Chapter 9: Virtual Memory

- Background
- Demand Paging
- Process Creation
- Page Replacement
- Allocation of Frames
- Thrashing
- Demand Segmentation
- Operating System Examples
Background

- Virtual memory – separation of user logical memory from physical memory.
  - Only part of the program needs to be in memory for execution.
  - Logical address space can therefore be much larger than physical address space.
  - Allows address spaces to be shared by several processes.
  - Allows for more efficient process creation.

- Virtual memory can be implemented via:
  - Demand paging
  - Demand segmentation

Virtual Memory That is Larger Than Physical Memory

The diagram illustrates a virtual memory system where pages are mapped to physical memory. Each page in virtual memory corresponds to one or more pages in physical memory, as indicated by the memory map. The diagram shows how pages can be swapped in and out of physical memory as needed, allowing the system to manage memory more efficiently when the physical memory is limited.
Virtual-address Space

Demand Paging

- Bring a page into memory only when it is needed
  - Less I/O needed
  - Less memory needed
  - Faster response
  - More users

- Page is needed $\Rightarrow$ reference to it
  - invalid reference $\Rightarrow$ abort
  - not-in-memory $\Rightarrow$ bring to memory
Transfer of a Paged Memory to Contiguous Disk Space

Valid-Invalid Bit

- With each page table entry a valid–invalid bit is associated
  (1 ⇒ in-memory, 0 ⇒ not-in-memory)
- Initially valid–invalid but is set to 0 on all entries
- Example of a page table snapshot:

<table>
<thead>
<tr>
<th>Frame #</th>
<th>valid-invalid bit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

- During address translation, if valid–invalid bit in page table entry is 0 ⇒ page fault
Page Table When Some Pages Are Not in Main Memory

If there is ever a reference to a page, first reference will trap to OS ⇒ page fault
OS looks at another table to decide:
- Invalid reference ⇒ abort.
- Just not in memory.

Get empty frame.
Swap page into frame.
Reset tables, validation bit = 1.
Restart instruction: Least Recently Used
- block move
- auto increment/decrement location
Steps in Handling a Page Fault

What happens if there is no free frame?

- Page replacement – find some page in memory, but not really in use, swap it out
  - algorithm
  - performance – want an algorithm which will result in minimum number of page faults
- Same page may be brought into memory several times
Thrashing

- If a process does not have "enough" pages, the page-fault rate is very high. This leads to:
  - low CPU utilization
  - operating system thinks that it needs to increase the degree of multiprogramming
  - another process added to the system

- **Thrashing** = a process is busy swapping pages in and out

Thrashing (Cont.)

[Diagram showing CPU utilization and degree of multiprogramming with a peak labeled "thrashing".]