Memory Management

Memory allocation

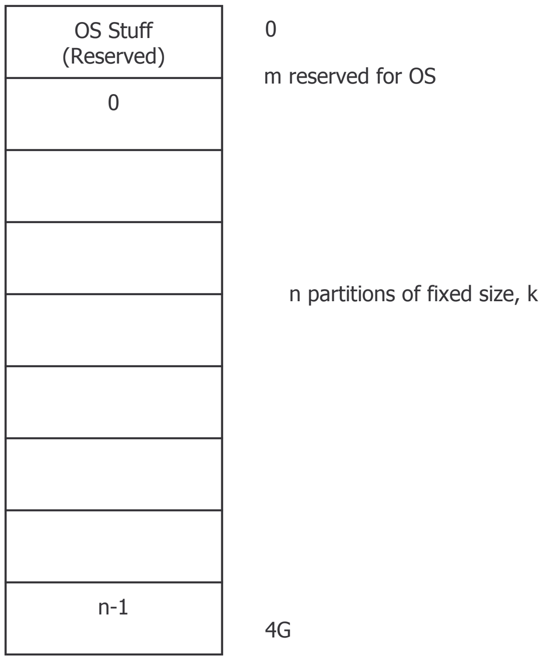
Process of granting available memory resources to the processes running in the OS

Two principal methods for contiguous allocation:

1. Fixed-size memory partitions. One per process
2. Variable-size memory partitions. One per process
    a. We get an unwanted behavior: fragmentation

Fixed-Size Memory Partitions

Simplest scheme
Most restrictive: limits the degree of multiprogramming to *n*



On New:

PCB Structure created
Partition to be identified
Base, limit in PCB are set
Partition table bit set to allocated

Number of available partitions: $n = \frac{2^A - m}{k}$

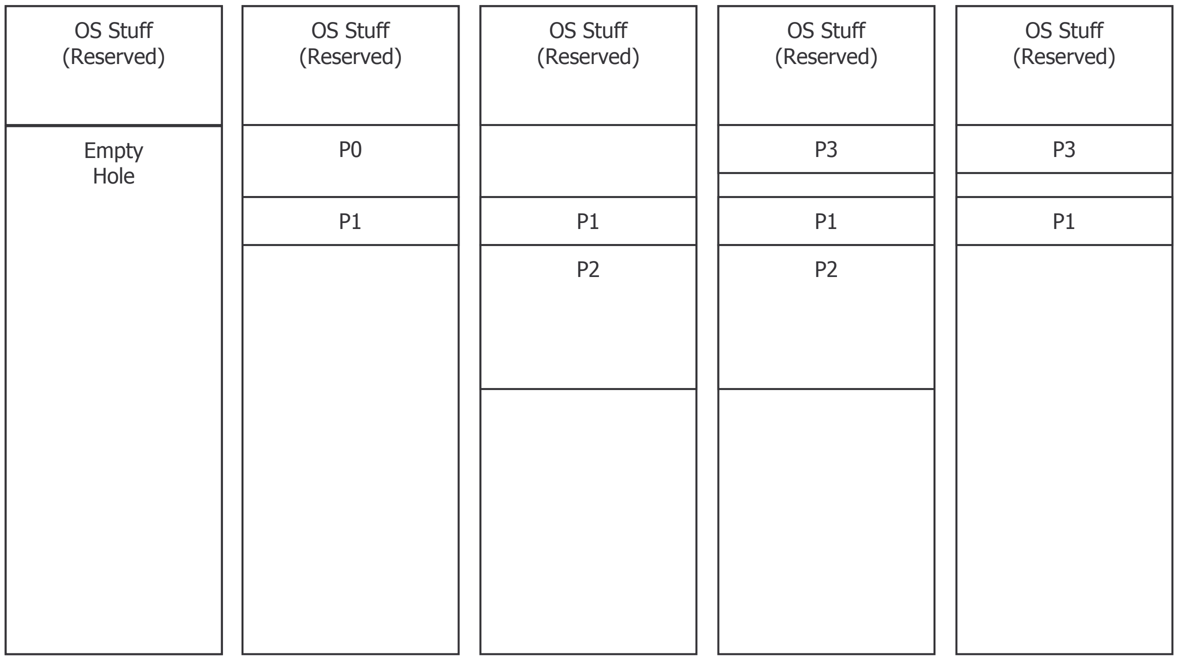
A = # of address bits
m = # of reserved OS bytes
k = size (bytes) of each block / partition
n = # of partitions (and degree of multiprogramming)

If a new process tries to come to life, it will have to wait until another process completes or is swapped out to disk.

