Memory Management

CSE 2431: Introduction to Operating Systems
Reading: §§8.1–8.3, [OSC]
Outline

• Basic Memory Management
• Swapping
• Variable Partitions
• Memory Management Problems
Basic Memory Management

• Storage Hierarchy
• Memory Management Problems
• Fixed Partitions for Multiprogramming
• Variable Sized Partitions
• Memory Allocation Strategies
Storage Hierarchy

Cost

- $600 a chip
- $10^{-2}$ per byte
- $10^{-4}$ per byte
- $10^{-8}$ per byte

Size

- CPU Reg: 32–64 bits
- Cache: 4–128 words
- Memory: 512–16K words
- Secondary Storage: 2^{13}–2^{40} bytes
General Memory Problem

• Limited (expensive) physical resources: Main memory
  – E.g. Windows Vista recommends 1G and prefers 2G RAM
  – We want to use it as efficiently as possible

• Abundant, slower resources: Disk

• OS needs to provide an abstraction of memory hierarchy to user level applications
Responsibilities of Memory Manager

- Manage memory hierarchy
  - Monitor used and free memory
  - Allocate memory to processes
  - Reclaim (De-allocate) memory
  - Swapping between main memory and disk
Scenarios of Memory Management Problems

• One program, the size is less than memory size
  – The simplest case
• One program, using lots of memory
  – Can you only keep part of the program in memory?
• Many programs, total size is less than memory size
  – Technically possible to pack them together
  – Will programs know about each other’s existence?
• Lots of programs, total size exceeds memory size
  – What programs are in memory, and how to decide?
Mono-Programming No Swapping

- Run one process at a time
  - simplest possible memory management scheme
- Memory is shared only between OS and the process.
- Three different ways to organize memory

```
<table>
<thead>
<tr>
<th>User Program</th>
<th>OS in ROM</th>
<th>Device drivers in ROM</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS in RAM</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>User Program</th>
<th>OS in ROM</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS in RAM</td>
<td></td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>User Program</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS in RAM</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>User Program</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS in RAM</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>User Program</th>
</tr>
</thead>
</table>
```

```
<table>
<thead>
<tr>
<th>User Program</th>
</tr>
</thead>
</table>
```

```
<table>
<thead>
<tr>
<th>User Program</th>
</tr>
</thead>
</table>
```
Overlaying

Used when the process memory requirement exceeds the physical memory space
Multiprogramming with Fixed Partitions

• Divide memory into $n$ (possibly unequal) partitions.

• Problem:
  – Internal Fragmentation
Fixed Partition Allocation

• Separate input queue for each partition
  – Requires sorting the incoming jobs and putting them into separate queues
  – Problems?

• One single input queue for all partitions.
  – Find a job for fitting in an available partition
    • Available Fit
    • Best Fit
  – Problems?
Relocation

- Correct starting address when a program should start in the memory
- Different jobs will run at different addresses
  - When a program is linked, the linker must know at what address the program will begin in memory.
- Logical addresses, Virtual addresses
  - Logical address space, range (0 to $max$
- Physical addresses, Physical address space
  - range ($R+0$ to $R+max$) for base value $R$
- User program never sees the real physical addresses
- Who translates virtual to physical addresses?
  - Program rewriting at loading time
  - Help from relocation registers at execution time
Relocation Register

- CPU Instruction Address
- Logical Address MA
- Base Register BA
- Physical Address MA+BA
- Memory

Diagram:

1. CPU Instruction Address
2. Logical Address MA
3. Base Register BA
4. Physical Address MA+BA
5. Memory
Protection

• Problem:
  – How to prevent a malicious process to write or jump into other user's or OS partitions

• Solutions:
  – Memory protection code
  – Base bounds registers
**Base Bounds Registers**

Logical Address LA  \( \rightarrow \) Base Address BA  \( \rightarrow \) MA+BA Memory

CPU Address  \( \rightarrow \) \( \lt \) Memory Address MA  \( \rightarrow \) Fault

Bounds Register  \( \rightarrow \) Base Register  \( \rightarrow \) Physical Address PA  \( \rightarrow \) Limit Address
Review

