Memory Management
CS 111
Operating Systems
Peter Reiher

Outline

- What is memory management about?
- Memory management strategies:
  - Fixed partition strategies
  - Dynamic domains
  - Buffer pools
  - Garbage collection
  - Memory compaction

Memory Management

- Memory is one of the key assets used in computing
- In particular, memory abstractions that are usable from a running program
  - Which, in modern machines, typically means RAM
- We have a limited amount of it
- Lots of processes want to use it
- How do we manage its use?

What Is Memory Used For?

- Anything that a program needs to access
  - Except control and temporary values, which are kept in registers
- The code
  - To allow the process to execute instructions
- The stack
  - To keep track of its state of execution
- The heap
  - To hold dynamically allocated variables

Other Uses of Memory

- The operating system needs memory itself
- For its own code, stack, and dynamic allocations
- For I/O buffers
- To hold per-process control data
- The OS shares the same physical memory that user processes rely on
- The OS provides overall memory management

Aspects of the Memory Management Problem

- Most processes can't perfectly predict how much memory they will use
- The processes expect to find their existing data when they need it where they left it
- The entire amount of data required by all processes may exceed physical memory
- Switching between processes must be fast
  - So you can't much delay for copying data from one place to another
- The cost of memory management itself must not be too high

Memory Management Strategies

- Fixed partition allocations
- Dynamic domains
- Paging
- Virtual memory
- We'll talk about the last two in the next class

Fixed Partition Allocation

- Pre-allocate partitions for *n* processes
  - Usually one partition per process
    - So *n* partitions
  - Reserving space for largest possible process
- Partitions come in one or a few set sizes
- Very easy to implement
  - Common in old batch processing systems
  - Allocation/deallocation very cheap and easy
- Well suited to well-known job mix

Memory Protection and Fixed Partitions

- Need to enforce the boundaries of each partition
- To prevent one process from accessing another's memory
- Could use hardware similar to domain registers for this purpose
- On the flip side, hard to arrange for shared memory
  - Especially if only one segment per process

Problems With Fixed Partition Allocation

- Presumes you know how much memory will be used ahead of time
- Limits the number of processes supported to the total of their memory requirements
- Not great for sharing memory
- *Fragmentation* causes inefficient memory use

Fragmentation

- A problem for all memory management systems
  - Fixed partitions suffer it especially badly
- Based on processes not using all the memory they requested
- As a result, you can't provide memory for as many processes as you theoretically could

Fragmentation Example

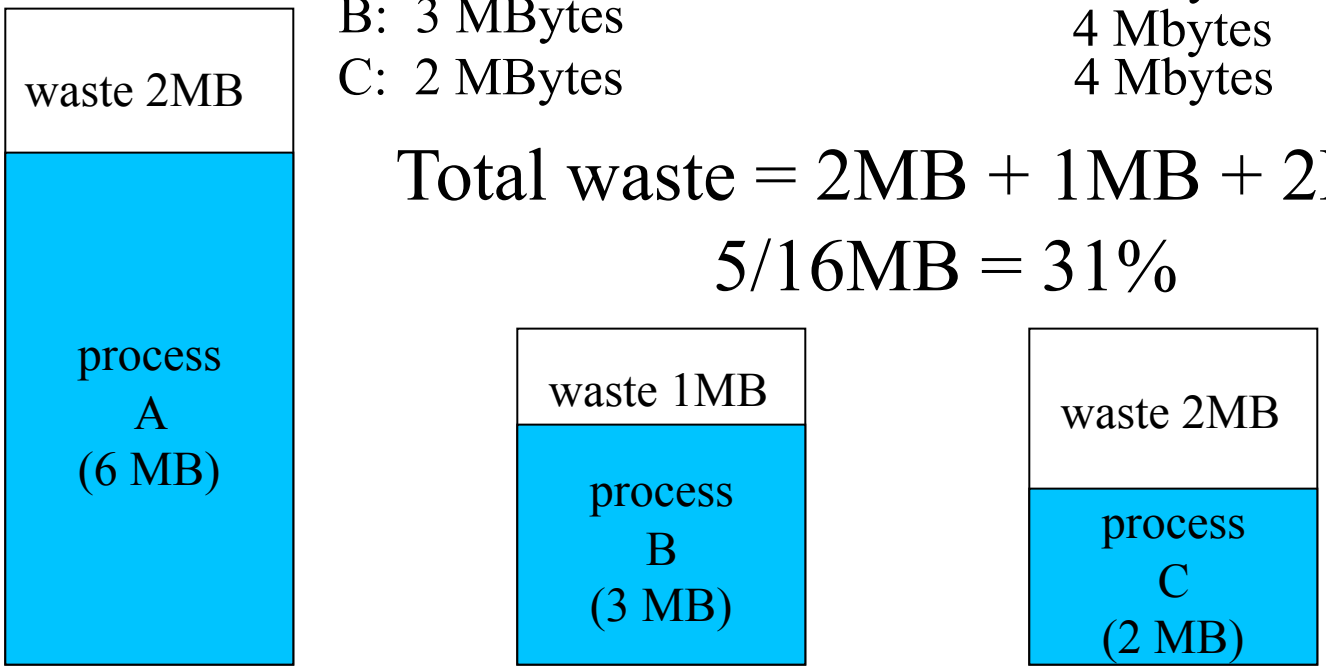
Let's say there are three processes, A, B, and C

Their memory requirements:

A: 6 MBytes
B: 3 MBytes
C: 2 MBytes

Available partition sizes:

8 Mbytes
4 Mbytes
4 Mbytes



Total waste = 2MB + 1MB + 2MB =
5/16MB = 31%

Internal Fragmentation

- Fragmentation comes in two kinds:
  - Internal and external
- This is an example of *internal fragmentation*
  - We'll see external fragmentation later
- Wasted space in fixed sized blocks
  - The requestor was given more than he needed
  - The unused part is wasted, can't be used for others
- Internal fragmentation can occur whenever you force allocation in fixed-sized chunks

More on Internal Fragmentation

- Internal fragmentation is caused by a mismatch between
  - The chosen sizes of a fixed-sized blocks
  - The actual sizes that programs use
- Average waste: 50% of each block
- Overall waste reduced by multiple sizes
  - Suppose blocks come in sizes S1 and S2
  - Average waste = ((S1/2) + (S2 - S1)/2)/2

Multiple Fixed Partitions

- You could allow processes to request multiple partitions
  - Of a single or a few sizes
- Doesn't really help the fragmentation problem
  - Now there were more segments to fragment
  - Even if each contained less memory

Summary of Fixed Partition Allocation

- Very simple
- Inflexible
- Subject to a lot of internal fragmentation
- Not used in many modern systems
  - But a possible option for special purpose systems, like embedded systems
  - Where we know exactly what our memory needs will be