

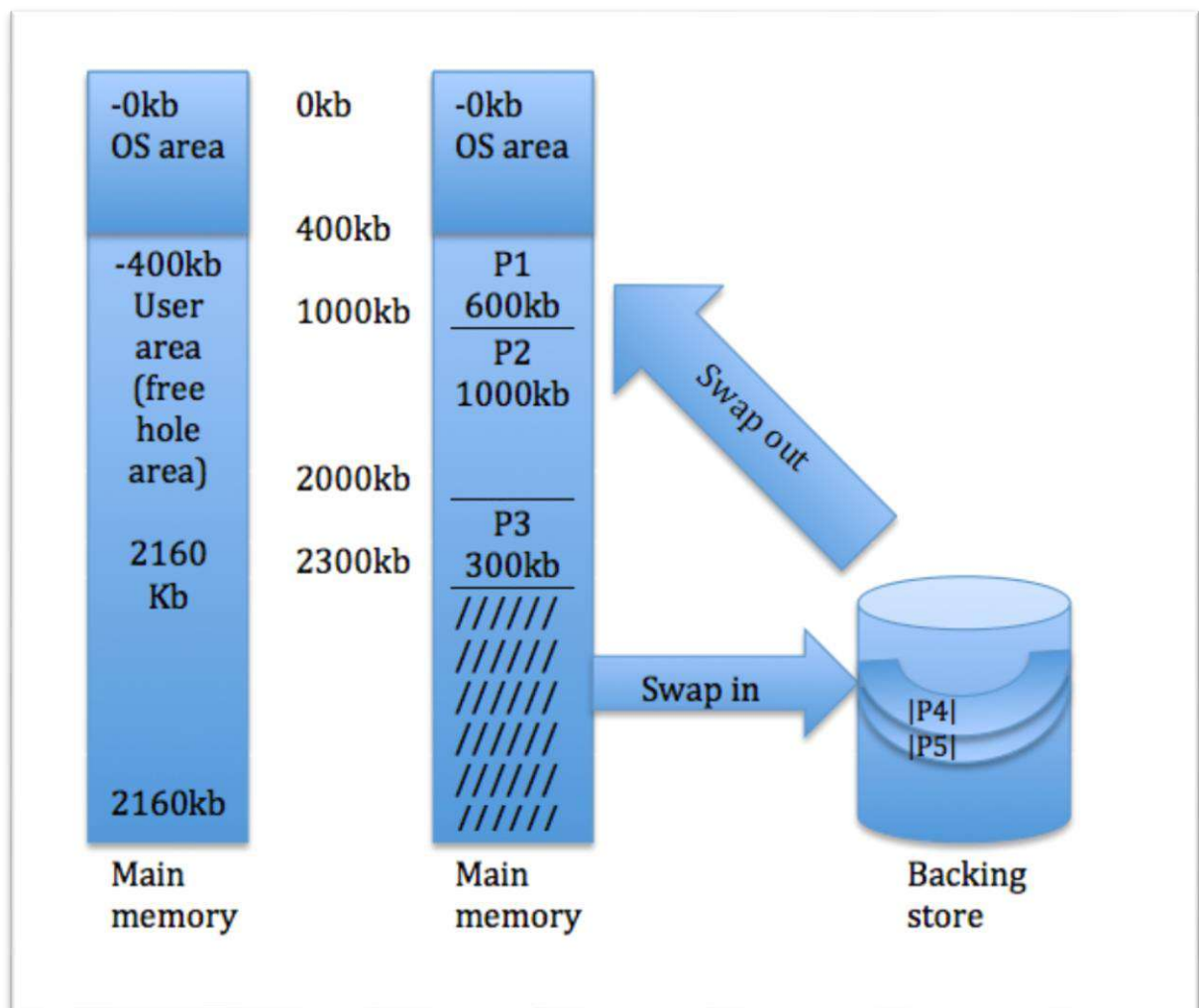
## Memory management

They are two types of memory partitioning strategies :

- Fixed memory partitioning strategy
- Dynamic (variable) memory positioning strategy

### Dynamic memory portioning strategy

| Process | Size (in Kb) | CPU Burst time (in ms) |
|---------|--------------|------------------------|
| P1      | 600          | 6                      |
| P2      | 1000         | 10                     |
| P3      | 300          | 3                      |
| P4      | 700          | 7                      |
| P5      | 500          | 5                      |



Swapper : It is a sub component of an OS. It is a software which does the job of swap-out and swap-in upon request from the CPU. The swapper transfers the process to and from the backing-store and the main memory.

