Principles of Operating Systems

CS 446/646

3. Memory Management

René Doursat

Department of Computer Science & Engineering
University of Nevada, Reno

Fall 2005

Principles of Operating Systems

CS 446/646

0. Course Presentation
1. Introduction to Operating Systems
2. Processes
3. Memory Management
4. CPU Scheduling
5. Input/Output
6. File System
7. Case Studies

Principles of Operating Systems

CS 446/646

3. Memory Management

a. Goals of Memory Management

b. Partitioning

c. Linking & Loading

d. Simple Paging & Segmentation

e. Virtual Memory

f. Page Replacement Algorithms

Principles of Operating Systems

CS 446/646

3. Memory Management

a. **Goals of Memory Management**
   - ✓ How to distribute multiple processes in memory?
   - ✓ Relocation of address references
   - ✓ Protection & sharing of address spaces
   - ✓ Logical vs. physical organization

b. **Partitioning**

c. **Linking & Loading**

d. **Simple Paging & Segmentation**

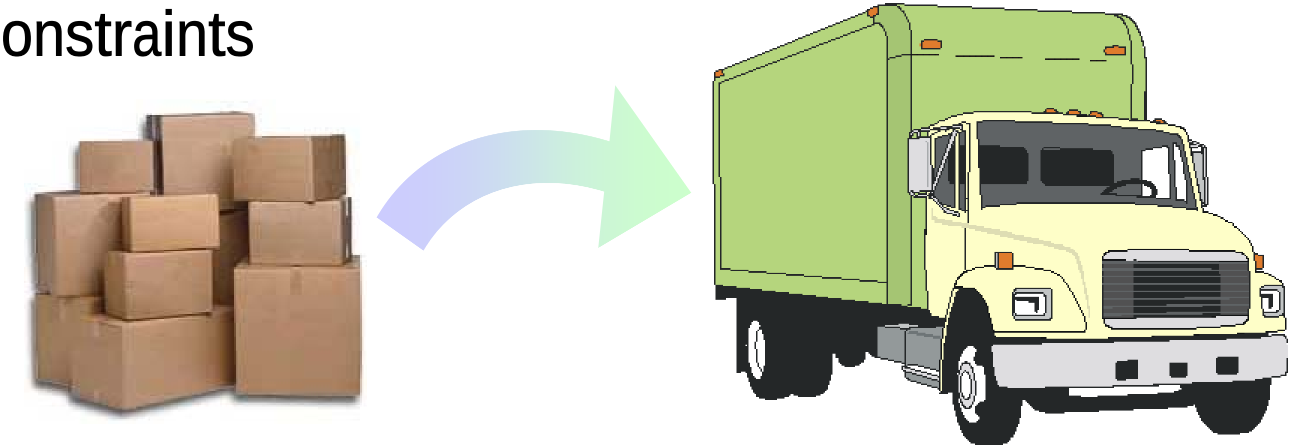
e. **Virtual Memory**

f. **Page Replacement Algorithms**

3.a Goals of Memory Management

How to distribute multiple processes in memory?

- The O/S must fit multiple processes in memory
  - ✓ memory needs to be subdivided to accommodate multiple processes
  - ✓ memory needs to be allocated to ensure a reasonable supply of ready processes so that the CPU is never idle
  - ✓ memory management is an **optimization** task under **constraints**



