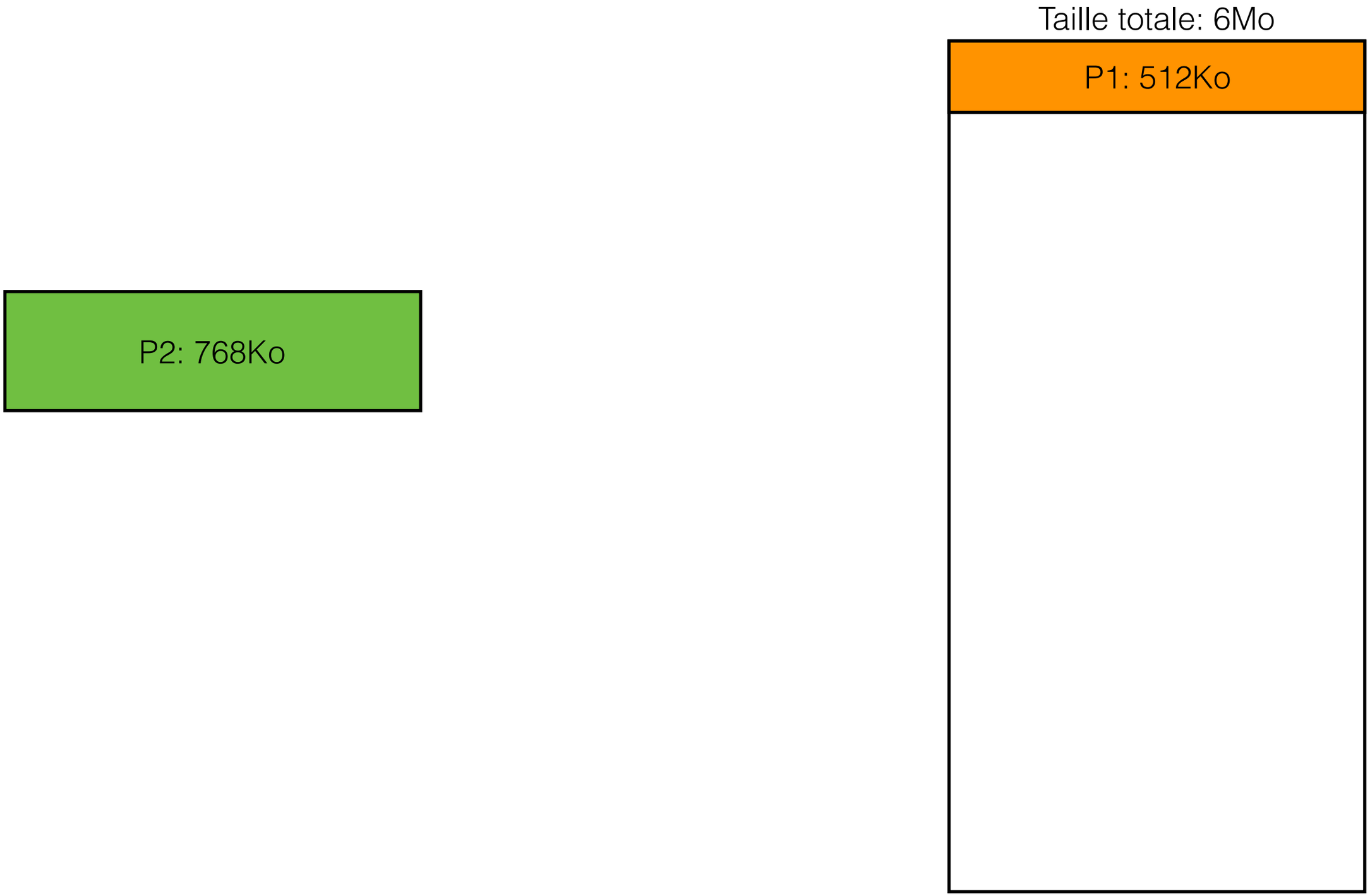
- Il peut parfois y avoir plusieurs endroits où une partition peut être créée: il faudra déterminer la bonne stratégie à employer!



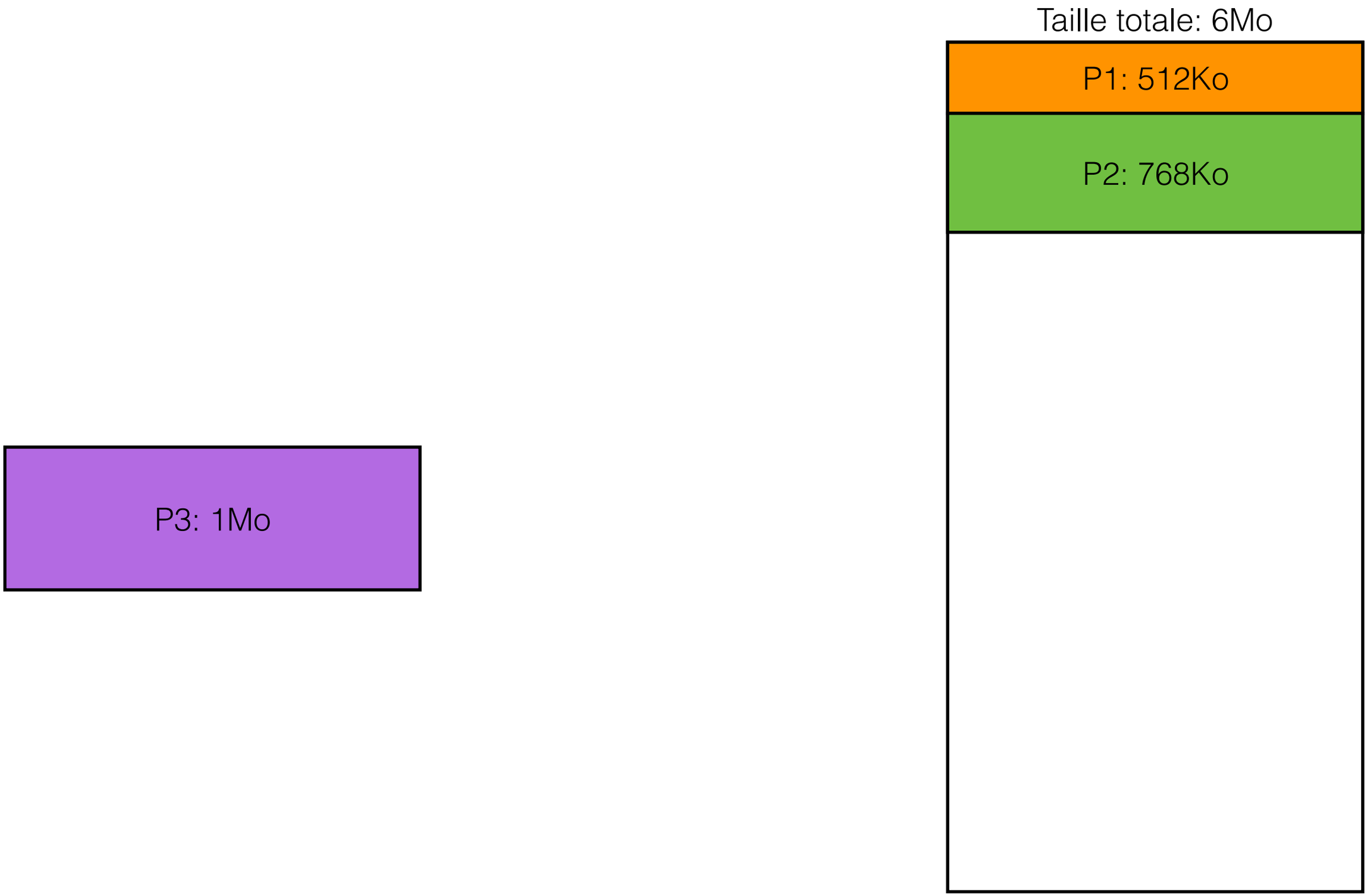
Allocation mémoire contigüe, taille **variable**



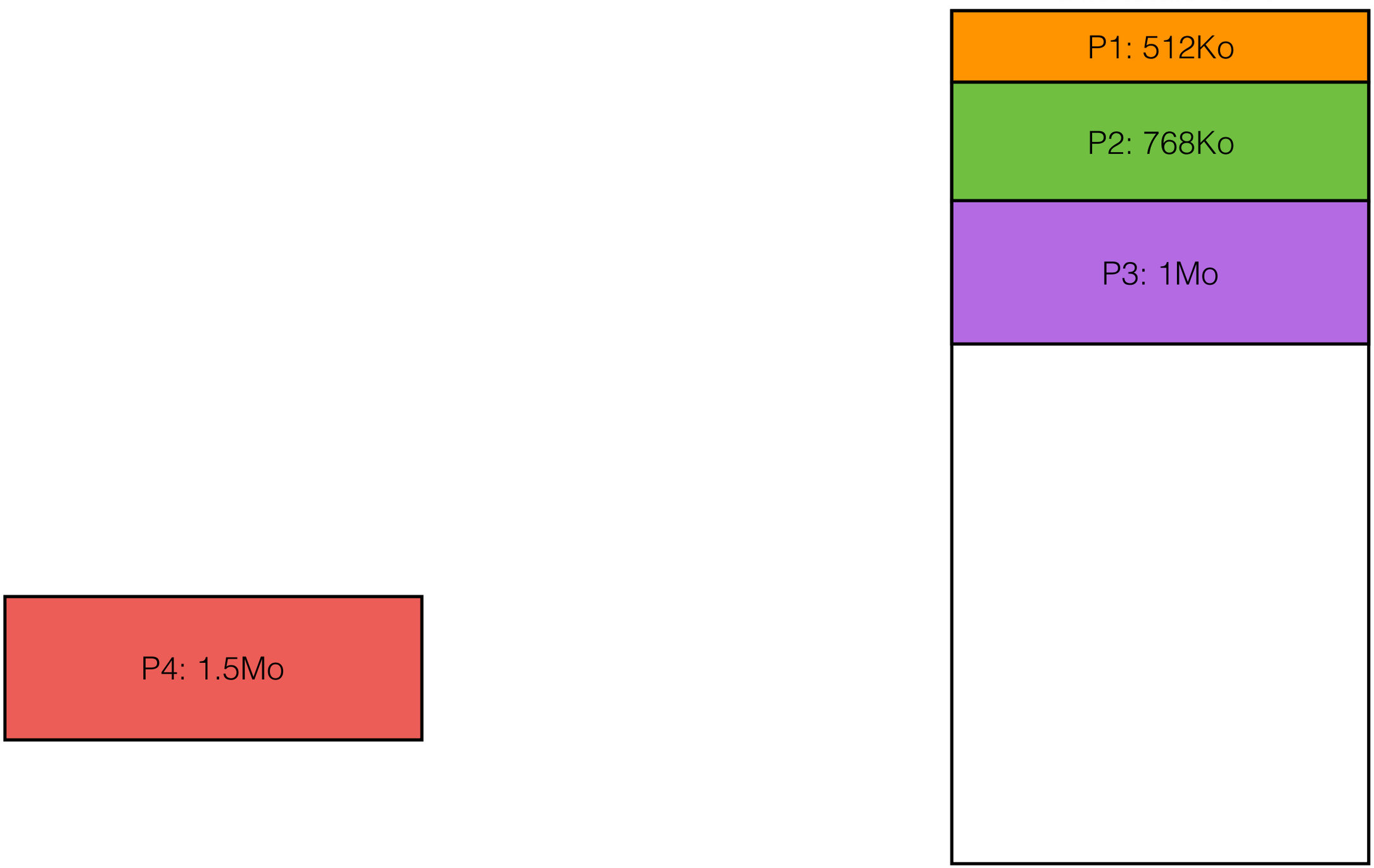
Allocation mémoire contigüe, taille **variable**



