Memory Management

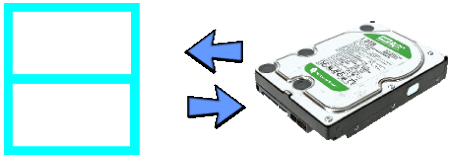
Johan Wahlström

ECM1413 Computers and the Internet

Processes

- A process is a program in execution
- Multiple processes may be active at the same time
- Each process is represented by a process control block (PCB) which is stored in memory (or on disk)
- How do we allocate space to processes in memory in an efficient way?

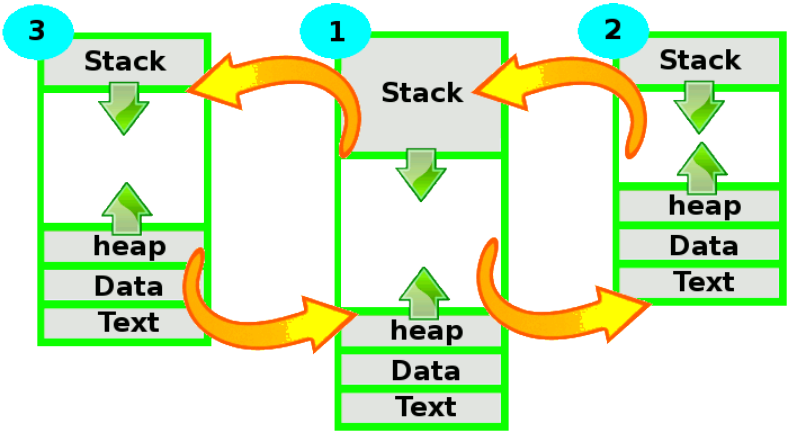
Memory Management Tasks



Memory management tasks include

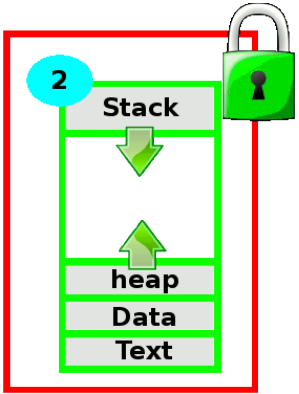
1. relocation
2. protection
3. sharing

(1) Relocation



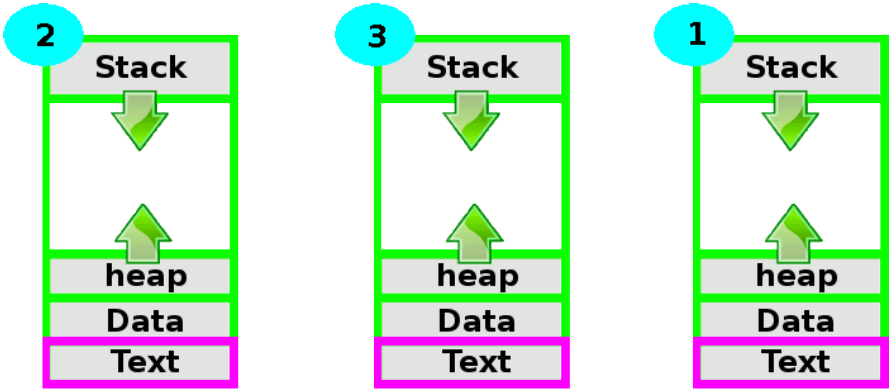
A memory management technique needs to be able to relocate a process to a new place in memory.

(2) Protection



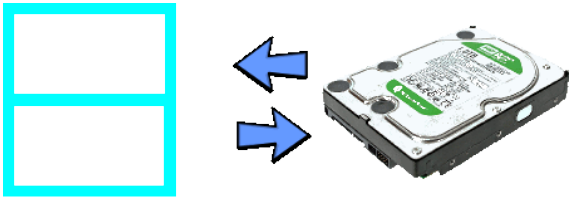
All processes need to be protected against interference from other processes.

(3) Sharing



Processes may sometimes need to share data.