Fitting processes into memory is like fitting boxes into a fixed amount of space

3.a Goals of Memory Management

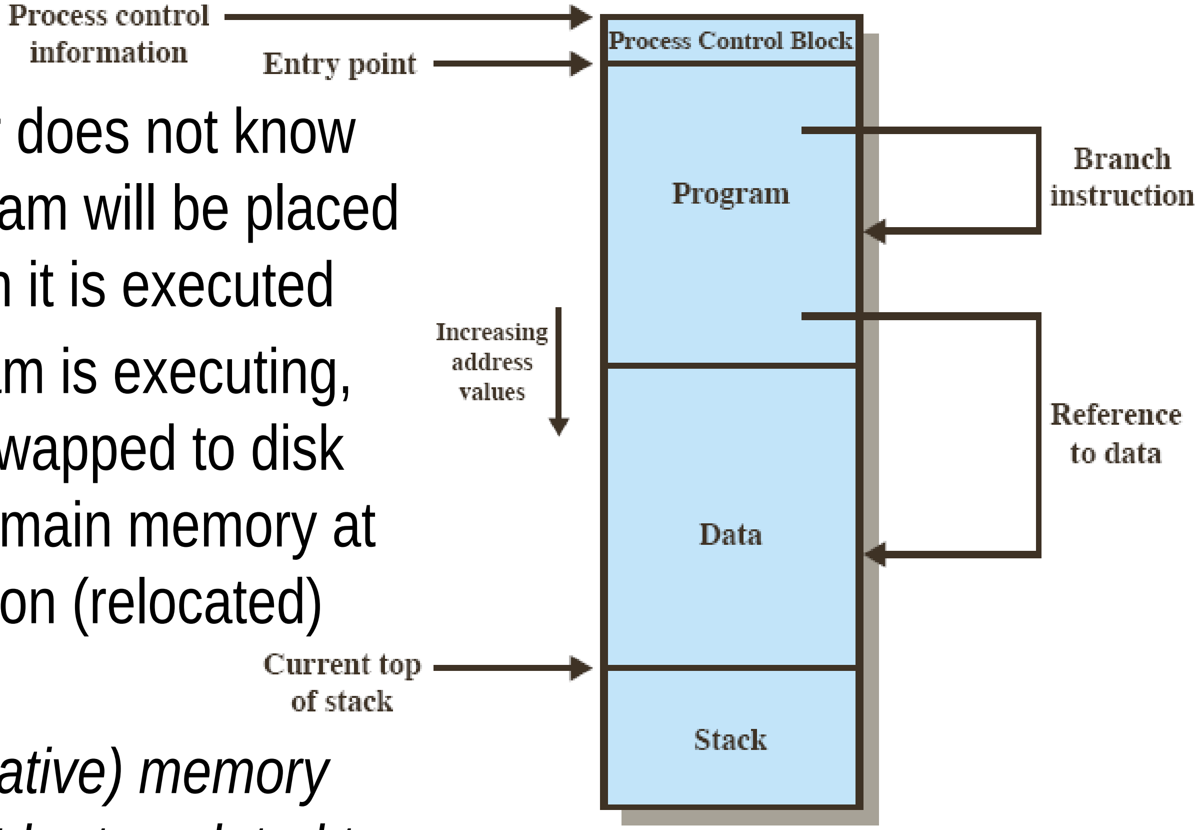
How to distribute multiple processes in memory?

- Memory management must satisfy various requirements
  - ✓ relocation of address references
    - must translate memory references to physical addresses
  - ✓ protection of memory spaces
    - forbid cross-process references
  - ✓ sharing of memory spaces
    - allow several processes to access a common memory area
  - ✓ logical organization (of programs)
    - programs are broken up into independent modules
  - ✓ physical organization (of memory)
    - fit multiple programs and modules in physical memory

3.a Goals of Memory Management

Relocation of address references

- Relocation of address references
  - ✓ the programmer does not know where the program will be placed in memory when it is executed
  - ✓ while the program is executing, it may also be swapped to disk and returned to main memory at a different location (relocated)
  - → *thus, logical (relative) memory references must be translated to physical (absolute) addresses*



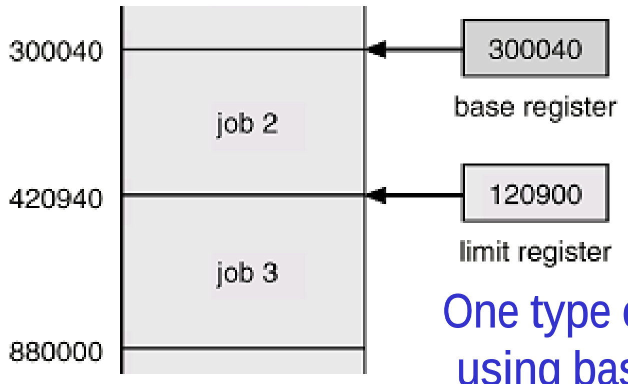
Stallings, W. (2004) *Operating Systems: Internals and Design Principles (5th Edition).*