Allocation mémoire contigüe, taille **variable**

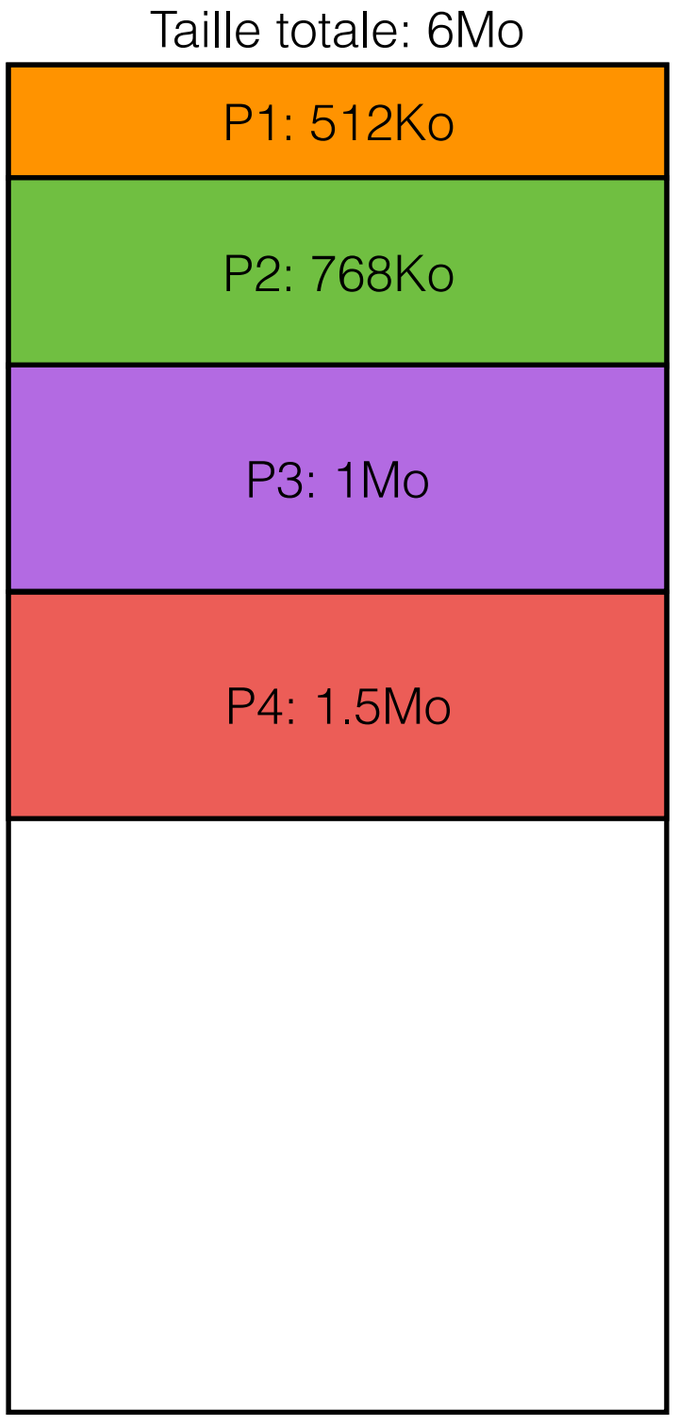


Allocation mémoire contigüe, taille **variable**



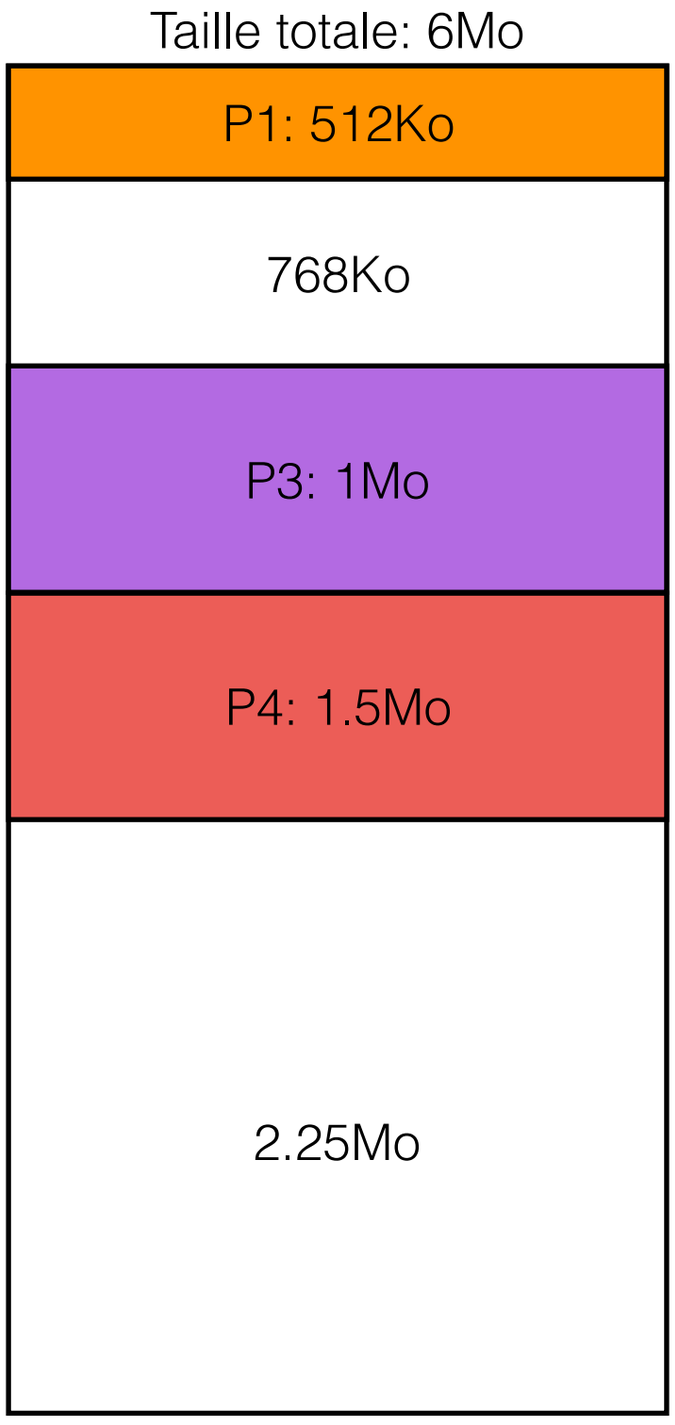
Allocation mémoire contigüe, taille **variable**

Qu’arrive-t-il si un processus (ex: P2) se termine?



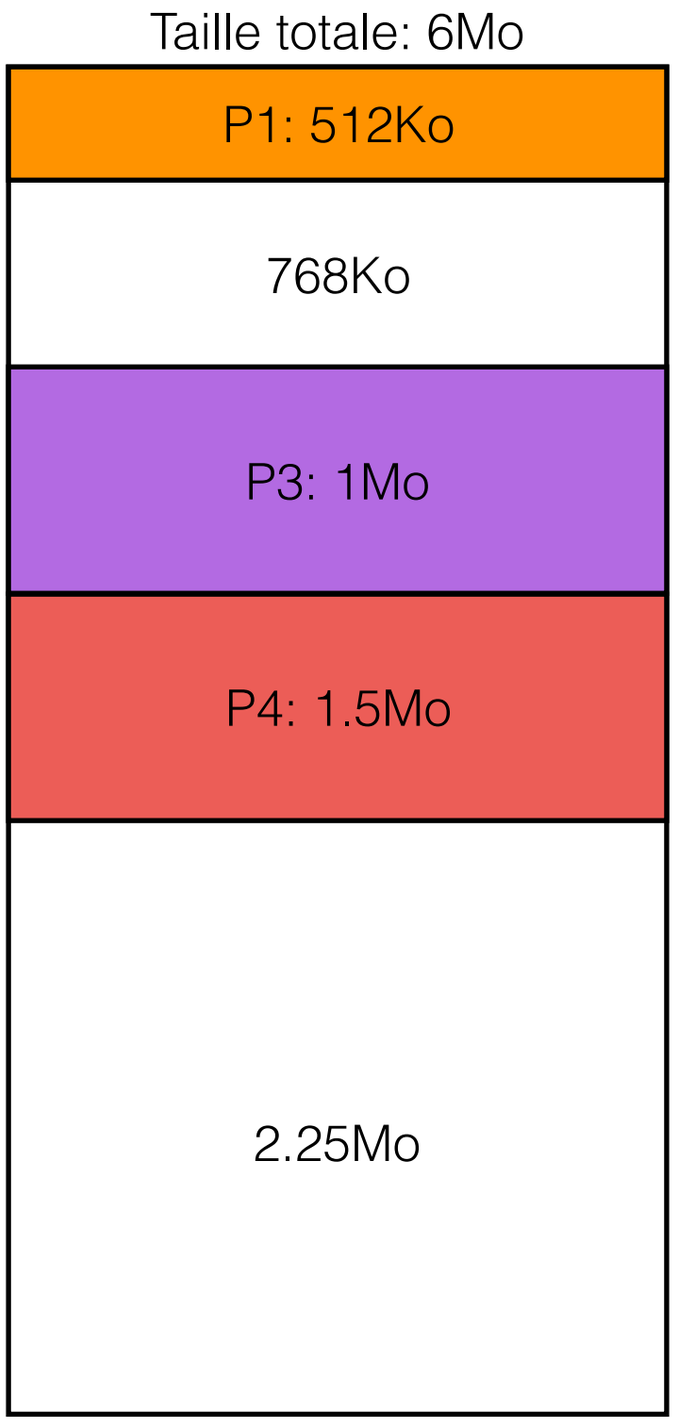
Allocation mémoire contigüe, taille **variable**

Cela crée un trou!



Allocation mémoire contigüe, taille **variable**

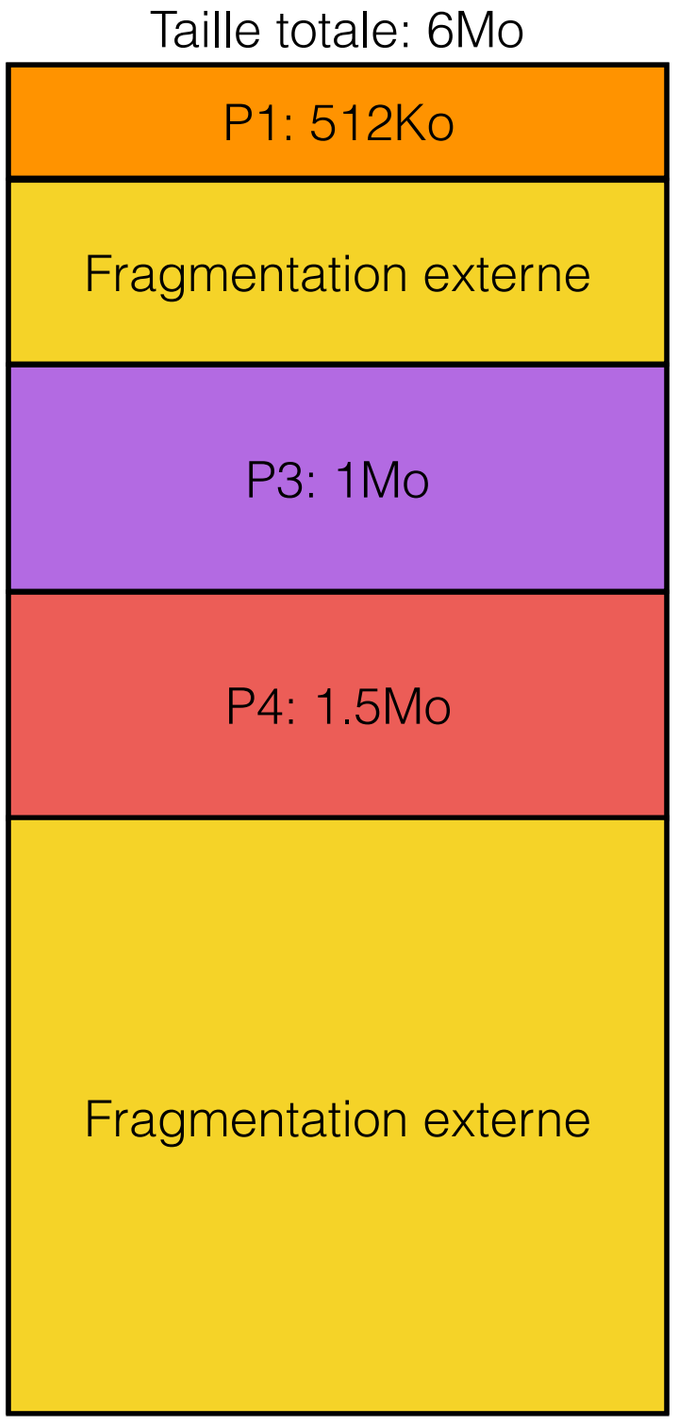
- Quels sont les problèmes avec l'allocation contigüe avec partitions de taille variable?
- Lorsqu'un processus se termine, il peut laisser des «trous».



Fragmentation

- Il en existe 2 types:
  - Fragmentation **interne**: espace perdu **à l'intérieur** une partition
  - Fragmentation **externe**: espace perdu **à l'extérieur** d'une partition

- Avec des partitions de taille **variable**, seule la fragmentation **externe** est possible
  - aucun espace à l'intérieur d'une partition n'est perdu car chaque partition est créée en fonction de la taille de son processus



Algorithmes d'allocation de mémoire

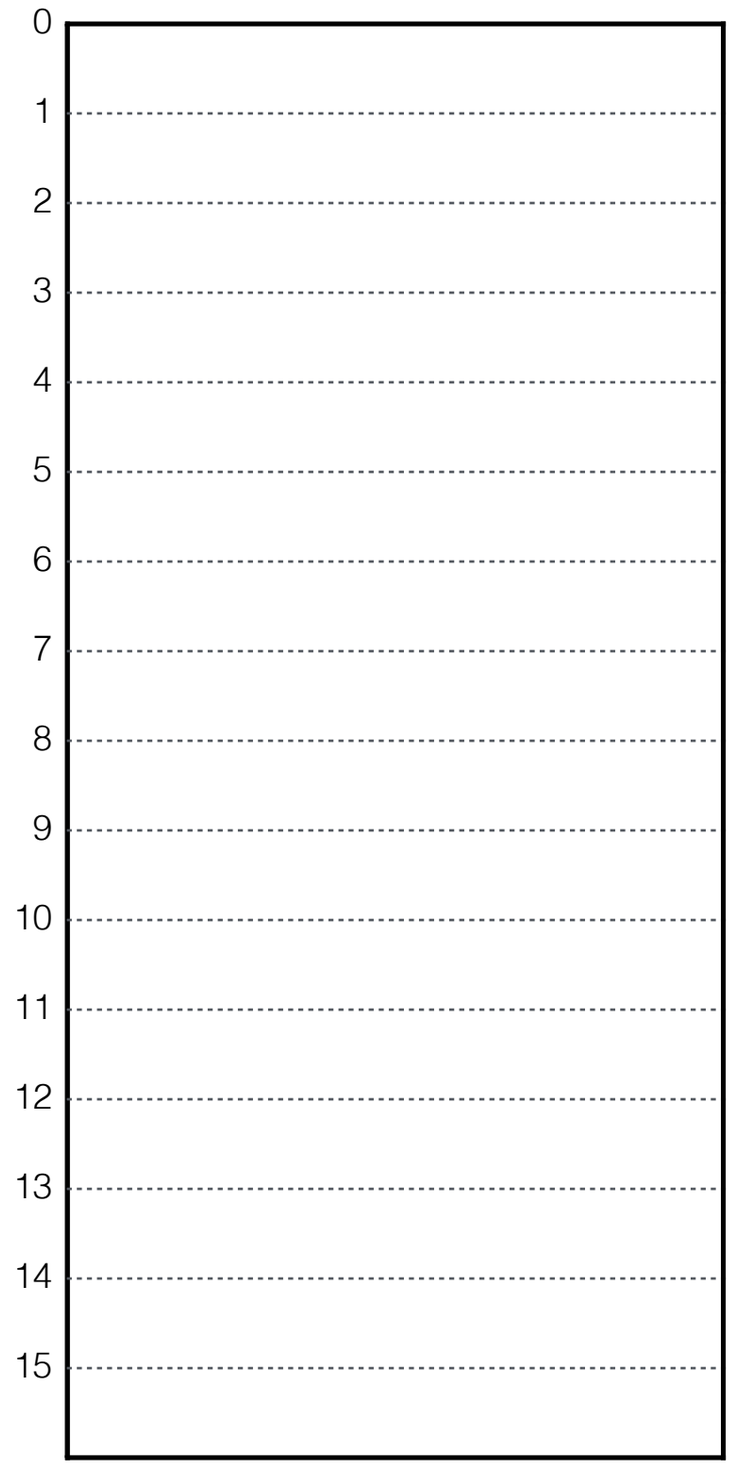
- Il existe plusieurs algorithmes afin de déterminer l'emplacement d'un processus en mémoire. Le but de tous ces algorithmes est de maximiser l'espace mémoire occupé.
  - **Première allocation (*First Fit*)**: Le processus est mis dans le premier bloc de mémoire suffisamment grand à partir du début de la mémoire.
    - Souvent utilisé malgré sa simplicité apparente!
  - **Prochaine allocation (*Next Fit*)**: Le processus est mis dans le premier bloc de mémoire suffisamment grand à partir du dernier bloc alloué.
    - Crée souvent un peu plus de fragmentation que « First Fit »
  - **Meilleure allocation (*Best Fit*)**: Le processus est mis dans le bloc de mémoire le plus petit dont la taille est suffisamment grande pour l'espace requis.
    - Semble meilleur, mais demande beaucoup de temps de calcul!
  - **Pire allocation (*Worse Fit*)**: Le processus est mis dans le bloc de mémoire le plus grand.

