Frame | A fixed-length block of main memory.
---|---
Page | A fixed-length block of data that resides in secondary memory (such as disk). A page of data may temporarily be copied into a frame of main memory.
Segment | A variable-length block of data that resides in secondary memory. An entire segment may temporarily be copied into an available region of main memory (segmentation) or the segment may be divided into pages which can be individually copied into main memory (combined segmentation and paging).
Memory Partitioning

- Memory management brings processes into main memory for execution by the processor
  - Involves virtual memory
  - Based on segmentation and paging
- Partitioning
  - Used in several variations in some now-obsolete operating systems
  - Does not involve virtual memory

<table>
<thead>
<tr>
<th>Technique</th>
<th>Description</th>
<th>Strengths</th>
<th>Weaknesses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed Partitioning</td>
<td>Main memory is divided into a number of static partitions at system generation time. A process may be loaded into a partition of equal or greater size.</td>
<td>Simple to implement; little operating system overhead.</td>
<td>Inefficient use of memory due to internal fragmentation.</td>
</tr>
<tr>
<td>Dynamic Partitioning</td>
<td>Partitions are created dynamically so that each process is loaded into a partition of exactly the same size as that process.</td>
<td>No internal fragmentation; more efficient use of main memory.</td>
<td>Inefficient use of processor due to the need for comparison to counter external fragmentation.</td>
</tr>
<tr>
<td>Simple Paging</td>
<td>Main memory is divided into a number of equal-size frames. Each process is divided into a number of equal-size pages of the same length as frames. A process is loaded by loading all of its pages into contiguous, frames.</td>
<td>No external fragmentation.</td>
<td>A small amount of internal fragmentation.</td>
</tr>
<tr>
<td>Simple Segmentation</td>
<td>Each process is divided into a number of segments. A process is loaded by loading all of its segments into dynamic partitions that need not be contiguous.</td>
<td>No internal fragmentation; improved memory utilization and reduced overhead compared to dynamic partitioning.</td>
<td>External fragmentation.</td>
</tr>
<tr>
<td>Virtual Memory Paging</td>
<td>As with simple paging, except that it is not necessary to load all of the pages of a process. Nonresident pages that are needed are brought in later automatically.</td>
<td>No internal fragmentation; higher degree of multiprogramming; large virtual address space.</td>
<td>Overhead of complex memory management.</td>
</tr>
<tr>
<td>Virtual Memory Segmentation</td>
<td>As with simple segmentation, except that it is not necessary to load all of the segments of a process. Nonresident segments that are needed are brought in later automatically.</td>
<td>No internal fragmentation, higher degree of multiprogramming, large virtual address space; protection and sharing support.</td>
<td>Overhead of complex memory management.</td>
</tr>
</tbody>
</table>

Table 7.2 Memory Management Techniques

(Table is on page 317 in textbook)
Disadvantages

- A program may be too big to fit in a partition
  - Program needs to be designed with the use of overlays

- Main memory utilization is inefficient
  - Any program, regardless of size, occupies an entire partition
  - Internal fragmentation
    - Wasted space due to the block of data loaded being smaller than the partition
The number of partitions specified at system generation time limits the number of active processes in the system

Small jobs will not utilize partition space efficiently
Dynamic Partitioning

- Partitions are of variable length and number
- Process is allocated exactly as much memory as it requires
- This technique was used by IBM’s mainframe operating system, OS/MVT

Figure 7.4 The Effect of Dynamic Partitioning
Dynamic Partitioning

External Fragmentation
- Memory becomes more and more fragmented
- Memory utilization declines

Compaction
- Technique for overcoming external fragmentation
- OS shifts processes so that they are contiguous
- Free memory is together in one block
- Time consuming and wastes CPU time

Placement Algorithms

Best-fit
- Chooses the block that is closest in size to the request

First-fit
- Begins to scan memory from the beginning and chooses the first available block that is large enough

Next-fit
- Begins to scan memory from the location of the last placement and chooses the next available block that is large enough
Buddy System

- Comprised of fixed and dynamic partitioning schemes
- Space available for allocation is treated as a single block
- Memory blocks are available of size $2^K$ words, $L \leq K \leq U$, where
  - $2^L = \text{smallest size block that is allocated}$
  - $2^U = \text{largest size block that is allocated}; \text{generally } 2^U \text{ is the size of the entire memory available for allocation}
**Figure 7.6** Example of Buddy System

<table>
<thead>
<tr>
<th>1 Mbyte block</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 M</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Request 100 K</th>
<th>A = 128K</th>
<th>128K</th>
<th>256K</th>
<th>512K</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request 240 K</td>
<td>A = 128K</td>
<td>128K</td>
<td>B = 256K</td>
<td>512K</td>
</tr>
<tr>
<td>Request 64 K</td>
<td>A = 128K</td>
<td>C = 64K</td>
<td>64K</td>
<td>B = 256K</td>
</tr>
<tr>
<td>Request 256 K</td>
<td>A = 128K</td>
<td>C = 64K</td>
<td>64K</td>
<td>B = 256K</td>
</tr>
<tr>
<td>Release B</td>
<td>A = 128K</td>
<td>C = 64K</td>
<td>64K</td>
<td>256K</td>
</tr>
<tr>
<td>Release A</td>
<td>128K</td>
<td>C = 64K</td>
<td>64K</td>
<td>256K</td>
</tr>
<tr>
<td>Request 75 K</td>
<td>E = 128K</td>
<td>C = 64K</td>
<td>64K</td>
<td>256K</td>
</tr>
<tr>
<td>Release C</td>
<td>E = 128K</td>
<td>128K</td>
<td>256K</td>
<td>D = 256K</td>
</tr>
<tr>
<td>Release E</td>
<td>512K</td>
<td>D = 256K</td>
<td>256K</td>
<td></td>
</tr>
<tr>
<td>Release D</td>
<td>1M</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 7.7** Tree Representation of Buddy System

1M

512K

256K

128K

64K

Leaf node for allocated block

Leaf node for unallocated block

Non-leaf node