Each process is logically sub-divided into smaller sub-processes called “pages”. A page refers to a position of a process.

Process is heavy while a page is light. Each process is sub-divided into light-weight pages so that each page can be easily transferred to and from the main memory and the backing-store.

Pages : Upon request from the CPU, pages connected to each process are transferred to and from the main memory and the backing-store. And this transfers activity is done by the pages. It is also a sub-component of an OS. It is a software.

Backing-store : It refers to a hard-disc.

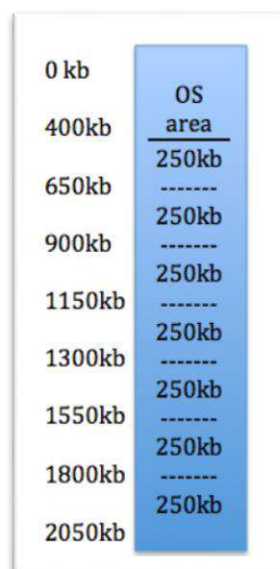
Every ended process, temporarily suspended processes are kept inside the backing-store.

Swap-in : Whenever the CPU requires the pages connected to the process are transferred from the backing-store to the main memory.

Swap-out : To accommodate new process inside the main memory, unwanted or aged processes will be transferred from the main memory to the backing-store.

Free hole : The left-over space in the main memory.

### Fixed memory partitioning strategy



*(erreur à partir de 1300, décalage de 100)*

In the fixed-sized memory partitioning strategy, the memory is partitioned into equal sized portions. Each portion (partition) can hold only one process at a time. Free hole space can not be re-used.

### Compaction technique

This technique merges all free holes together in the main memory to create bigger one in order to be available for processes. These holes can be used for bigger processes than if we didn't have used the compaction technique.

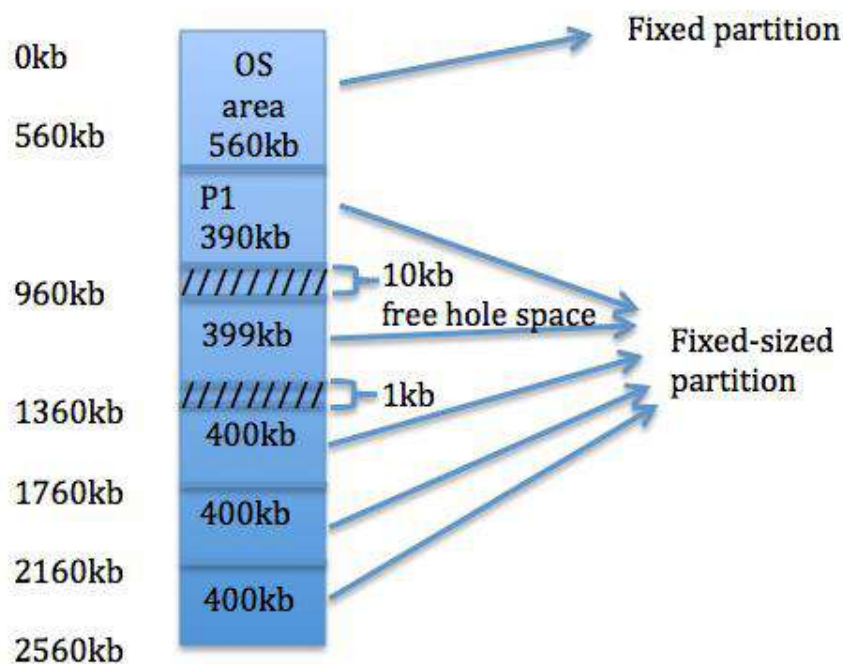
This free hole is placed at the low address (If they are two free holes, 300 to 400kb and 600 to 1000kb, the final free hole will be from 300 to 800kb).

This technique is used as the best solution to solve the problem of "External fragmentation". The dynamic memory partition suffers from this "External fragmentation".