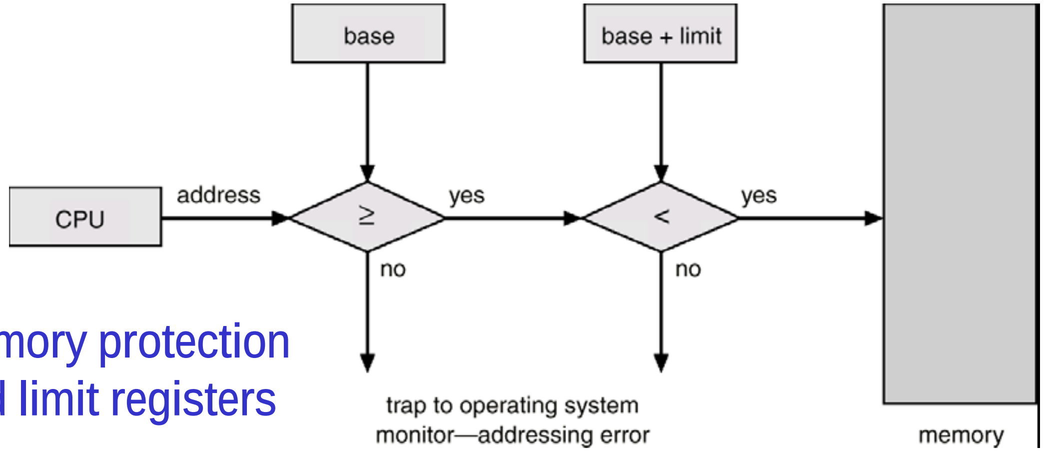
Process addressing requirements

3.a Goals of Memory Management

Protection of address spaces

- ➢ Protection of address spaces
  - ✓ processes must not use memory locations in other processes
  - ✓ addressing must be checked at run time, as it is impossible to check physical addresses at compile time
  - ✓ however, the operating system cannot anticipate all of the (calculated) memory references a program will make
  - → *thus, fine-level protection is ultimately carried out in hardware*





One type of memory protection using base and limit registers

Silberschatz, A., Galvin, P. B. and Gagne. G. (2003) *Operating Systems Concepts with Java (6th Edition).*

3.a Goals of Memory Management

Sharing of address spaces

- Sharing of address spaces
  - ✓ conversely, it should be possible to allow several processes to access the same portion of memory
    - for example, processes executing the same program can save resources by sharing the same copy of code in memory
    - also, processes cooperating on some task may need access to the same data structure
  - → *we will see that mechanisms supporting relocation also support sharing capabilities*

3.a Goals of Memory Management

Logical organization

- Logical organization (of programs)
  - ✓ the linear 1-D organization of memory does not reflect the way programs are typically constructed
  - ✓ large programs are often organized into modules and the O/S should be able to handle modular programs, so that:
    - ▪ modules can be written and compiled independently
    - ▪ different degrees of protection can be given to different modules: read-only, execute-only, read-and-write, etc.
    - ▪ modules can be shared among processes
  - → *segmentation is a memory management technique that supports modularization*

3.a Goals of Memory Management

Physical organization

- Physical organization (of memory)
  - ✓ two-level scheme
    - main memory: fast access, high cost, volatile
    - secondary memory: slow access, cheaper, long-term storage
  - → *thus, a major O/S concern is the flow of information between main and secondary memory (it should not be a user concern)*

Principles of Operating Systems

CS 446/646

3. Memory Management

a. **Goals of Memory Management**
  - ✓ How to distribute multiple processes in memory?
  - ✓ Relocation of address references
  - ✓ Protection & sharing of address spaces
  - ✓ Logical vs. physical organization

b. **Partitioning**

c. **Linking & Loading**

d. **Simple Paging & Segmentation**

e. **Virtual Memory**

f. **Page Replacement Algorithms**

Principles of Operating Systems

CS 446/646

3. Memory Management

a. Goals of Memory Management

b. **Partitioning**

- ✓ Fixed partitioning: shelving the boxes
- ✓ Dynamic partitioning: stacking the boxes
- ✓ “Buddy system”: splitting & merging the shelves