Exercice mémoire contigüe

- Effectuez les étapes suivantes pour un système ayant une mémoire de 16Mo:
  - Les processus suivants sont alloués en mémoire (dans l'ordre)
    - P1, 3Mo
    - P2, 5Mo
    - P3, 2Mo
    - P4, 2Mo
- P1 et P3 se terminent
- À quel endroit en mémoire le processus (P5, 2Mo) sera-t-il alloué si l'on emploie chacun des algorithmes d'allocation mémoire

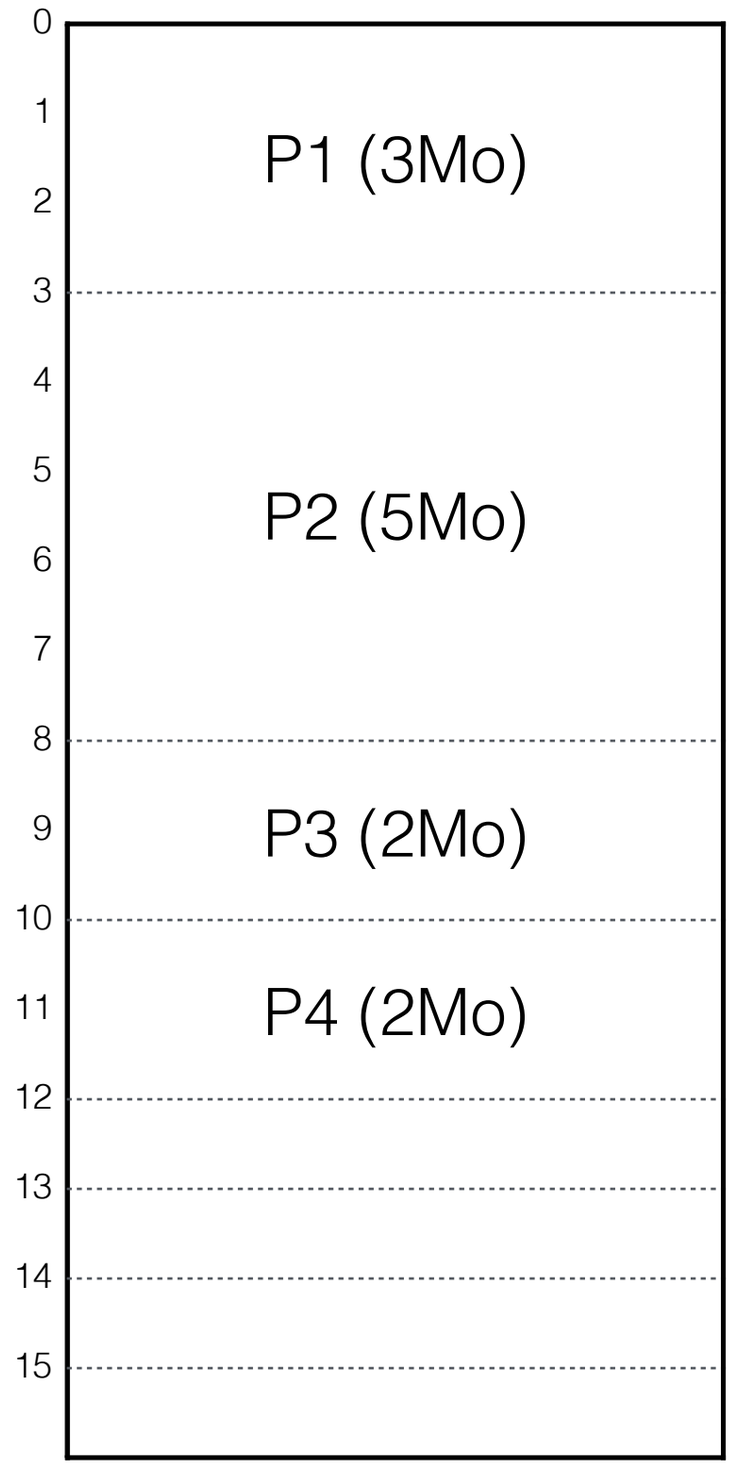
Exercice mémoire contigüe

- Effectuez les étapes suivantes pour un système ayant une mémoire de 16Mo:
  - Les processus suivants sont alloués en mémoire (dans l'ordre)
    - P1, 3Mo
    - P2, 5Mo
    - P3, 2Mo
    - P4, 2Mo
  - P1 et P3 se terminent
  - À quel endroit en mémoire le processus (P5, 2Mo) sera-t-il alloué si l'on emploie chacun des algorithmes d'allocation mémoire



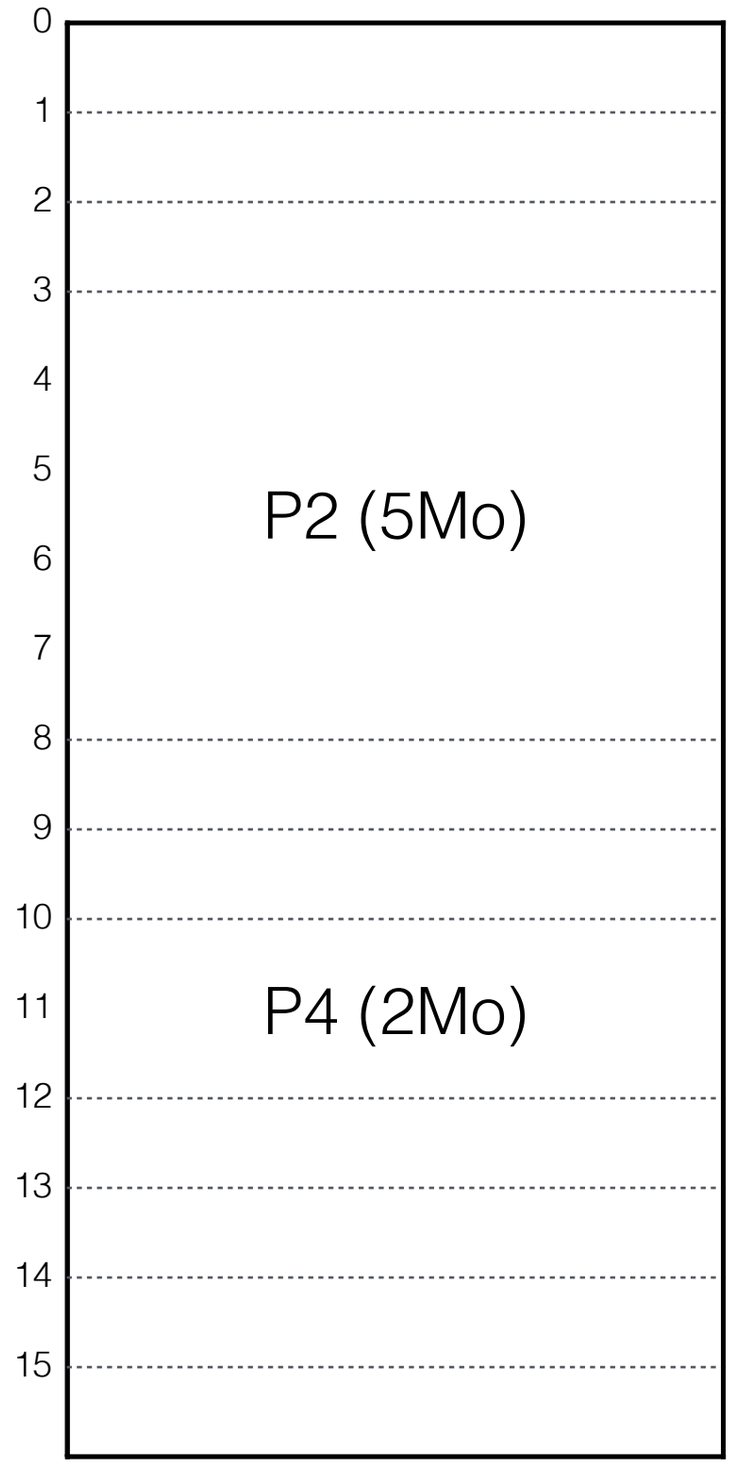
Exercice mémoire contigüe

- Effectuez les étapes suivantes pour un système ayant une mémoire de 16Mo:
  - Les processus suivants sont alloués en mémoire (dans l'ordre)
    - P1, 3Mo
    - P2, 5Mo
    - P3, 2Mo
    - P4, 2Mo
- P1 et P3 se terminent
- À quel endroit en mémoire le processus (P5, 2Mo) sera-t-il alloué si l'on emploie chacun des algorithmes d'allocation mémoire



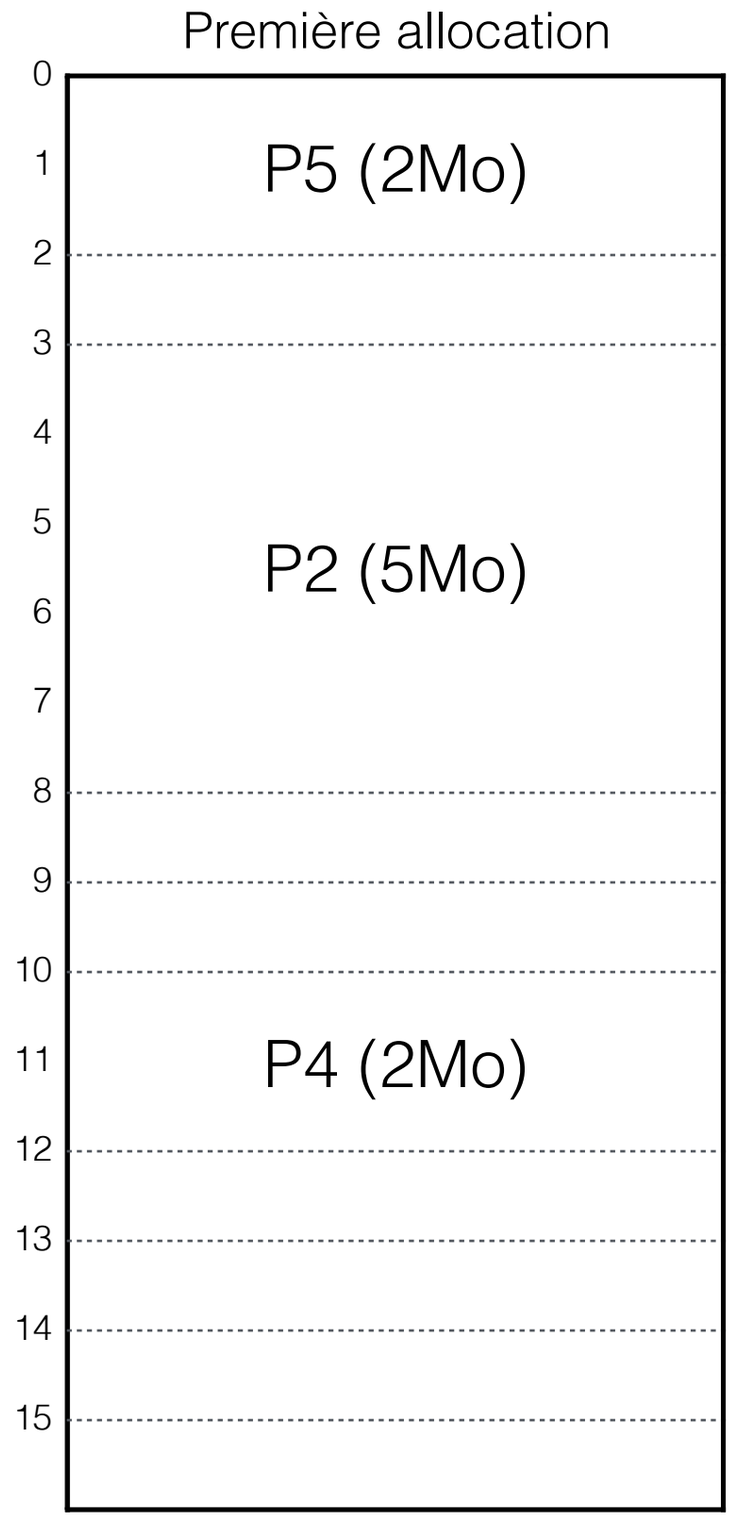
Exercice mémoire contigüe

- Effectuez les étapes suivantes pour un système ayant une mémoire de 16Mo:
  - Les processus suivants sont alloués en mémoire (dans l'ordre)
    - P1, 3Mo
    - P2, 5Mo
    - P3, 2Mo
    - P4, 2Mo
- P1 et P3 se terminent
- À quel endroit en mémoire le processus (P5, 2Mo) sera-t-il alloué si l'on emploie chacun des algorithmes d'allocation mémoire