Eventhough the total size of all free hole areas is bigger than the size of the needy process (example P4), the CPU is unable to accommodate the process P4 inside the RAM (main memory). This situation is called "External fragmentation".

The best solution to overcome is problem is "compaction". The problem with this solution is that it will slow down the performance of the system.

### Internal fragmentation



Allocated size = 400kb

Request = 390kb

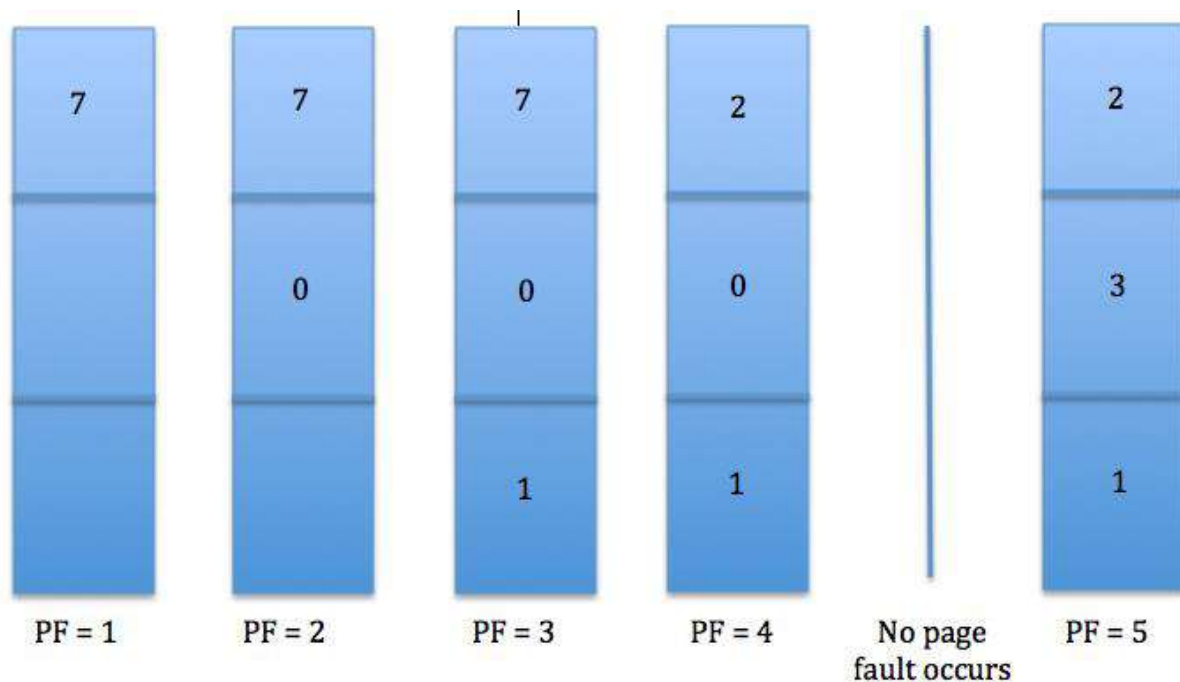
Internal fragmentation : Difference between the allocated space and the requested space. Here we have 10kb internal fragmentation.

There is no solution to solve the "internal fragmentation" problem.

## Page-Replacement algorithms

**Frame** : The main memory space is logically partitioned into equal sized partitions called frames. Each partition is called “frame”, is can hold only one page at a time.

Considering the following page reference string : 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1  
All frames are initially empty and the memory has 3 frames.



The algorithm checks every frame in the RAM before making a page fault. If the page is already in the RAM, even if not in the current frame, the page fault won't occur.

*Note* : There are many algorithms used such as FIFO (as in the example) or LRU (Least recently used).

The FIFO-Page replacement algorithm is the most simplest page replacement algorithm. In this algorithm, the time of arrival of each page is recorded. Newly arrived page is added to tail of the FIFO-queue while pages are removed from the head position in the FIFO-queue.

The FIFO-page replacement algorithm is not an efficient page replacement algorithm because it causes more number of page-faults. Whenever the page required by the CPU is not found inside the RAM a page-fault event occurs.

In this algorithm, the page who came first into RAM, is chosen for swap-out or replacement.