Memory Management

Types

• Basic two types of memory management techniques
• Swapping
• Demand Paging
Basic Concept Demand Paging

- Partition physical memory into smaller chunks (pages)
- A process memory is managed in terms of pages
- Kernel only loads frames when needed
- Use a page table to hold referenced frames (pages)

Benefits Of Paging

- Increased utilization of memory
- Reduced I/O for swapping a process in/out of memory
- Faster startup time for processes
Cons of Page

• Addition kernel structures required for virtual memory management
• Complexity

Data Structures for DP

• Disk block descriptors
• Page frame data table (pfdata)
• Swap use table
Region Structure

Page Table & Disk Block Disp.

<table>
<thead>
<tr>
<th>Page Table Entry</th>
<th>Disk Block Descriptor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page (Physical)</td>
<td>Swap Dev</td>
</tr>
<tr>
<td>Address</td>
<td>Block Num</td>
</tr>
<tr>
<td>Age</td>
<td>Type (swap, file,</td>
</tr>
<tr>
<td>Cp/Wrt/Mod</td>
<td>fill 0, demand fill)</td>
</tr>
<tr>
<td>Ref</td>
<td>Val</td>
</tr>
<tr>
<td>Prot</td>
<td></td>
</tr>
</tbody>
</table>

Figure 9.13. Page Table Entries and Disk Block Descriptors
Pfdata

- Data contained in Pfdata entry
  - Page state (swap device, executable, currently being copied, etc.)
  - Number of processes that reference the page
  - Logical device and block number
  - Pointers to other pfdata entries

Rel. Data Structures for DP

Figure 9.14. Relationship of Data Structures for Demand Paging
Fork in Demand Paging

- Does not copy every region of parent process, instead it manipulates region tables, page table entries and pfdata table entries.
- Incrementing reference counts for shared regions
- Allocates new region tables entries for private.

Fork in Demand Paging

- If page is valid in parent, increment count in pfdata table entry of child processes
- Only when page is modified is a copy of the page made for child/parent process.
- Similar process if page is on swap device.
Exec in Demand Paging

- Entire file may not be read into memory initially
- Instead
  - kernel creates an “Block List” and attaches it to the Inode for the file
  - Disk block descriptor are filled with logical block No. within Block List
  - The block list contains the actual disk block number

Example of Exec (DP)

![Diagram of Inode, Disk Block Descriptor, Logical Block 84, Block List, and Region Mapping](image)
Running out of Free Pages

- Can the system run out of free pages?
- Page Stealer
  - Kernel Process
  - Swaps out pages not in a process working set
  - Runs periodically

Page Stealer Algorithm

Examine every page
If locked
  skip
Else
Age page
If pageAge > ageLimit
  swap page out
Continue
States for a Swappable Page

• Swap device does not have a copy of the page
• Swap device does have a copy of the page and the page has not been modified
• Swap device has a copy of the page, but the in-core copy of page has been modified.
• Of the identified states which require page be copied out to swap device?

Page Fault

• Occurs when process accesses a page who’s valid bit is not set.
• When is the valid bit of a page table entry not set?
• Validity faults
  – Address that are part of virtual memory, but do not have a physical page assigned
• Protection Faults
  – Address that are out side of the virtual address space of the process
Valid Page Fault Cases

- Basically Five cases
  1. On Swap Device & not in memory
  2. On free page list in memory
  3. In an executable file
  4. Marked “demand zero”
  5. Marked “demand fill”

Valid Fault Example

![Diagram showing page table entries and disk block usage](Figure 9.22. Occurrence of a Volatility Fault)
Valid Fault Example

1. Reference addr. 66K
2. Valid == false
3. Read file from disk block 847.
4. Disk BLK 847 not in page cache.
5. Kernel assigns new page (phy. 1776) to hold contents of virtual addr. 66K

Swapping Graph