Memory Management

Jin-Soo Kim (jinsookim@skku.edu)
Computer Systems Laboratory
Sungkyunkwan University
http://csl.skku.edu
Today’s Topics

- Why is memory management difficult?

- Old memory management techniques:
  - Fixed partitions
  - Variable partitions
  - Overlays
  - Swapping

- Introduction to virtual memory
Memory Management (1)

- Goals
  - To provide a convenient abstraction for programming.
  - To allocate scarce memory resources among competing processes to maximize performance with minimal overhead.
  - To provide isolation between processes.

- Why is it so difficult?
Single/Batch Programming

- An OS with one user process
  - Programs use physical addresses directly.
  - OS loads job, runs it, unloads it.
Multiprogramming

- Need multiple processes in memory at once.
  - To overlap I/O and CPU of multiple jobs
  - Each process requires variable-sized and contiguous space.

- Requirements
  - Protection: restrict which addresses processes can use.
  - Fast translation: memory lookups must be fast, in spite of protection scheme.
  - Fast context switching: updating memory hardware (for protection and translation) should be quick.
Fixed Partitions (1)

Base register
0x2000

Virtual address
0x0362

0x2362

Partition 0
0x1000

Partition 1
0x2000

Partition 2
0x3000

Partition 3
0x4000

Partition 4
0x5000

Operating System
0

Operating System | Fall 2010 | Jin-Soo Kim (jinsookim@skku.edu)
Fixed Partitions (2)

- **Physical memory is broken up into fixed partitions**
  - Size of each partition is the same and fixed
  - the number of partitions = degree of multiprogramming
  - Hardware requirements: base register
    - Physical address = virtual address + base register
    - Base register loaded by OS when it switches to a process

- **Advantages**
  - Easy to implement, fast context switch

- **Problems**
  - **Internal fragmentation**: memory in a partition not used by a process is not available to other processes
  - **Partition size**: one size does not fit all
    - Fragmentation vs. fitting large programs
Fixed Partitions (3)

- **Improvement**
  - Partition size need not be equal.
  - Allocation strategies
    - Maintain a separate queue for each partition size
    - Maintain a single queue and allocate to the closest job whose size fits in an empty partition (first fit)
    - Search the whole input queue and pick the largest job that fits in an empty partition (best fit)
  - IBM OS/MFT
    (Multiprogramming with a Fixed number of Tasks)

<table>
<thead>
<tr>
<th>Partition</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partition 0</td>
<td>0x1000</td>
</tr>
<tr>
<td>Partition 1</td>
<td>0x2000</td>
</tr>
<tr>
<td>Partition 2</td>
<td>0x4000</td>
</tr>
<tr>
<td>Partition 4</td>
<td>0x8000</td>
</tr>
</tbody>
</table>
Variable Partitions (1)

Virtual address offset

Limit register
P1’s Limit

Base register
P1’s Base

<?

Yes

No

Yes

No

Protection fault

Partition 3

Partition 2

Partition 1

Partition 0

Operating System
Variable Partitions (2)

- Physical memory is broken up into variable-sized partitions
  - IBM OS/MVT
  - Hardware requirements: base register and limit register
    - Physical address = virtual address + base register
    - Base register loaded by OS when it switches to a process
  - The role of limit register: protection
    - If (physical address > base + limit), then raise a protection fault.

- Allocation strategies
  - First fit: Allocate the first hole that is big enough
  - Best fit: Allocate the smallest hole that is big enough
  - Worst fit: Allocate the largest hole
Variable Partitions (3)

- **Advantages**
  - No internal fragmentation
    - Simply allocate partition size to be just big enough for process
    - But, if we break the physical memory into fixed-sized blocks and allocate memory in unit of block sizes (in order to reduce bookkeeping), we have internal fragmentation.

- **Problems**
  - External fragmentation
    - As we load and unload jobs, holes are left scattered throughout physical memory
  - Solutions to external fragmentation:
    - Compaction
    - Paging and segmentation
Overlays (1)

- Overlays for a two-pass assembler
Overlays (2)

- **Overlays**
  - Keep in memory only those instructions and data that are needed at any given time.
  - Normally implemented by user

- **Advantages**
  - Needed when a process is larger than the amount of memory allocated to it.
  - No special support needed from operating system.

- **Problems**
  - Programming design of overlay structure is complex.
Swapping (1)

1. swap out

2. swap in

- Operating system
- User space
- Main memory
- Process $P_1$
- Process $P_2$
- Backing store
Swapping (2)

- **Swapping**
  - A process can be swapped temporarily out of memory to a backing store and then brought back into memory later for continued execution.
  - **Backing store**
    - Fast disk large enough to accommodate copies of all memory images for all users
    - Must provide direct access to these memory images
  - **Major part of swap time is transfer time.**
    - Directly proportional to the amount of memory swapped.
  - **Swapping a process with a pending I/O**
    - Do not swap a process with pending I/O
    - Execute I/O operations only into OS buffers
Virtual Memory (1)

- Example

```c
#include <stdio.h>

int n = 0;

int main ()
{
    printf ("&n = 0x%08x\n", &n);
}

% ./a.out
&n = 0x08049508
% ./a.out
&n = 0x08049508
```

- What happens if two users simultaneously run this application?
Virtual Memory (VM)

- Use virtual addresses for memory references.
  - Large and contiguous
- CPU performs address translation at run time.
  - From a virtual address to the corresponding physical address
- Physical memory is dynamically allocated or released on demand.
  - Programs execute without requiring their entire address space to be resident in physical memory.
  - Lazy loading
- Virtual addresses are private to each process.
  - Each process has its own isolated virtual address space.
  - One process cannot name addresses visible to others.
Virtual Memory (3)

- **Virtual addresses**
  
  - To make it easier to manage memory of multiple processes, make processes use virtual addresses (logical addresses)
    - Virtual addresses are independent of the actual physical location of data referenced.
    - OS determines location of data in physical memory
    - Instructions executed by the CPU issue virtual addresses.
    - Virtual addresses are translated by hardware into physical addresses (with help from OS).
    - The set of virtual addresses that can be used by a process comprises its **virtual address space**.
  
  - Many ways to translate virtual addresses into physical addresses...
Virtual Memory (4)

- **Advantages**
  - Separates user’s logical memory from physical memory.
    - Abstracts main memory into an extremely large, uniform array of storage.
    - Frees programmers from the concerns of memory-storage limitations.
  - Allows the execution of processes that may not be completely in memory.
    - Programs can be larger than physical memory.
    - More programs could be run at the same time.
    - Less I/O would be needed to load or swap each user program into memory.
  - Allows processes to easily share files and address spaces.
  - Provides an efficient mechanism for protection and process creation.
Virtual Memory (5)

- **Disadvantages**
  - Performance!!!
    - In terms of time and space

- **Implementation**
  - Paging
  - Segmentation