• Basic Memory Management
  – Memory Manager’s Responsibilities
  – Mono-Programming
  – Multi-Programming with Fixed Partitions
    • Internal fragmentation
  – Relocation and Protection
Memory Management (More…)

• Batch System
  – Multiprogramming with fixed partitions
  – In the memory until job finishes
  – Keep CPU busy

• Timesharing Systems
  – Not enough memory to hold all active processes
  – Swapping (whole process)
  – Virtual memory (partial process)
Swapping

• Move the whole process to/from disk
• Allows several processes to share a fixed partition
• Processes that grow can be swapped out and swapped back in a bigger partition
Outline

• Basic Memory Management
• Swapping
• Variable Partitions
• Memory Management Problems
Swapping (1)

Monitor

User Partition

Job 1

Disk
Swapping (2)

Monitor

Job 1

Disk

User Partition

Job 1
Swapping (3)
Swapping (4)
Swapping (5)
Swapping (6)
Outline

• Basic Memory Management
• Swapping
• Variable Partitions
• Memory Management Problems
## Variable Partitions and Fragmentation

<table>
<thead>
<tr>
<th></th>
<th>Monitor</th>
<th>Job 1</th>
<th>Job 2</th>
<th>Job 3</th>
<th>Job 4</th>
<th>Free</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

• Fixed Partitions ⇔ Variable Partitions
• External Fragmentation
  - *How to solve it?*
Compaction

- What is it? What does it look like?
- Assumes all programs are relocatable
- Processes must be suspended during compaction
- Need be done only when fragmentation gets very bad

5
Monitor | Job 7 | Job 5 | Job 3 | Job 8 | Job 6

6
Monitor | Job 7 | Job 5 | Job 3 | Job 8 | Job 6

7
Monitor | Job 7 | Job 5 | Job 3 | Job 8 | Job 6

8
Monitor | Job 7 | Job 5 | Job 3 | Job 8 | Job 6

9
Monitor | Job 7 | Job 5 | Job 3 | Job 8 | Job 6 | Free
Outline

• Basic Memory Management
• Swapping
• Variable Partitions
• Memory Management Problems
Memory Management with Bitmaps

- Use bitmaps for memory status (free or allocated)
- Memory is divided into allocation units.
  - One allocation unit corresponds to 1 bit in the bitmap
  - 0: free, 1: allocated
- Size of allocation unit
  - The smaller the allocation unit, the larger the bitmap.
- Problem: allocation
  - When a new process arrives, the manager must find consecutive 0 bits in the map.
  - Searching a bitmap for a run of a given length is a slow operation.

```
1 0 1 0 1 0 0 0 1 0 1 0 0 1 0
```
Memory Management with Linked Lists

- Use a linked list of allocated and free memory segments (called hole)
  - Sorted by the address or by the size

Four neighbor combinations for the terminating process X
Memory Allocation Strategies

• **Best Fit**
  – Uses the hole whose size is equal to the need, or if none is equal, the hole that is larger but closest in size.
  – Problem: Creates small holes that can't be used.

• **First Fit**
  – Uses the first available hole whose size is sufficient to meet the need.
  – Problem: Creates average size holes.

• **Next Fit**
  – Minor variation of first fit: search from the last stopped place.
  – Problem: slightly worse performance than first fit.

• **Worst Fit**
  – Uses the largest available hole.
  – Problem: Gets rid of large holes making it difficult to run large programs.

• **Quick Fit**
  – Maintains separate lists of holes for some of the more common sizes requested. When a request comes for placement it finds the closest fit.
  – A very fast scheme, but merge is expensive. If merge is not done, memory will quickly fragment in a large number of holes into which no processes fit.
How Bad Is Fragmentation?

- Statistical arguments - Random sizes
- First-fit
- Given $N$ allocated blocks, $0.5*N$ blocks will be lost because of fragmentation
- Known as 50% RULE
Memory Management Problems

• Fixed partitions
  – Suffer from internal fragmentation
• Variable partitions
  – Suffer from external fragmentation
• Compaction
  – Suffer from overhead
• Overlays
  – Painful for programmers
• Swapping
  – Requires writing to disk sectors
Summary

• Basic Memory Management
• Swapping
• Variable Partitions
• Memory Management Problems