c. **Linking & Loading**

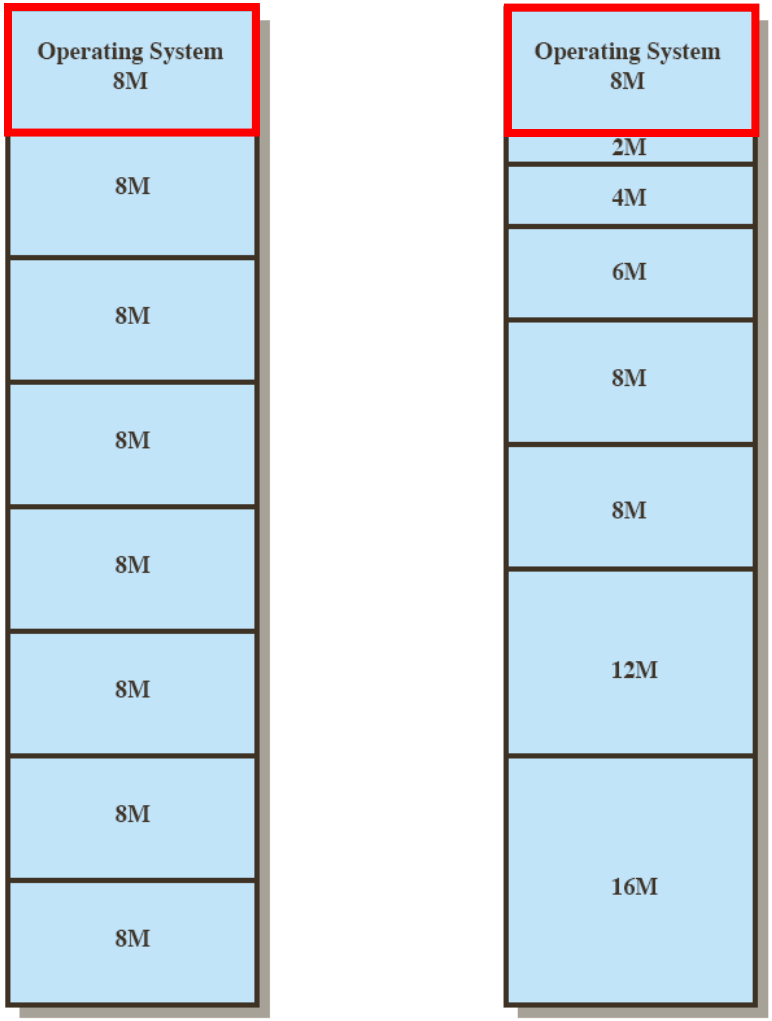
d. **Simple Paging & Segmentation**

e. **Virtual Memory**

f. **Page Replacement Algorithms**

3.b Partitioning

Fixed partitioning: shelving the boxes



(a) Equal-size partitions

(b) Unequal-size partitions

Stallings, W. (2004) *Operating Systems: Internals and Design Principles (5th Edition).*

Example of fixed partitioning of a 64MB memory (the "shelves")