Memory Management Techniques

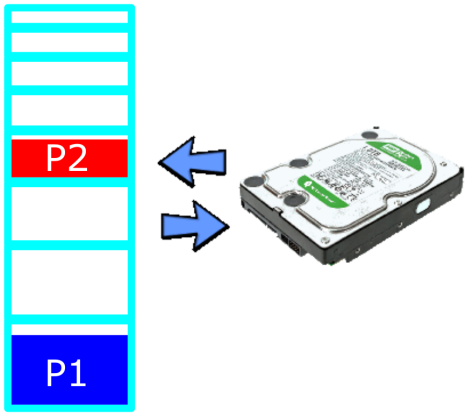


We will discuss the following memory management techniques:

1. fixed partitioning
2. dynamic partitioning
3. simple segmentation
4. virtual memory segmentation
5. simple paging
6. virtual memory paging

(1) Fixed Partitioning

- Memory is divided into partitions of a fixed size; the size may vary between partitions
- A process can be loaded into any partition whose size is equal to or greater than the size of the process
- One-to-one mapping between partitions and processes

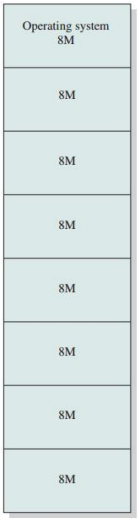


(1) Fixed Partitioning – Unequal or Equal Partition Sizes

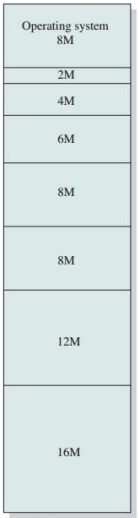
Risks with equal-sized partitions:

- Internal fragmentation: Space is wasted inside partitions when the size of the process is smaller than the size of the partition
- There may not be any partition that can fit a large process

⇒ Unequal-sized partitions are more efficient.



Equal-size partitions



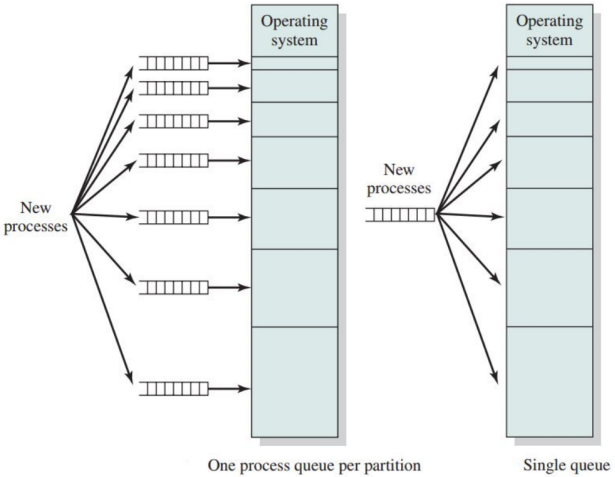
Unequal-size partitions

(1) Fixed Partitioning – Queuing

Assume unequal-sized partitions.

How do we match processes to partitions?

- Alternative 1 (one process queue per partition): Assign each process to the smallest partition where it will fit
  - Could mean that some partitions remain empty
- Alternative 2 (single queue): Assign each process to the smallest available partition where it will fit



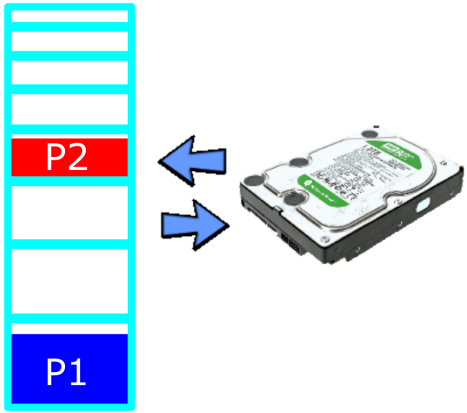
(1) Fixed Partitioning – Pros and Cons

Pros:

- Easy to understand and implement

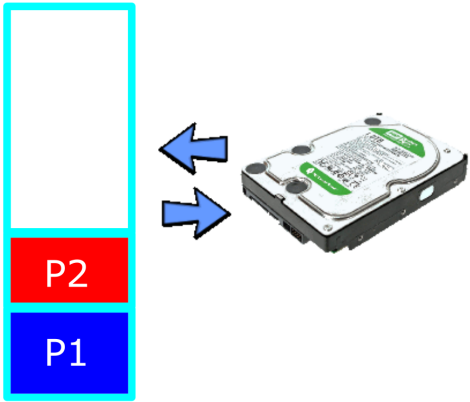
Cons:

- Internal fragmentation (wasted space inside partitions)
- A pre-specified upper limit on the number of processes
- A pre-specified upper limit on the size of the largest process