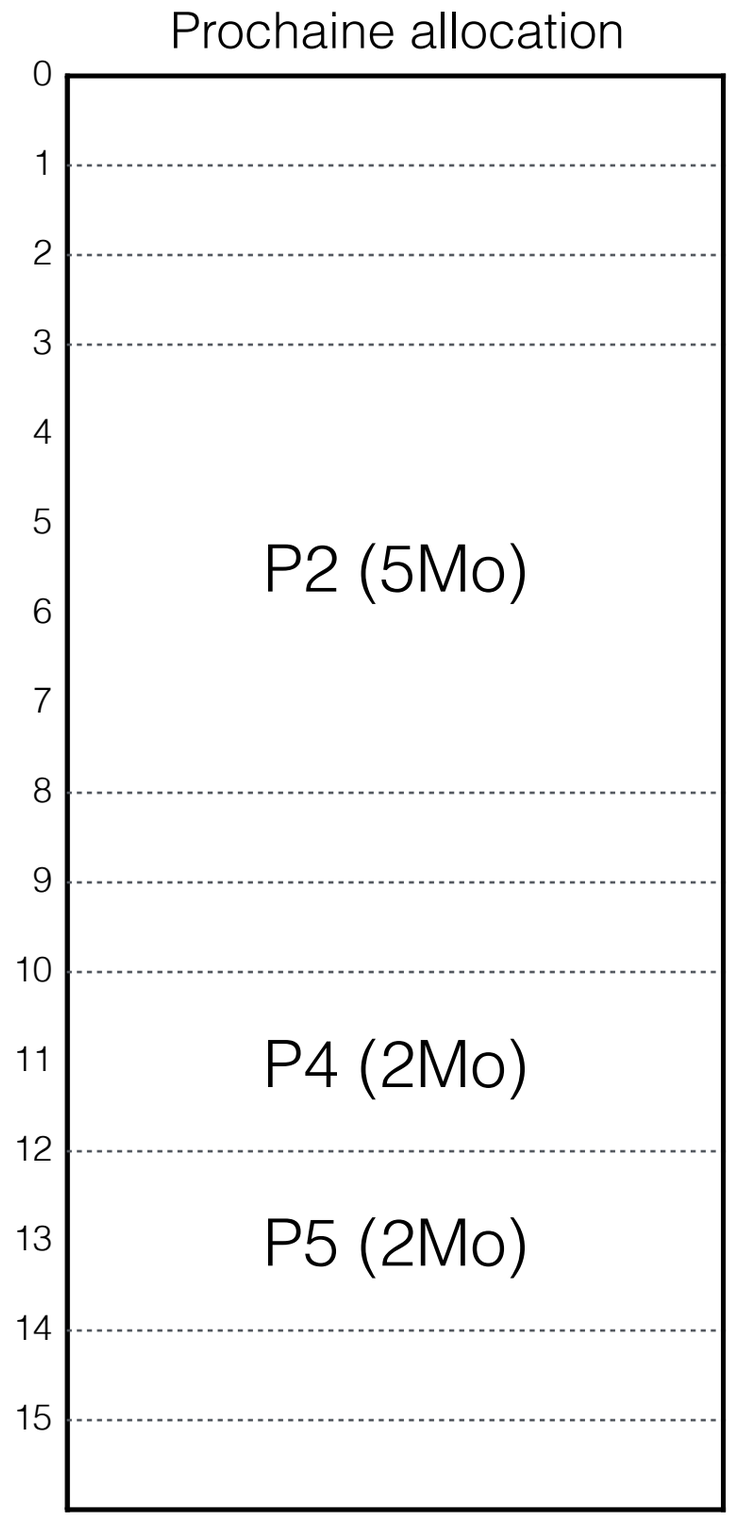
Exercice mémoire contigüe

- Effectuez les étapes suivantes pour un système ayant une mémoire de 16Mo:
  - Les processus suivants sont alloués en mémoire (dans l'ordre)
    - P1, 3Mo
    - P2, 5Mo
    - P3, 2Mo
    - P4, 2Mo
- P1 et P3 se terminent
- À quel endroit en mémoire le processus (P5, 2Mo) sera-t-il alloué si l'on emploie chacun des algorithmes d'allocation mémoire



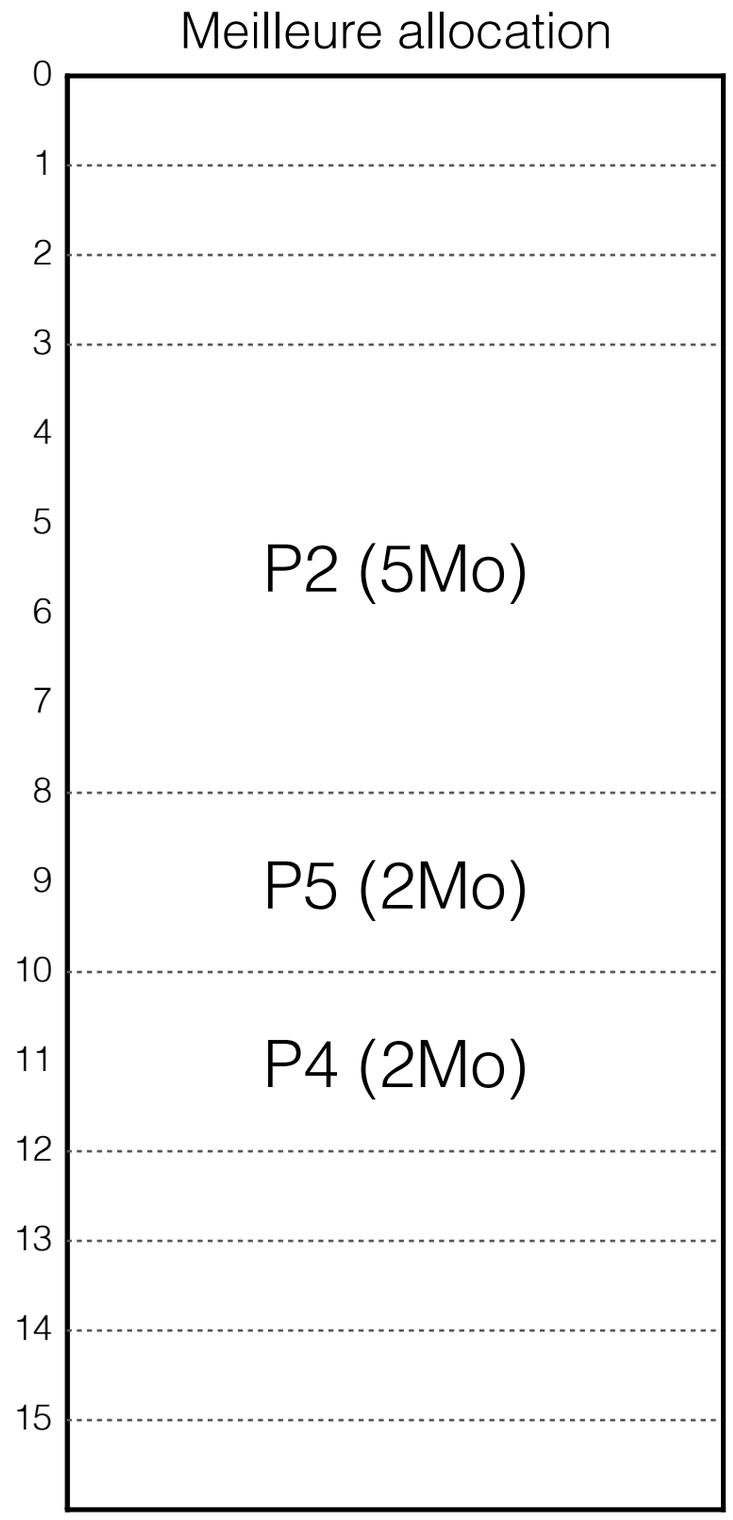
Exercice mémoire contigüe

- Effectuez les étapes suivantes pour un système ayant une mémoire de 16Mo:
  - Les processus suivants sont alloués en mémoire (dans l'ordre)
    - P1, 3Mo
    - P2, 5Mo
    - P3, 2Mo
    - P4, 2Mo
- P1 et P3 se terminent
- À quel endroit en mémoire le processus (P5, 2Mo) sera-t-il alloué si l'on emploie chacun des algorithmes d'allocation mémoire



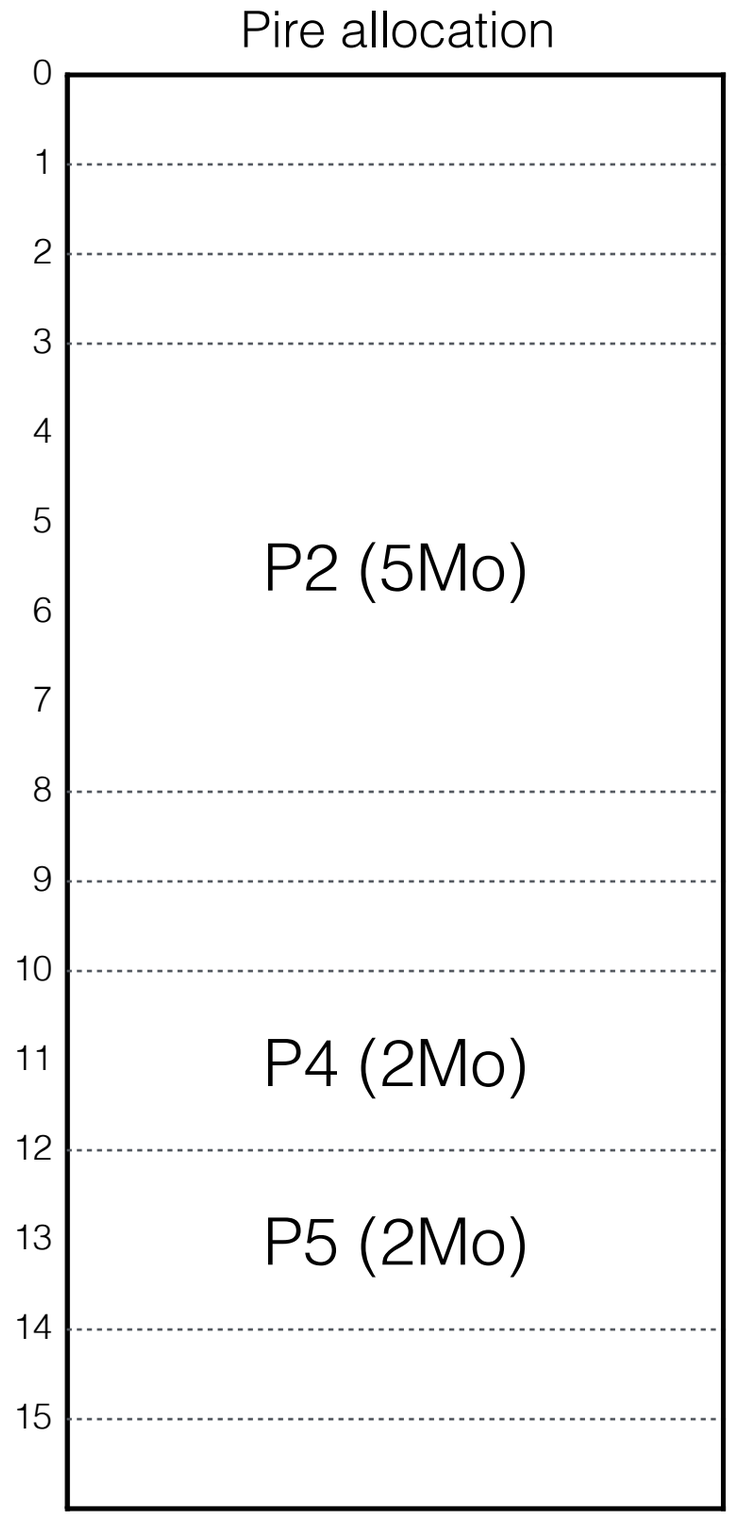
Exercice mémoire contigüe

- Effectuez les étapes suivantes pour un système ayant une mémoire de 16Mo:
  - Les processus suivants sont alloués en mémoire (dans l'ordre)
    - P1, 3Mo
    - P2, 5Mo
    - P3, 2Mo
    - P4, 2Mo
- P1 et P3 se terminent
- À quel endroit en mémoire le processus (P5, 2Mo) sera-t-il alloué si l'on emploie chacun des algorithmes d'allocation mémoire



Exercice mémoire contigüe

- Effectuez les étapes suivantes pour un système ayant une mémoire de 16Mo:
  - Les processus suivants sont alloués en mémoire (dans l'ordre)
    - P1, 3Mo
    - P2, 5Mo
    - P3, 2Mo
    - P4, 2Mo
- P1 et P3 se terminent
- À quel endroit en mémoire le processus (P5, 2Mo) sera-t-il alloué si l'on emploie chacun des algorithmes d'allocation mémoire



Exercice mémoire contigüe

- Un système possédant une mémoire de 16Mo doit allouer les processus suivants:

Processus	Taille	Temps de création	Durée
P1	3Mo	0	3
P2	2Mo	1	3
P3	4Mo	2	3
P4	2Mo	3	3
P5	3Mo	4	3

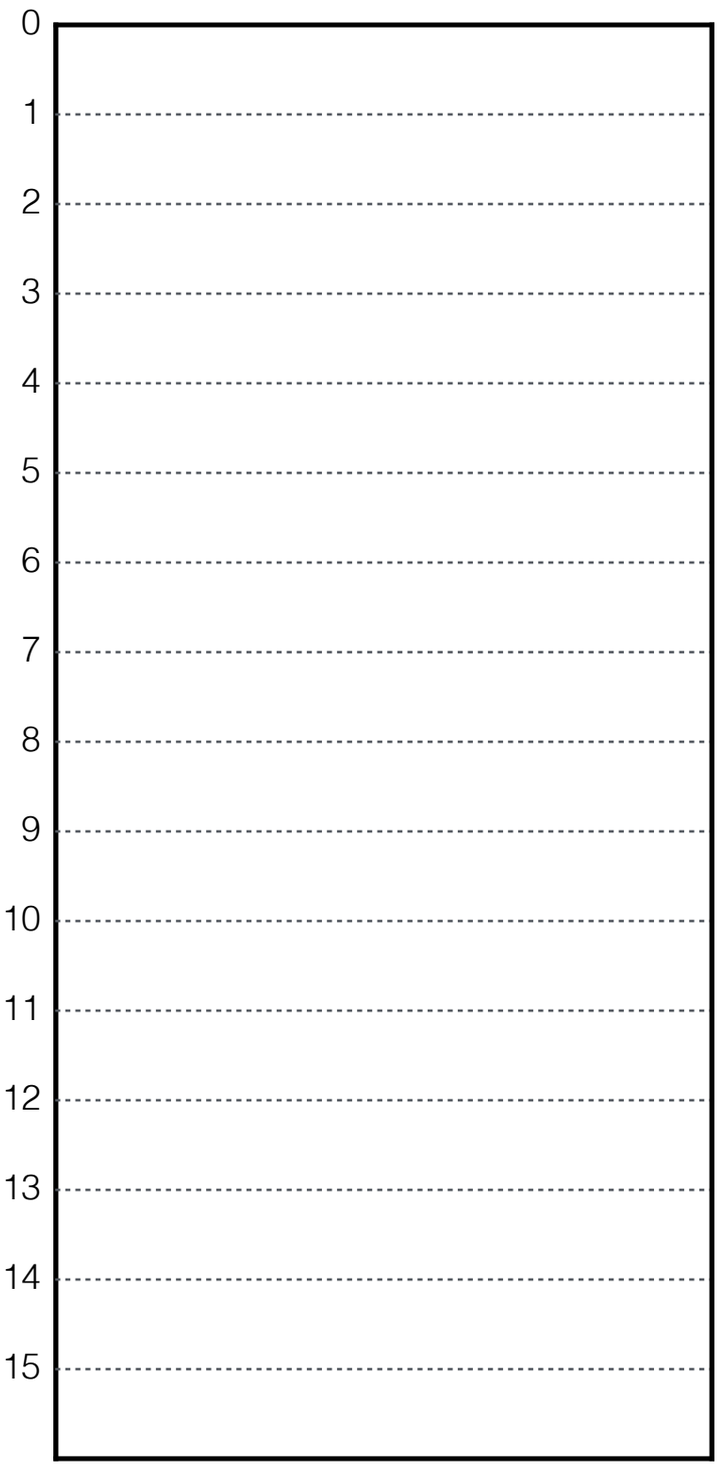
- Indiquez le contenu de la mémoire après l'allocation du processus P5 en utilisant l'algorithme de la **meilleure allocation**