3.b Partitioning

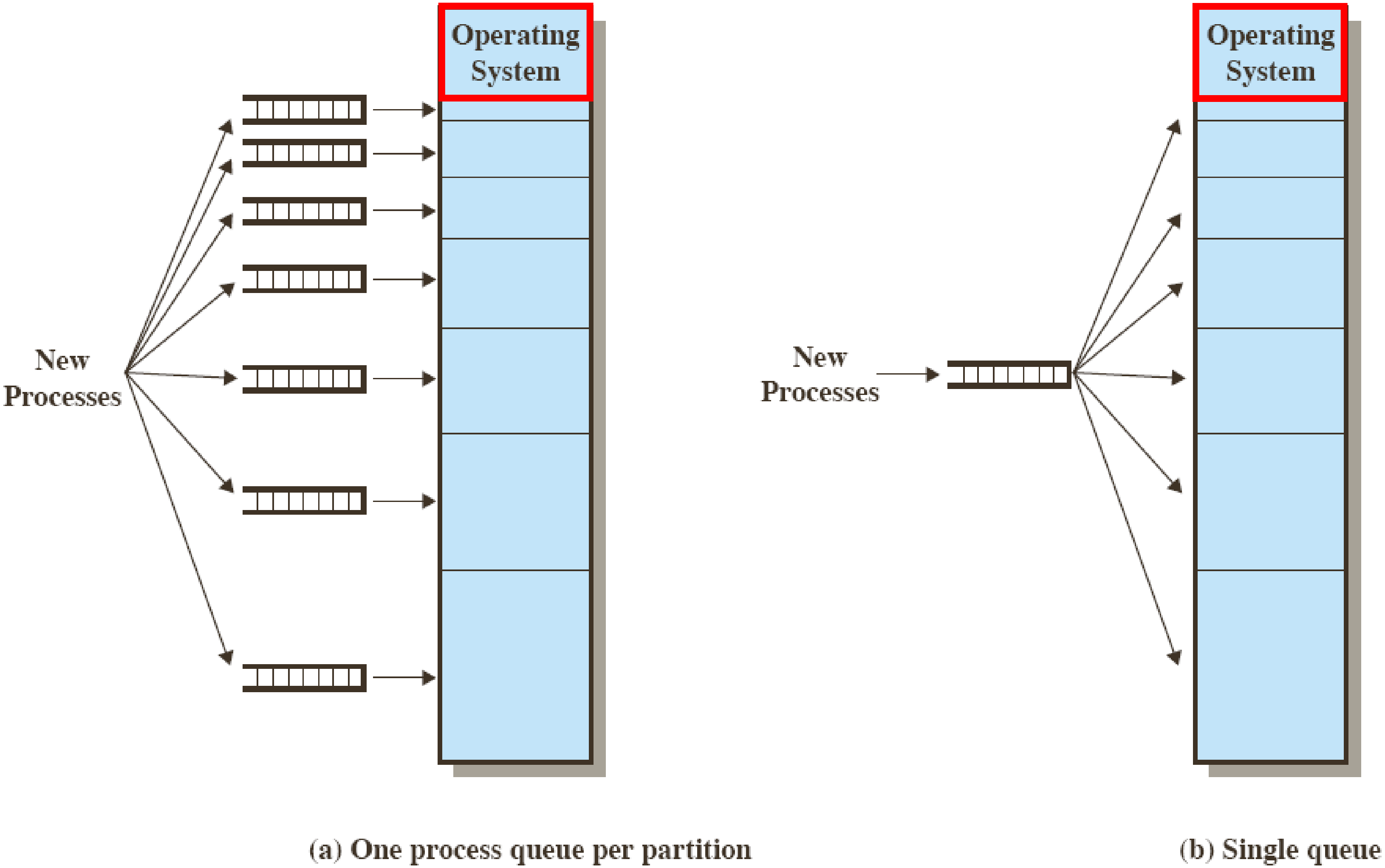
Fixed partitioning: shelving the boxes

- Fixed partition establishes fixed boundaries in memory
  - 👍 the oldest and simplest scheme to manage memory space
    - any process whose size is less than or equal to a partition size can be loaded into an available partition
    - if all partitions are full, the operating system can swap a process out of a partition
  - 👎 also the least adequate scheme
    - larger programs may not fit in any of the partitions, so the programmer must design “overlaying” modules
    - memory use is very inefficient: even small programs occupy entire partitions, thus wasting space internal to the partitions

    → this waste of space is called **internal fragmentation**

3.b Partitioning

Fixed partitioning: shelving the boxes



(a) One process queue per partition

(b) Single queue

Stallings, W. (2004) *Operating Systems: Internals and Design Principles (5th Edition).*