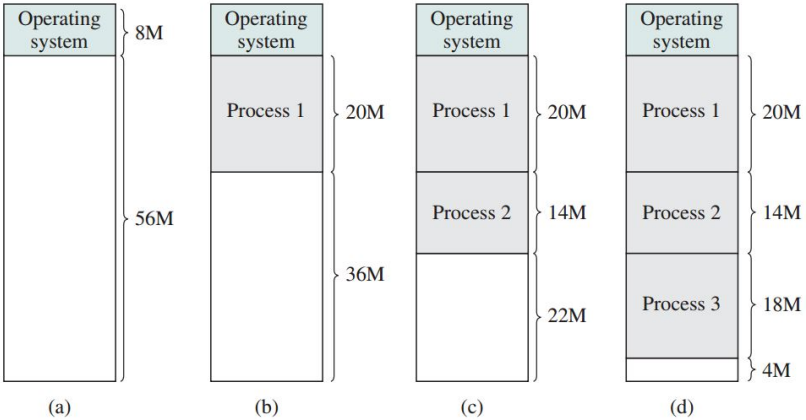


(2) Dynamic Partitioning

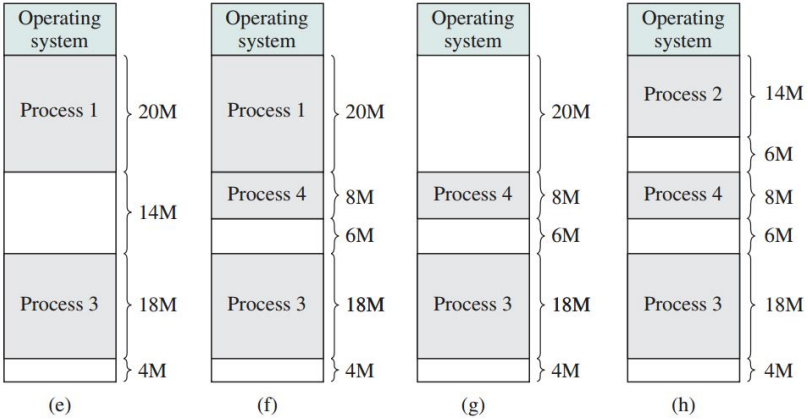
- The partition sizes adjust to the sizes of processes
- One-to-one mapping between partitions and processes



(2) Dynamic Partitioning – Example

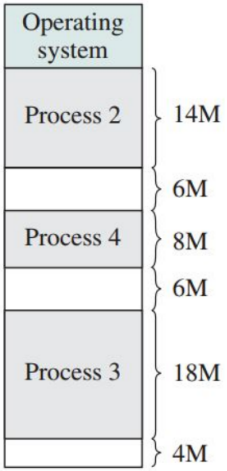


(2) Dynamic Partitioning – Example (ctd)



(2) Dynamic Partitioning – External Fragmentation

- External fragmentation: Space is wasted between partitions
- Compaction: Shift processes so that they are contiguous and so that all free memory is gathered in one continuous block
  - Downside: Takes a lot of time and resources



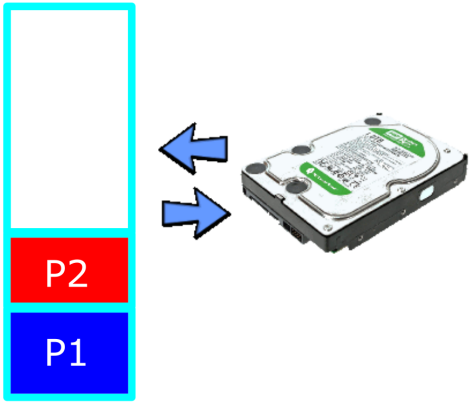
(2) Dynamic Partitioning – Pros and Cons

Pros:

- No internal fragmentation
- No limit on the number of processes or on the size of the largest process

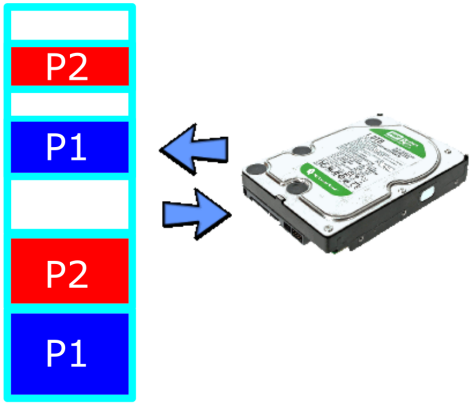
Cons:

- External fragmentation (wasted space between partitions)
- Time and resources wasted on compaction



(3) Simple Segmentation

- Each program is divided into segments (e.g., text and data segments)
- Segments are loaded into memory as in dynamic partitioning
- All segments need to be loaded into memory, but they do not need to be contiguous