Exercice mémoire contigüe

Processus	Taille	Temps de création	Durée
P1	3Mo	0	3
P2	2Mo	1	3
P3	4Mo	2	3
P4	2Mo	3	3
P5	3Mo	4	3

Temps:

Processus



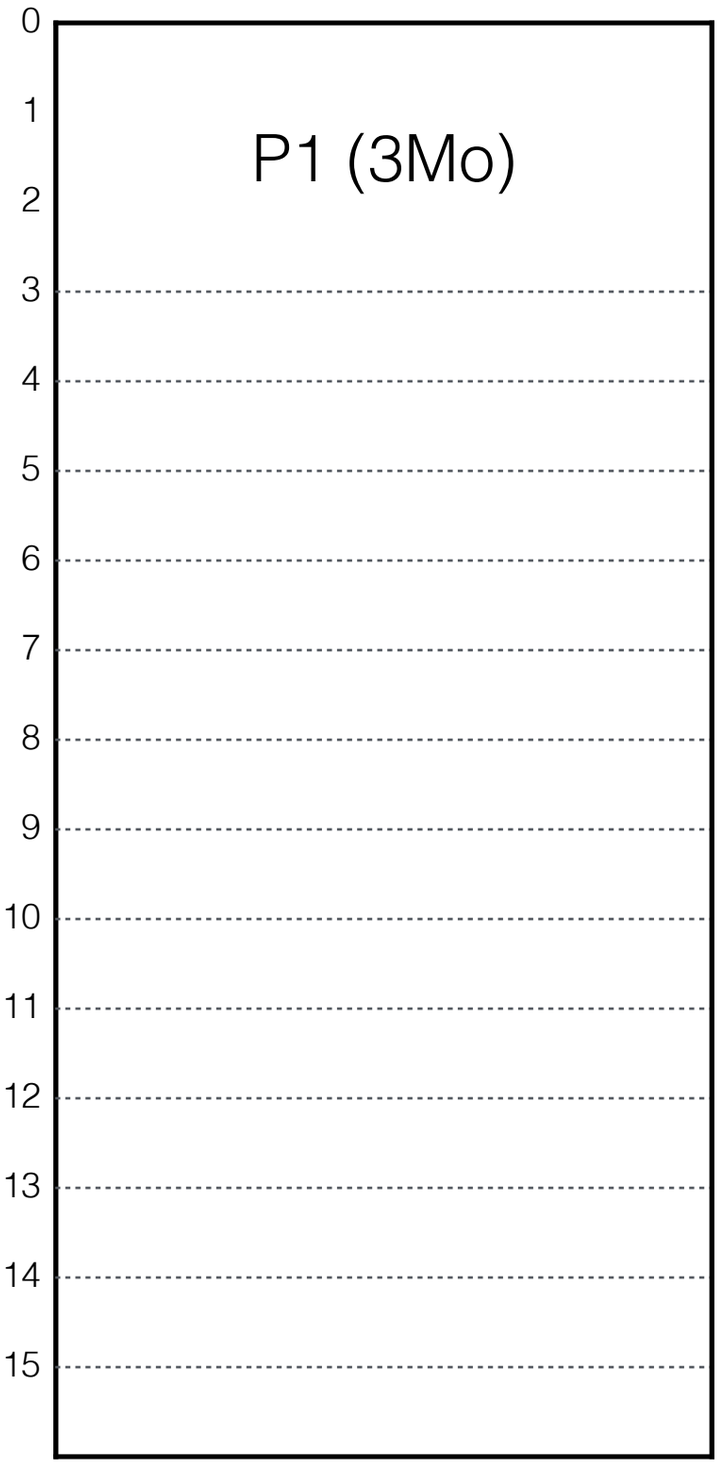
Exercice mémoire contigüe

Processus	Taille	Temps de création	Durée
P1	3Mo	0	3
P2	2Mo	1	3
P3	4Mo	2	3
P4	2Mo	3	3
P5	3Mo	4	3

Temps: 0

Processus

P1: 3



Exercice mémoire contigüe

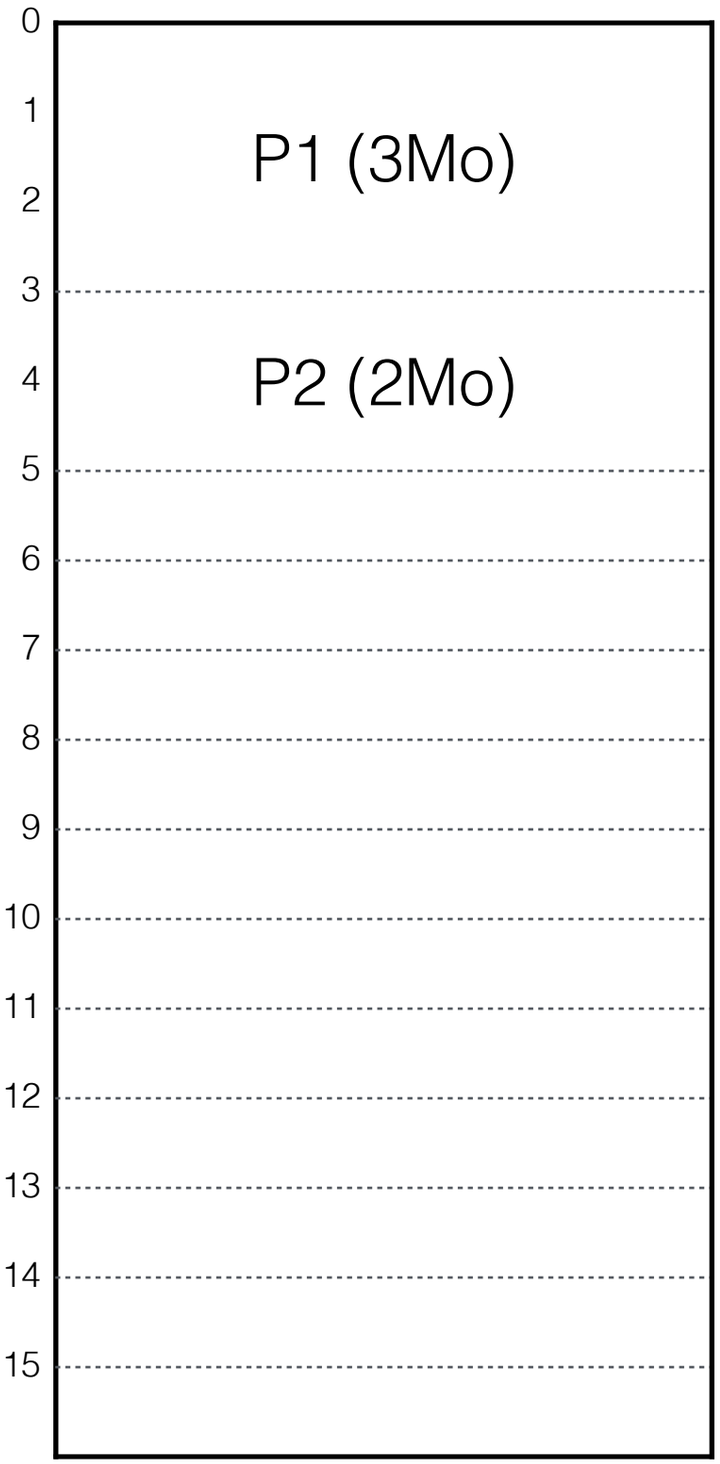
Processus	Taille	Temps de création	Durée
P1	3Mo	0	3
P2	2Mo	1	3
P3	4Mo	2	3
P4	2Mo	3	3
P5	3Mo	4	3

Temps: 1

Processus

P1: 2

P2: 3



Exercice mémoire contigüe

Processus	Taille	Temps de création	Durée
P1	3Mo	0	3
P2	2Mo	1	3
P3	4Mo	2	3
P4	2Mo	3	3
P5	3Mo	4	3

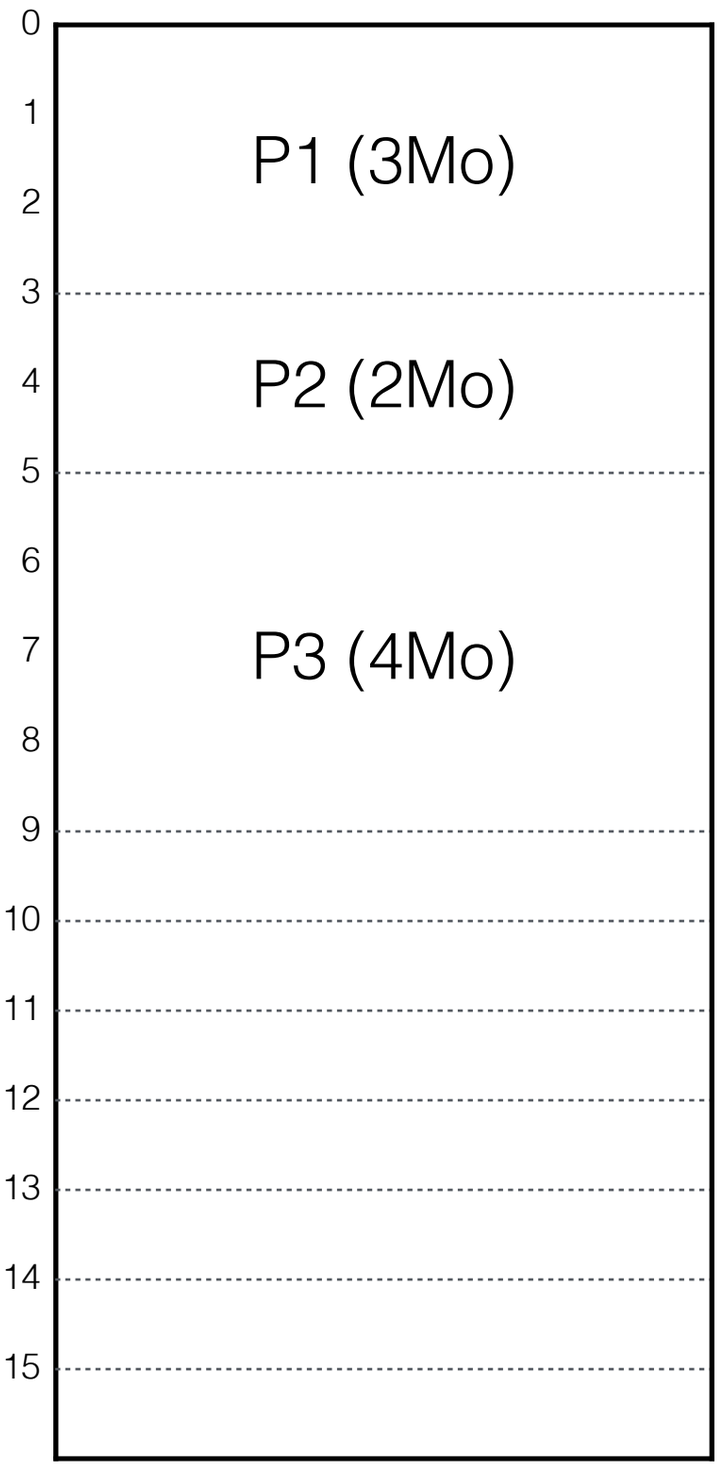
Temps: 2

Processus

P1: 1

P2: 2

P3: 3



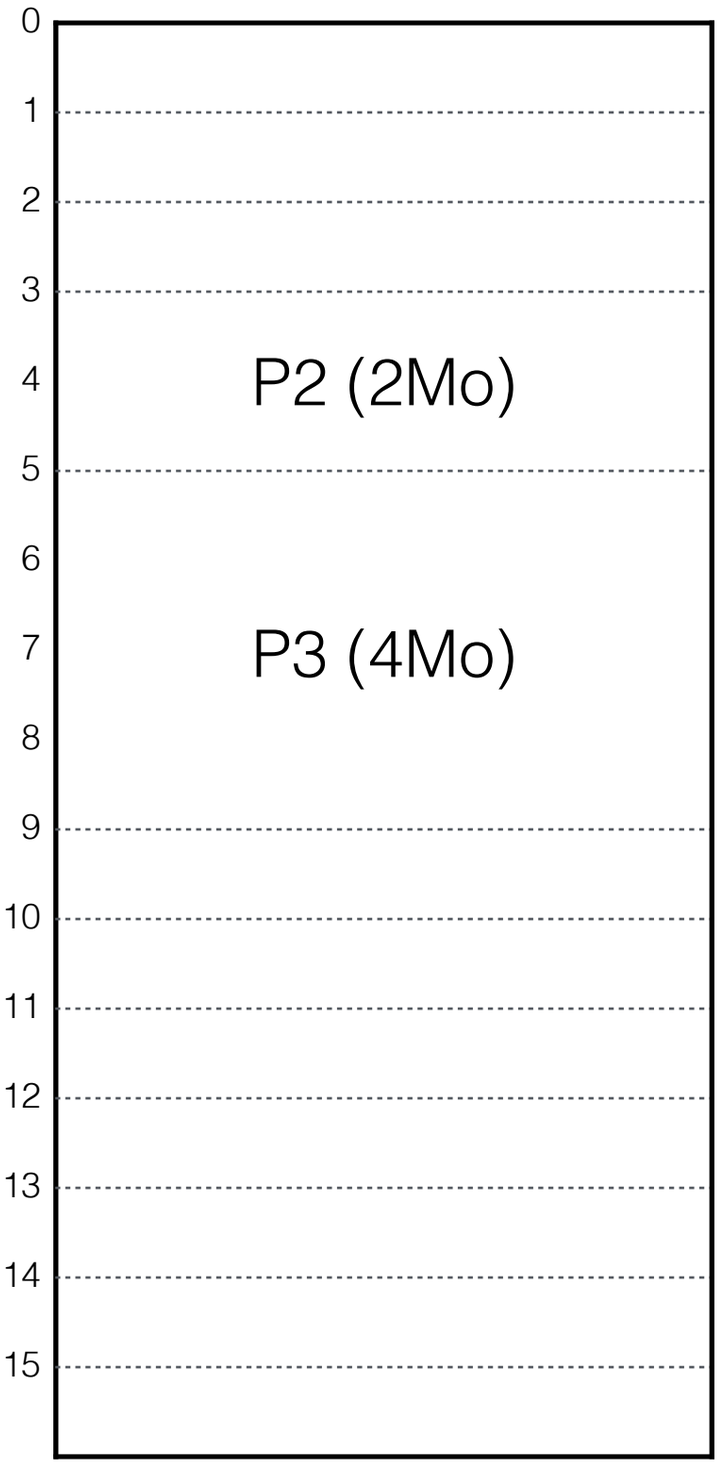
Exercice mémoire contigüe

Processus	Taille	Temps de création	Durée
P1	3Mo	0	3
P2	2Mo	1	3
P3	4Mo	2	3
P4	2Mo	3	3
P5	3Mo	4	3

