CMPSCI 377 Operating Systems **Fall 2005**

Lecture 10: October 18

Lecturer: Vitaliy Lvin *Scribes: Szymon Kaflinski and Matthew Tassinari*

Last time: Read/Write Locks, Monitors, Condition Variables.

Today we're going to learn about Memory Management:

- Terminology
- Addresses
- Uniprogramming
- Multiprogramming
- Relocation
- Allocation
- Swapping

10.1 Terminology

We know that executable programs are stored on disk. When programs are executed the OS loads them from disk into memory. While the program executes it grabs instructions from the allocated memory for that program. It also reads and writes data for that program from memory. The following is a list of different terms used in memory management.

Segment:

- a memory spot assigned to process

Physical address:

- real address in memory

Virtual address:

- relative address to start of the current processs address space
- by use of a look up table these addresses are assigned

10.2 Addresses

Now that we know what addresses are and the different types, where do they come from? The source of an address is made available for data and instructions in there different manners.

Compile-time:

- an exact physical location in memory is assigned starting from fixed position
- prereserved memory chunk and always the same chuck

Load-time:

- lets the OS determine processs starting position
- fixes up addresses and can run in different parts

Execution time:

- lets the OS place address anywhere in physical memory
- this solution cause the least problems

10.3 Uniprogramming

This is a model from the past that allows only one program at a time. In this case the memory management is very simple.

The OS gets the highest memory spot Only load one process at a time

- load at address 0
- executes in contiguous memory

The compiler will only generate physical addresses:

- maximum address = memory size OS size (highest location)
- OS is protected from running process by checking addresses

IMPORTANT: No overlap of I/O (see lecture slides for example diagram)

10.4 Multiprogramming

As the name implies this model allows for multiple programs to run at once. In order to have a correct memory management system these three properties must hold true:

Transparency:

- processes are not aware memory is shared
- give the process no constraints on physical memory

Safety:

- no processes overwrites each other
- most important the OS

Efficiency:

- performance not degraded due to sharing
- it needs to be fast

10.5 Relocation

- the Operating System gets loaded into high memory.
- processes start at memory address 0.
- processes are allocated contiguous memory with base and limit addresses.

There are two types of Relocation: Static Relocation: At process load, the process' addresses are calculated. Processes do not move after relocation. Dynamic Relocation: Hardware adds relocation/base register to virtual address to get physical address. Hardware compares address with limit register.

There are some pros and cons of relocation:

Pros

- easy movement of processes during execution
- OS allows a process to grow
- simple, fast hardware (2 registers add and compare)

Cons

- slows system due to constant calculation
- can't share memory between processes
- process limited to physical memory size - OS
- limited multiprogramming
- complicated memory management

Relocation Properties: Transparency: processes unaware of sharing, Safety: each memory ref checked, Efficiency: checks fast if in hardware, process growth can cause slow movement

10.6 Allocation

As processes start, grow and end, the OS must track available and in-use memory. This can leave holes which the OS must determine how to manage.

Memory allocation policies:

- First Fit: use first hole in which process fits.
- Best Fit: use smallest hole that process will fit in.
- Worst-Fit: use the largest hole.

10.7 Fragmentation/ Compaction

Fragmentation: Memory unavailable for allocation but unused by processes. Internal Fragmentation:

- many small holes caused by loading/ unloading.
- no contiguous space large enough for processes.

External Fragmentation:

- space allocated to processes but unused.

Compaction: Eliminate holes from fragmentation.

Issues:

- amount of memory used.
- block size.

10.8 Swapping

Copy process to disk and release all memory.

- must reload processes when they become active.
- in static relocation they must go back to the same location.
- dynamic relocation is easier

If swapping is supported, compaction simple

- not supported in modern OSs

10.9 Summary

In order for proceess to execute they have the reside in memory.

Generally they use a virtual address.

- Translated to physical addresses before accessing memory

Segmentation:

- Allows processes to share memory
- Expensive to grow over time

Swapping:

- Total being used by all can exceed main memory
- Increases context switch time