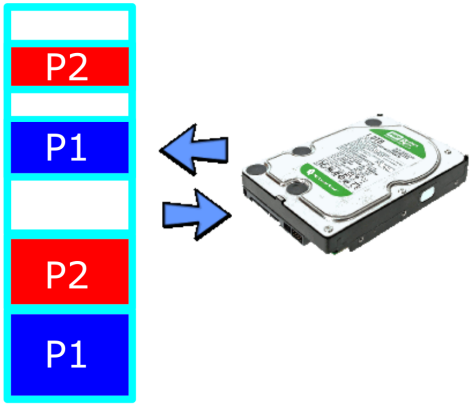
(3) Simple Segmentation – Pros and Cons

Pros:

- Easier to fit processes in memory than with dynamic partitioning
- No internal fragmentation
- No limit on the number of processes or on the size of the largest process

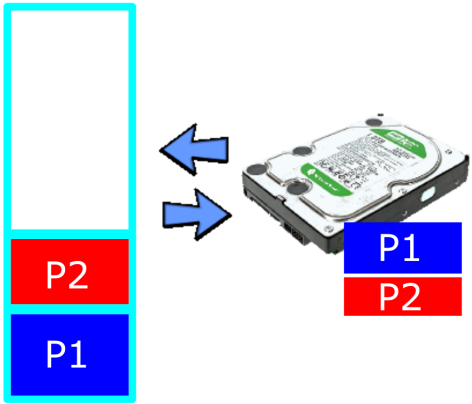
Cons:

- External fragmentation (less than dynamic partitioning)
- Time and resources wasted on compaction



(4) Virtual Memory Segmentation

- Virtual memory: Use disk storage as if it was main memory
  - Makes it seem as if main memory is very big
- Segments are loaded into memory as in dynamic partitioning, except that not all segments need to be loaded at the same time



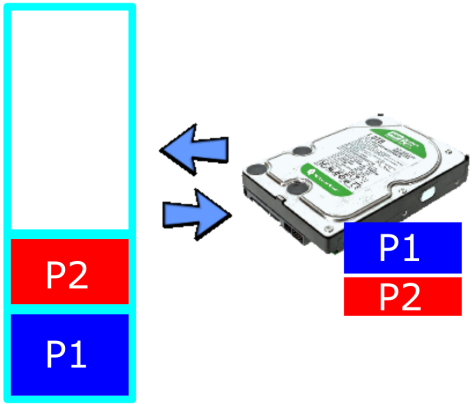
(4) Virtual Memory Segmentation – Pros and Cons

Pros:

- Easier to fit processes in memory than with simple segmentation
- No internal fragmentation
- No limit on the number of processes or on the size of the largest process

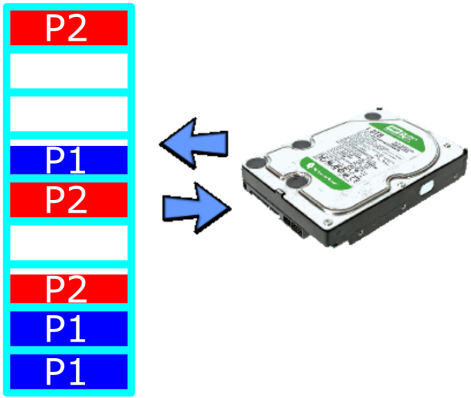
Cons:

- More overhead required for virtual memory
- External fragmentation (less than dynamic partitioning)
- Time and resources wasted on compaction