Temps: 3

Processus

- ~~P1: 0~~
- P2: 1
- P3: 2



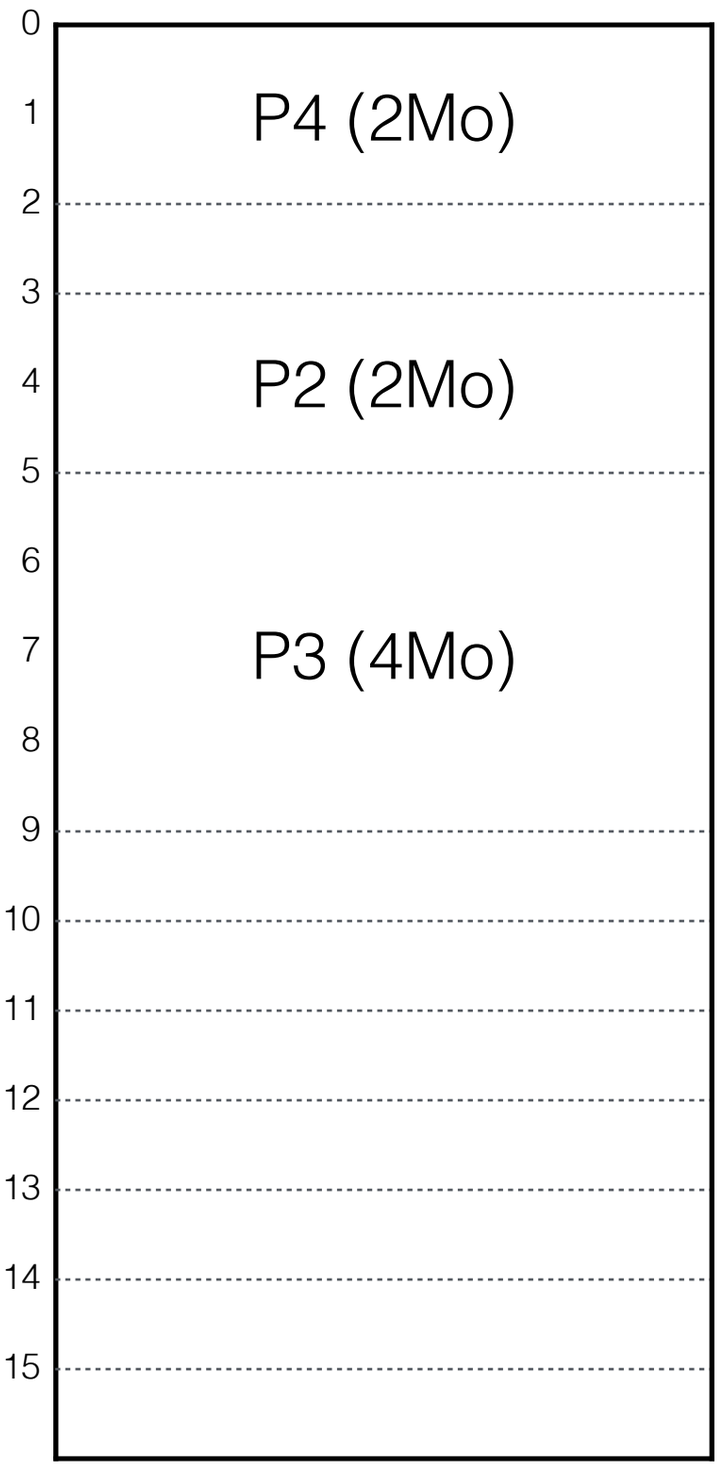
Exercice mémoire contigüe

Processus	Taille	Temps de création	Durée
P1	3Mo	0	3
P2	2Mo	1	3
P3	4Mo	2	3
P4	2Mo	3	3
P5	3Mo	4	3

Temps: 3

Processus

- ~~P1: 0~~
- P2: 1
- P3: 2
- P4: 3



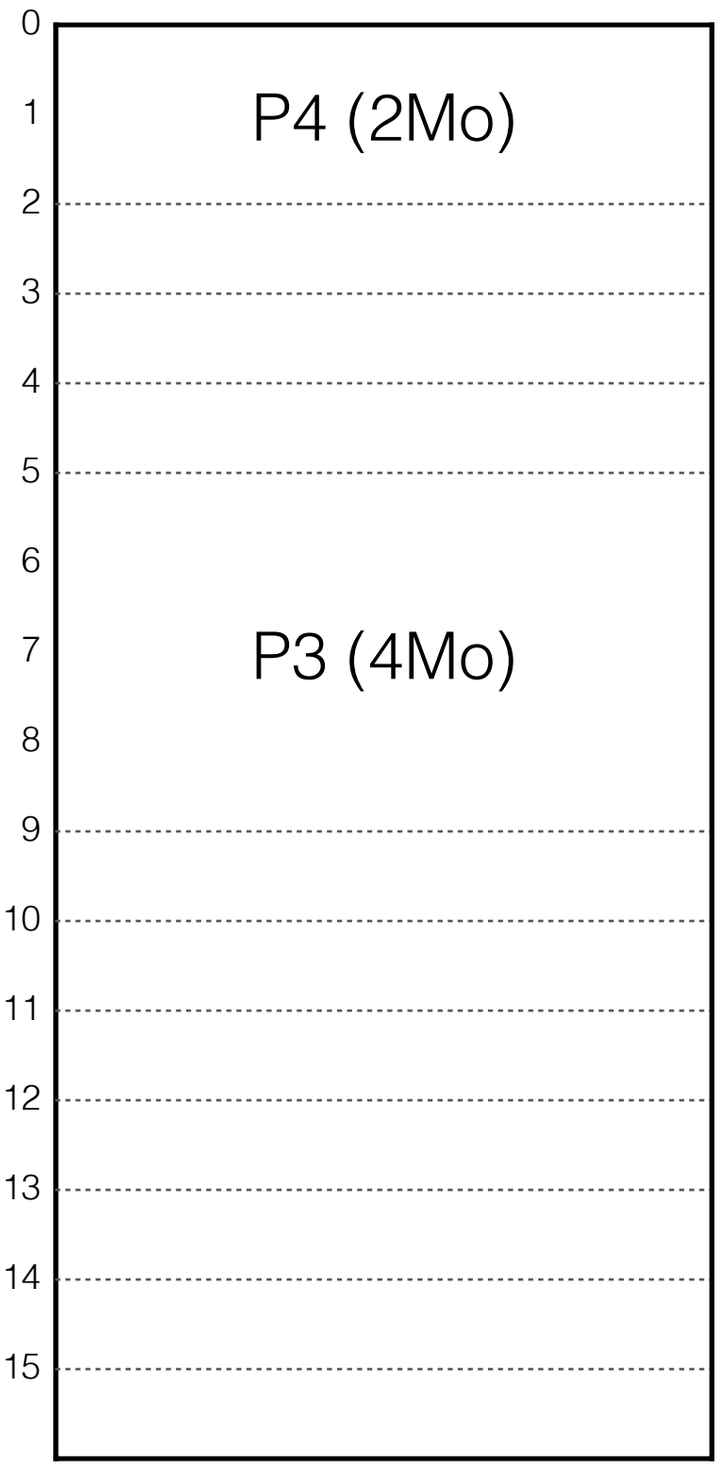
Exercice mémoire contigüe

Processus	Taille	Temps de création	Durée
P1	3Mo	0	3
P2	2Mo	1	3
P3	4Mo	2	3
P4	2Mo	3	3
P5	3Mo	4	3

Temps: 4

Processus

- ~~P1: 0~~
- ~~P2: 0~~
- P3: 1
- P4: 2



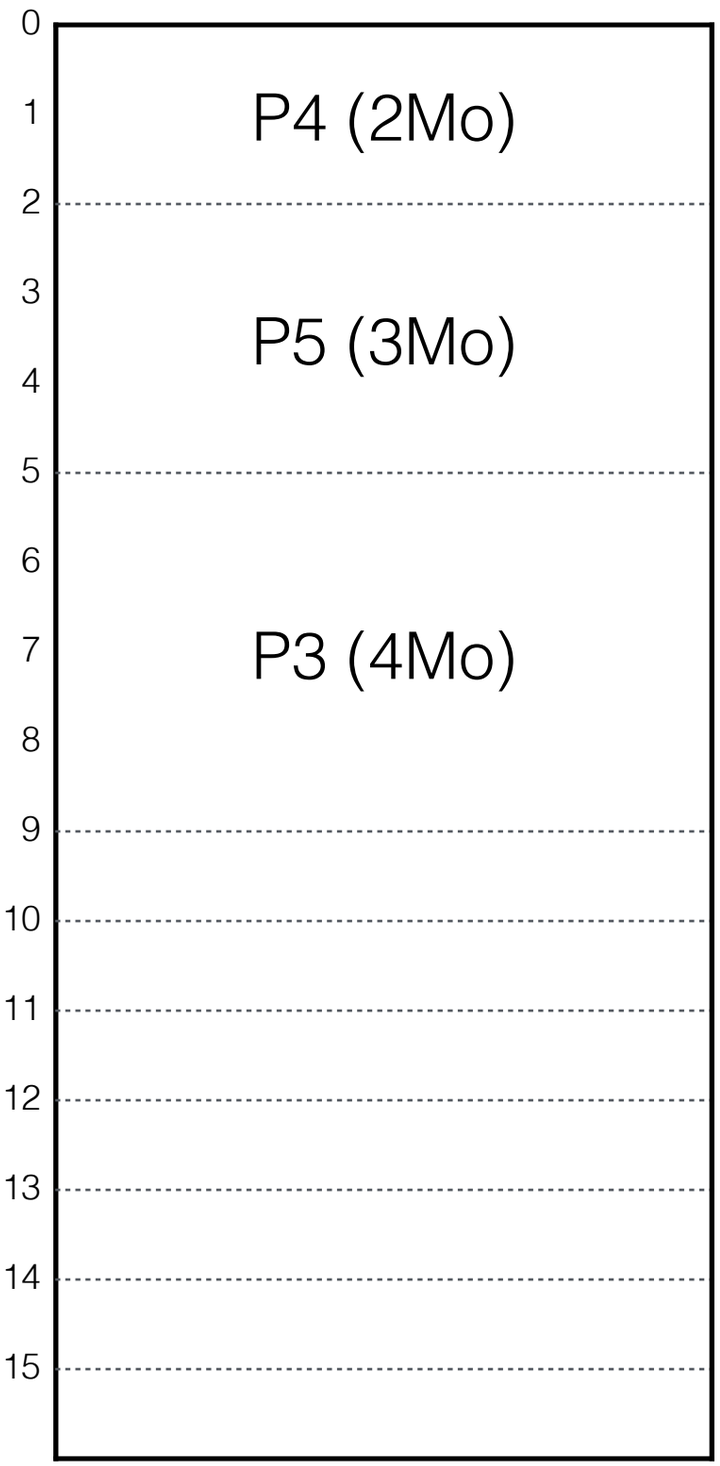
Exercice mémoire contigüe

Processus	Taille	Temps de création	Durée
P1	3Mo	0	3
P2	2Mo	1	3
P3	4Mo	2	3
P4	2Mo	3	3
P5	3Mo	4	3