Works fairly well in a batch environment (user doesn't expect jobs to be done immediately). Interactivity does not function quite as nicely.

Variable Size Memory Partitions

Memory usage is dynamically allocated and thus lends itself to better support of interactive-type processes.



Like a hole – we fill the hole with material

Fragmentation occurs when the blocks of memory are no longer contiguous

The OS doing this type of memory management must have a table so that it knows what bases and limits have been assigned. (hole table)

Dynamic allocation problem strategies

1. **First fit**
    a. Hole table is scanned from beginning to end and process is assigned the *entire* first hole that it can fit into and the base and limit values in the PCB are set. That hole is marked as being allocated.
    b. **50% rule:** Under first fit, if you allocate n blocks of memory, another .5n blocks will be unusable
2. **Best fit**

   a. Waste is not tolerated. All of the holes are searched. The one that most closely matches the requirements is assigned.

3. **Worst fit**
   a. The one that is the biggest compared to what the process needs is assigned.

First fit is the fastest and easiest to implement. Best and worst fit take a performance hit because you have to determine how the holes fit it comparison to the needed size, but best fit helps to minimize the total fragmentation that occurs.

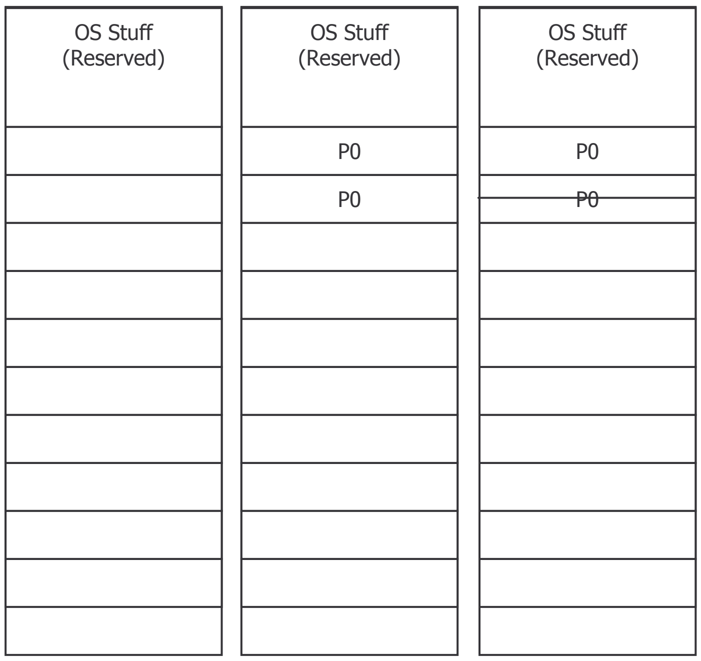
External Fragmentation

The result of best fit and worst fit. Occurs because, as processes move in and out of memory, you get many small non-contiguous holes.

A process attempts to be loaded, but there are no contiguous holes that are large enough.

Internal Fragmentation

Results when a process does not need all of the bytes in its allocated holes and therefore renders those bytes unusable by other processes



(2k) bytes (1.5k) bytes

Compaction (Defragmentation)

The process of searching what is currently in memory, moving those processes to empty holes so that you end up with one big empty hole at the end.

Re-ordering and reassignment to leave one big contiguous hole.

The bigger the memory, the longer it will take to defragment.

Time-exhaustive, but it can be used as needed to put the system back in order so that the degree of programming can go up.