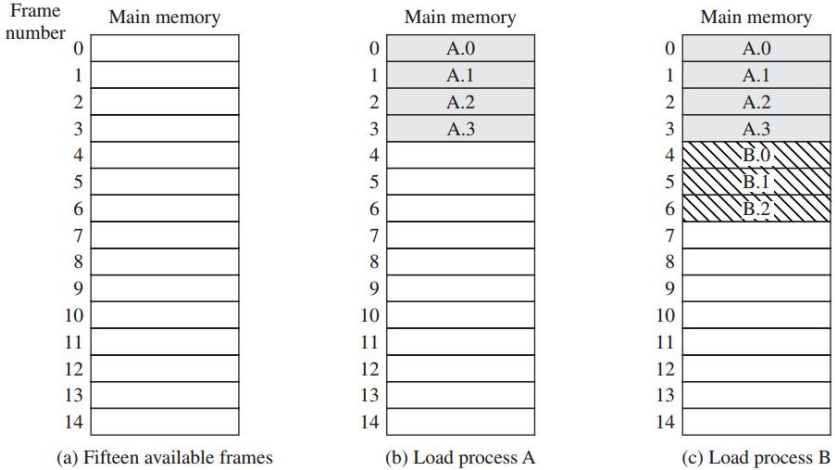


(5) Simple Paging

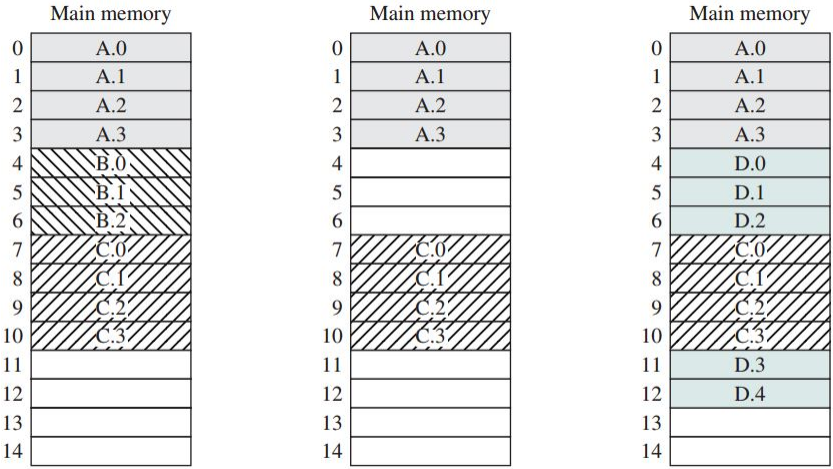
- Memory is divided into fixed-size frames of equal size
- Each process is divided into pages of the same size
- Internal fragmentation is small since
  - it only affects the last frame of each process
  - the frame/page-size is fairly small



(5) Simple Paging – Example



(5) Simple Paging – Example (ctd)



(d) Load process C (e) Swap out B (f) Load process D

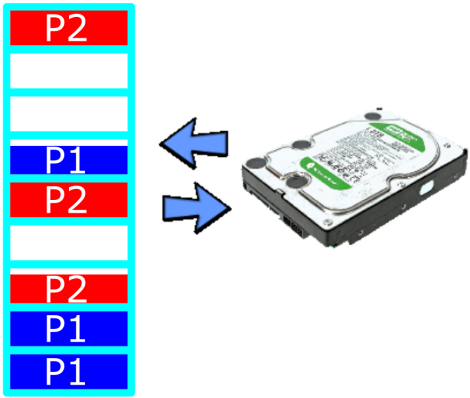
(5) Simple Paging – Pros and Cons

Pros:

- No external fragmentation
- All pages fit perfectly into frames

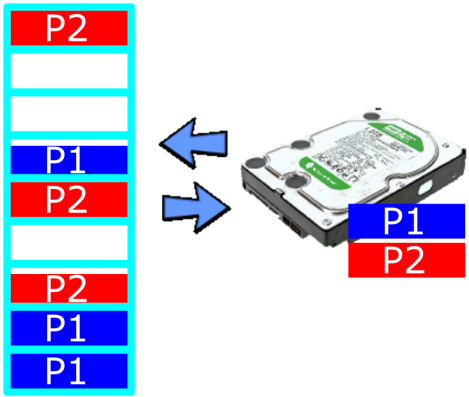
Cons:

- (small) Internal fragmentation
- Each process requires a page tables (which consumes memory)



(6) Virtual Memory Paging

- Pages are loaded into frames as in simple paging, except that not all pages need to be loaded at the same time



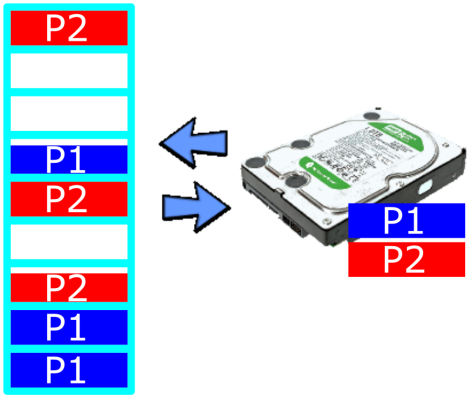
(6) Virtual Memory Paging – Pros and Cons

Pros:

- Easier to fit processes in memory than with simple paging
- No external fragmentation
- All pages fit perfectly into frames

Cons:

- More overhead required for virtual memory
- (small) Internal fragmentation
- Each process requires a page tables (which consumes memory)



Protection and Sharing within Segmentation and Paging

Goals:

- Protection: Each process can only access its own pages/segments
- Sharing: Two processes access the same page/segment

Implementation:

- Each frame/segment has a number of protection bits specifying
  - read-write or read-only
  - which process can access

Protection/sharing is generally easier with segmentation than with paging since

- the segment table is smaller than the page table
- segments are more natural units for protection/sharing