Temps: 4

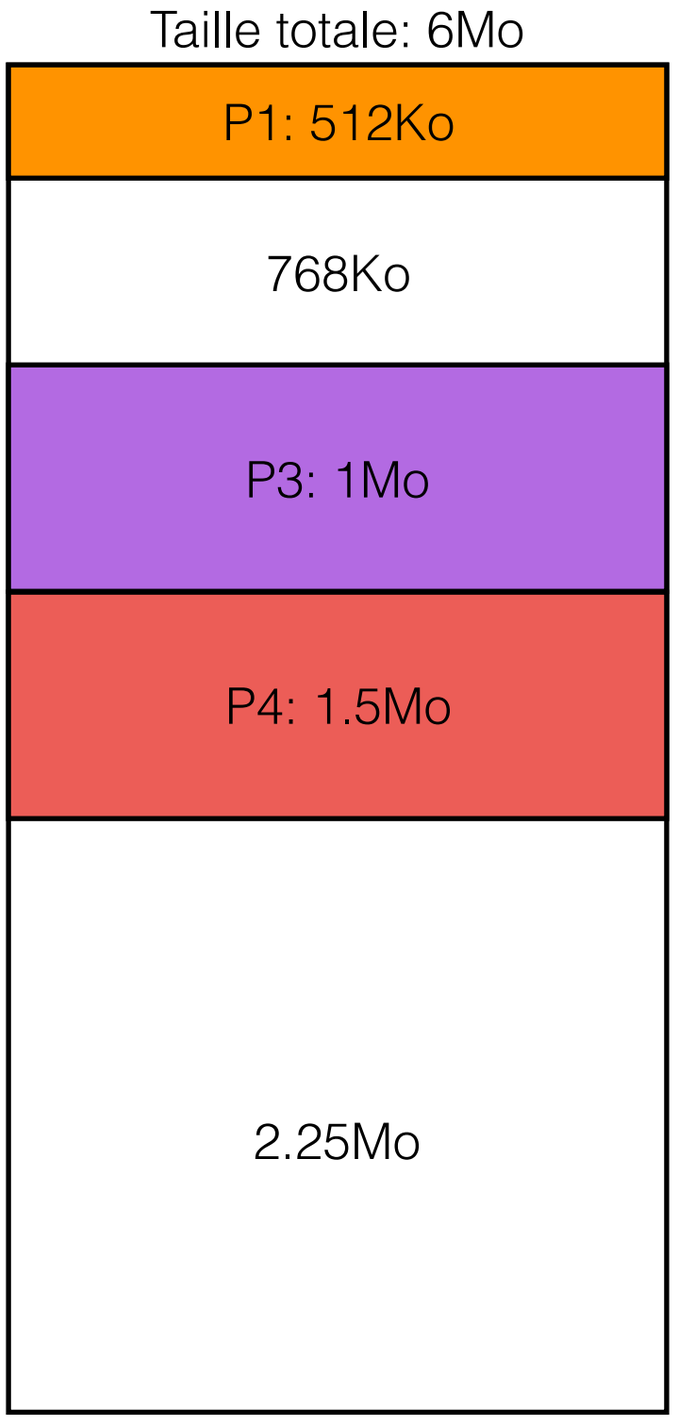
Processus

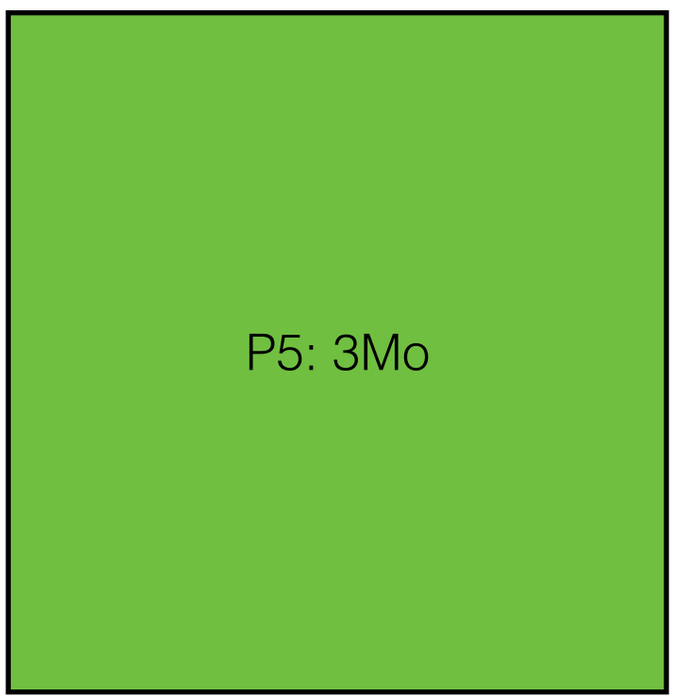
- ~~P1: 0~~
- ~~P2: 0~~
- P3: 1
- P4: 2
- P5: 3



Allocation contigüe, taille variable

- Quels sont les problèmes avec l'allocation contigüe avec partitions de taille variable?
  - Lorsqu'un processus se termine, il peut laisser des «trous».
  - Que faire si aucun «trou» n'est assez grand pour un processus?



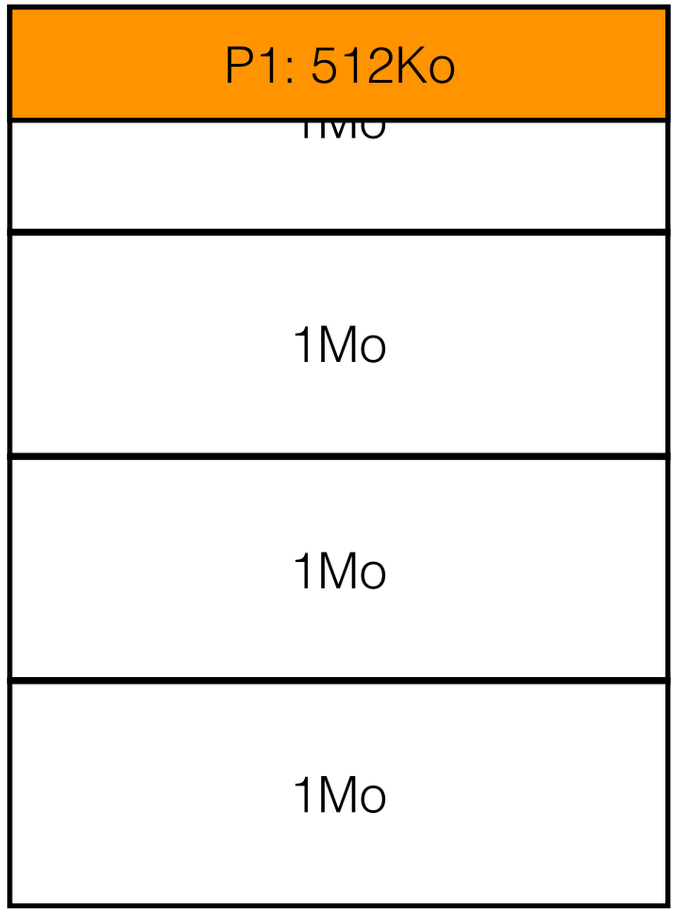


Ne peut être admis en mémoire!
(même si l'espace total libre est suffisant: 2.25Mo+768Ko = 3Mo)

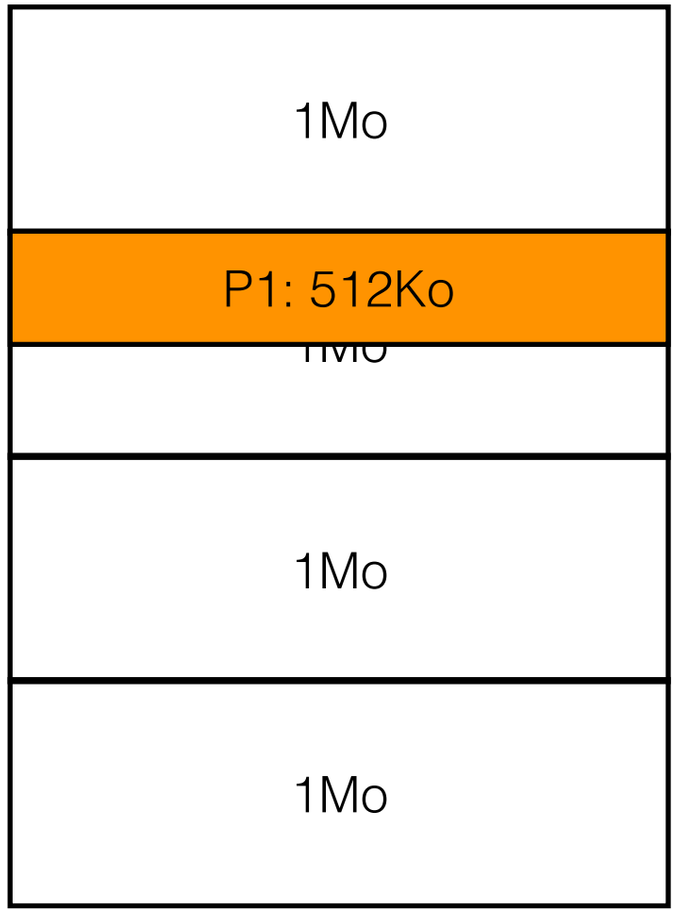
Adresses **virtuelles** et adresses **physiques**

- Est-ce qu'un processus doit absolument savoir à quel endroit il est placé en mémoire?

- Pour P1, est-ce que d'être placé dans le premier ou le second bloc fait une différence?



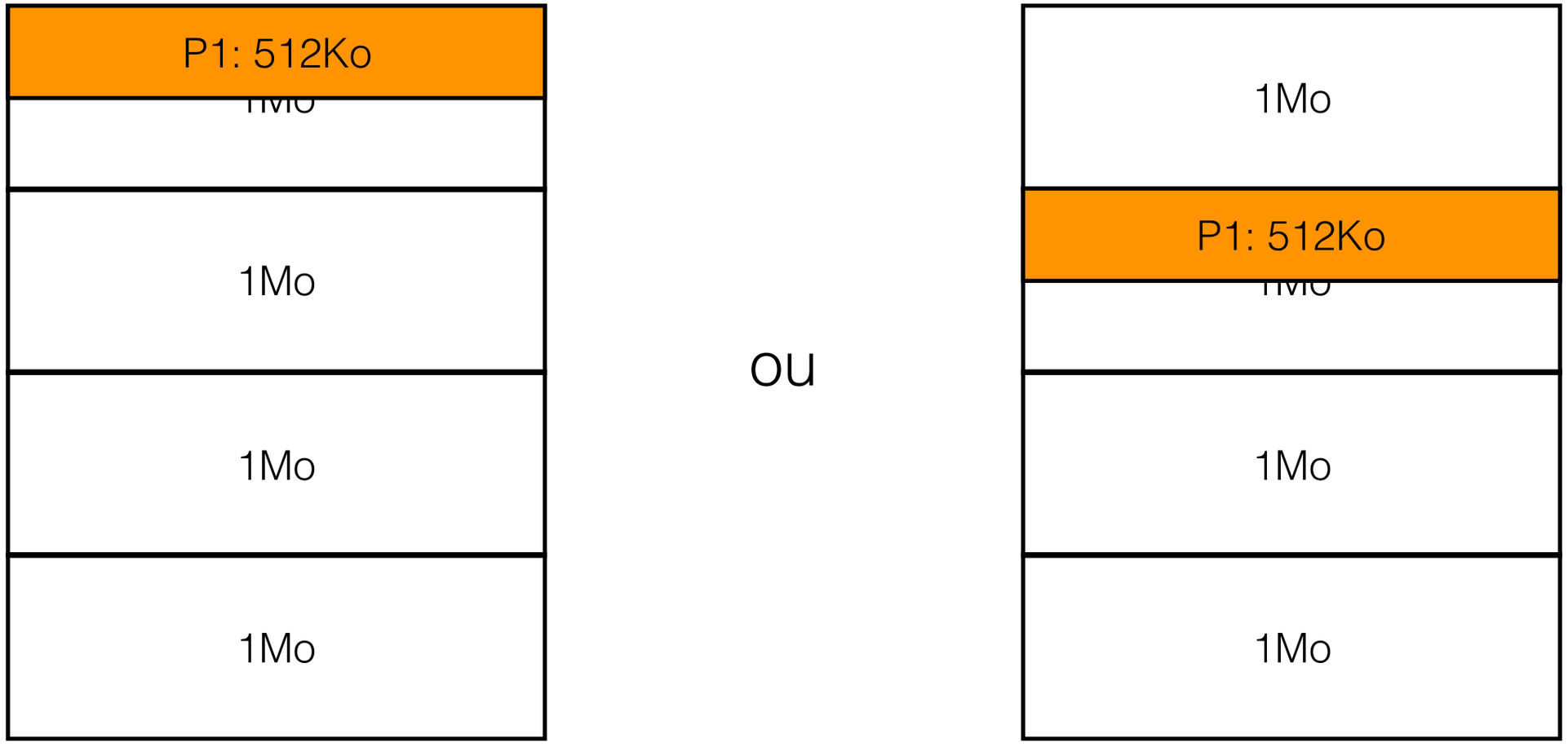
ou



Adresses **virtuelles** et adresses **physiques**

- Aucune différence!

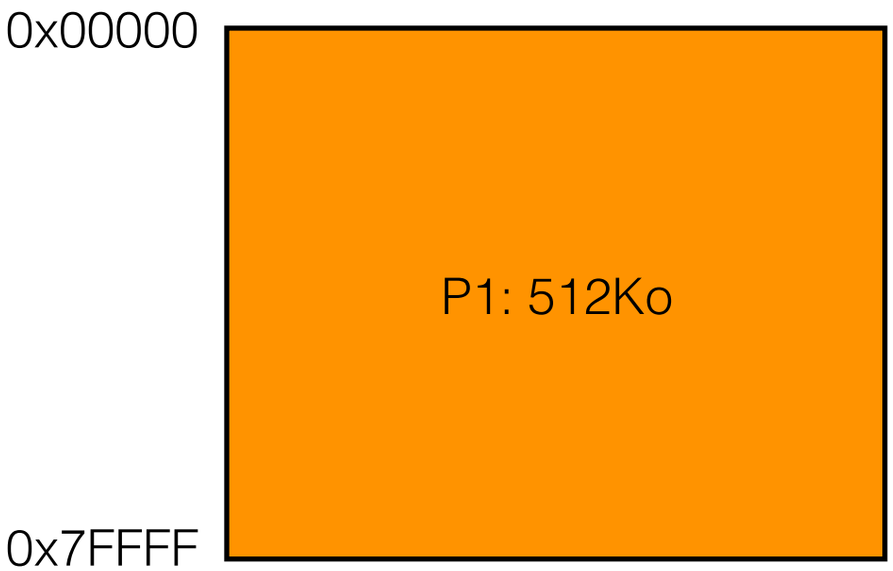
- Tout ce que le processus doit savoir, c'est comment accéder à chacune des adresses dans son bloc de mémoire.



Adresses **virtuelles** et adresses **physiques**

Adresses **virtuelles**

du point de vue du **processus**



Adresses **physiques**

du point de vue de la **mémoire**

Adresses **virtuelles** et adresses **physiques**

- Un processus utilise des adresses **virtuelles**
- La mémoire utilise des adresses **physiques**
- Il faut donc «traduire» entre les deux: c'est le travail du ***Memory Management Unit*** (MMU)

Allocation contigüe (partitions fixes): MMU

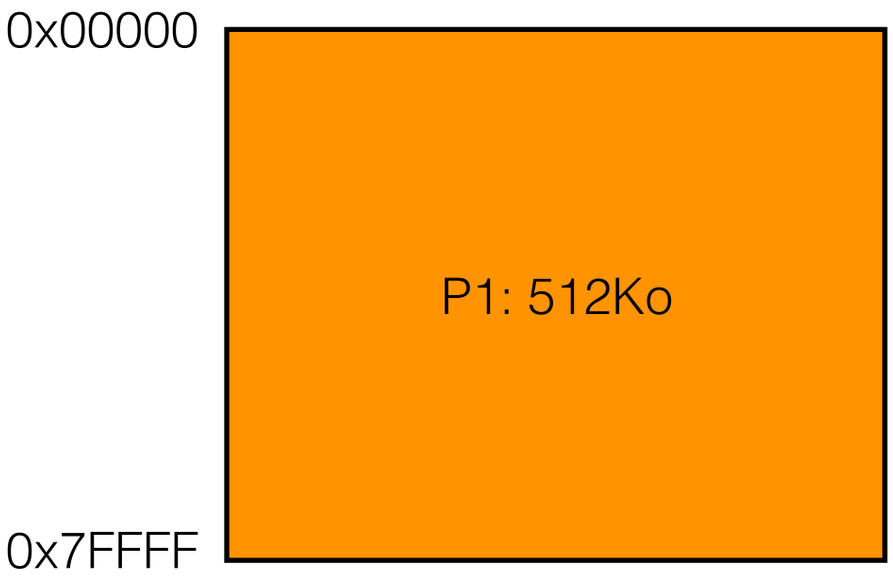
- En allocation contigüe avec partitions de taille fixe, le **MMU** effectue le calcul suivant pour traduire une adresse **virtuelle** en adresse **physique**:

$$a_{physique} = a_{virtuelle} + a_{partition}$$

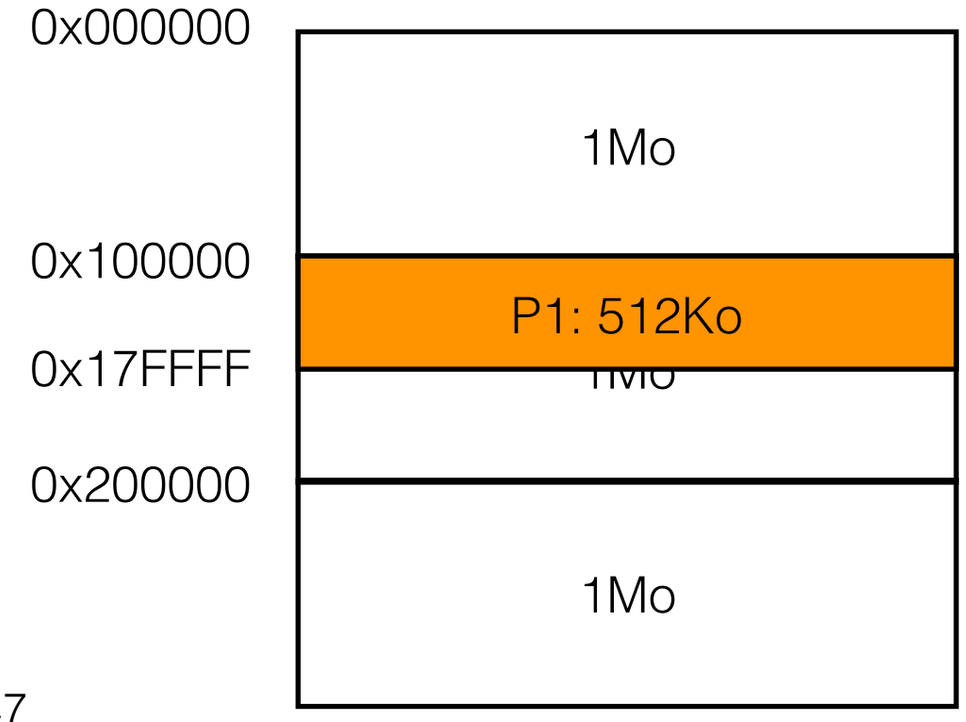
$a_{partition}$ est la première adresse de la partition attribuée au processus

- Par exemple, l'adresse virtuelle 0x00AB3 correspond à l'adresse physique 0x100AB3 dans l'exemple ci-bas.

Adresses **virtuelles**
(du point de vue du **processus**)



Adresses **physiques**
(du point de vue de la **mémoire**)



Allocation contigüe (partitions variables): MMU

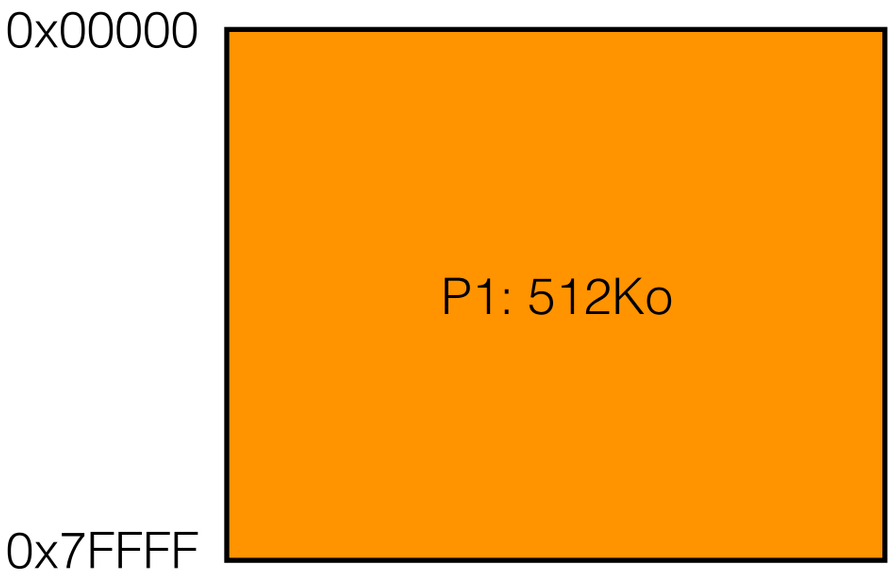
- En allocation contigüe avec partitions de taille variable, le **MMU** effectue le calcul suivant pour traduire une adresse **virtuelle** en adresse **physique**:

$$a_{physique} = a_{virtuelle} + a_{partition}$$

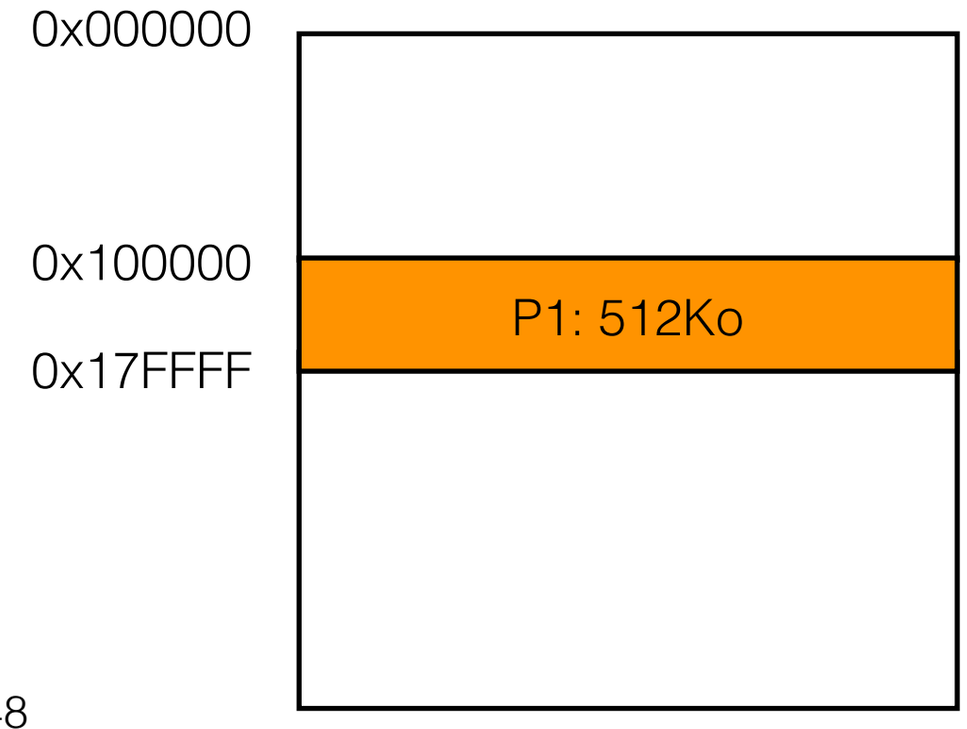
$a_{partition}$ est la première adresse de la partition attribuée au processus

- Par exemple, l'adresse virtuelle 0x00AB3 correspond à l'adresse physique 0x100AB3 dans l'exemple ci-bas.

Adresses **virtuelles**
(du point de vue du **processus**)



Adresses **physiques**
(du point de vue de la **mémoire**)



Allocation contigüe (partitions variables): MMU

- En allocation contigüe avec partitions de taille variable, le **MMU** effectue le calcul suivant pour traduire une adresse **virtuelle** en adresse **physique**:

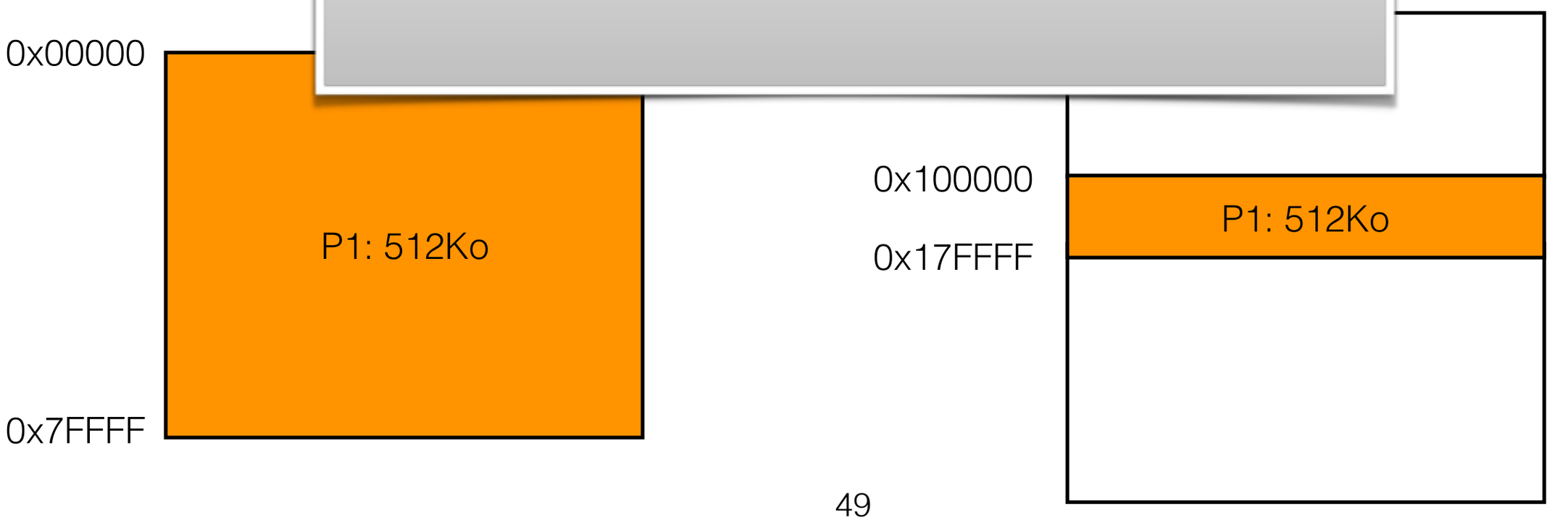
$$a_{physique} = a_{virtuelle} + a_{partition}$$

$a_{partition}$ est la première adresse de la partition attribuée au processus

- Par exemp hysique 0x100AB3

Adre **ques** (du point c **mémoire**)

Que les partitions soient de taille variable ou de taille fixe, le calcul effectué par le MMU est le même.



Exercice mémoire virtuelle

- Dans un système en allocation contigüe avec partitions à taille variable, un processus P1 occupe les adresses en mémoire physique 0x2000 à 0x4FFF.
  - Aux «yeux» de P1, quelle est sa plage d'adresses (virtuelles) disponible?

Rappel

$a_{physique} = a_{virtuelle} + a_{partition}$

Exercice mémoire virtuelle

- Dans un système en allocation contigüe avec partitions à taille variable, un processus P1 occupe les adresses en mémoire physique 0x2000 à 0x4FFF.
  - Aux «yeux» de P1, quelle est sa plage d'adresses (virtuelles) disponible?
    - 0x0000 à 0x2FFF

Rappel

$a_{physique} = a_{virtuelle} + a_{partition}$

Exercice mémoire virtuelle

- Dans un système en allocation contigüe avec partitions à taille variable, un processus P1 occupe les adresses en mémoire physique 0x2000 à 0x4FFF.
  - Quelle est la taille totale de mémoire virtuelle disponible pour ce processus?

Rappel

$a_{physique} = a_{virtuelle} + a_{partition}$

Exercice mémoire virtuelle

- Dans un système en allocation contigüe avec partitions à taille variable, un processus P1 occupe les adresses en mémoire physique 0x2000 à 0x4FFF.
  - Quelle est la taille totale de mémoire virtuelle disponible pour ce processus?
  - 12Ko

Rappel

$a_{physique} = a_{virtuelle} + a_{partition}$

Exercice mémoire virtuelle

- Dans un système en allocation contigüe avec partitions à taille variable, un processus P1 occupe les adresses en mémoire physique 0x2000 à 0x4FFF.
  - Quelle est l'adresse physique correspondant aux adresses virtuelles:
    - 0x0010?
    - 0x1234?

Rappel

$a_{physique} = a_{virtuelle} + a_{partition}$

Exercice mémoire virtuelle

- Dans un système en allocation contigüe avec partitions à taille variable, un processus P1 occupe les adresses en mémoire physique 0x2000 à 0x4FFF.
  - Quelle est l'adresse physique correspondant aux adresses virtuelles:
    - 0x0010? — 0x2010
    - 0x1234? — 0x3234

Rappel

$a_{physique} = a_{virtuelle} + a_{partition}$

Récapitulation: allocation contigüe

1. Un nouveau programme est copié dans un emplacement disponible en mémoire, dans un bloc de mémoire contigu.

2. On peut créer des partitions de taille:

    - **fixe**: la première partition disponible est choisie quand un nouveau processus doit être alloué
    - **variable**: on doit déterminer où créer la partition, nécessite le choix d'un algorithme d'allocation mémoire plus compliqué

3. Le programme utilise des adresses «virtuelles»

4. Le **MMU** traduit les adresse **virtuelles** en adresses **physiques**