Placement algorithms for unequal-size fixed partitioning

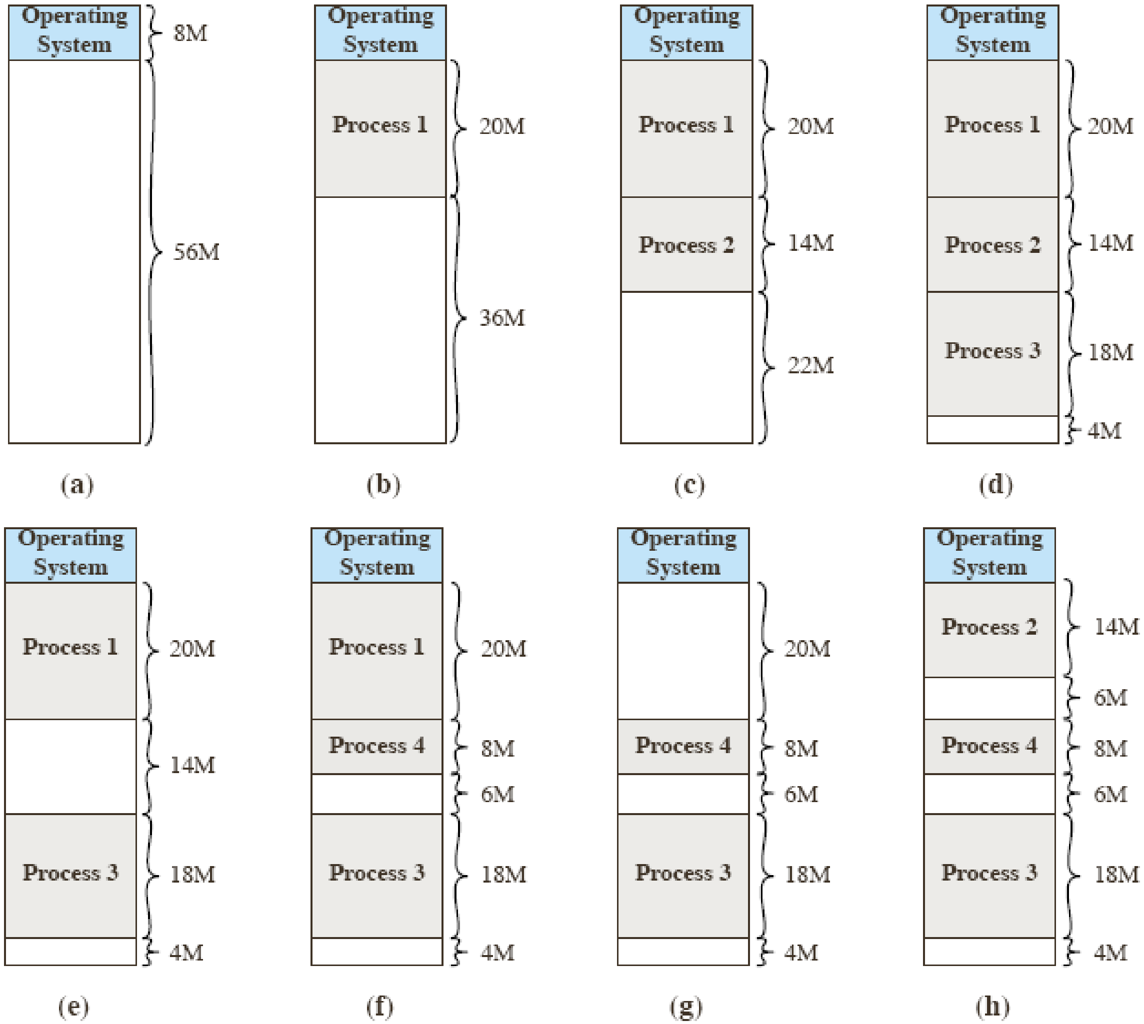
3.b Partitioning

Fixed partitioning: shelving the boxes

- Placement algorithms for fixed partitioning
  - ✓ equal-size partitions
    - because all partitions are of equal size, it does not matter which partition is used
    - → no special algorithm is needed
  - ✓ unequal-size partitions
    - per-partition queue: to minimize internal fragmentation, processes must wait for a partition that best fits their size
    - global queue: however, doing so needlessly prevents a process from running while another (bigger) partition might be available
    - → tradeoff between wasting space and wasting time

3.b Partitioning

Dynamic partitioning: stacking the boxes



The effect of dynamic partitioning

Stallings, W. (2004) *Operating Systems: Internals and Design Principles (5th Edition).*

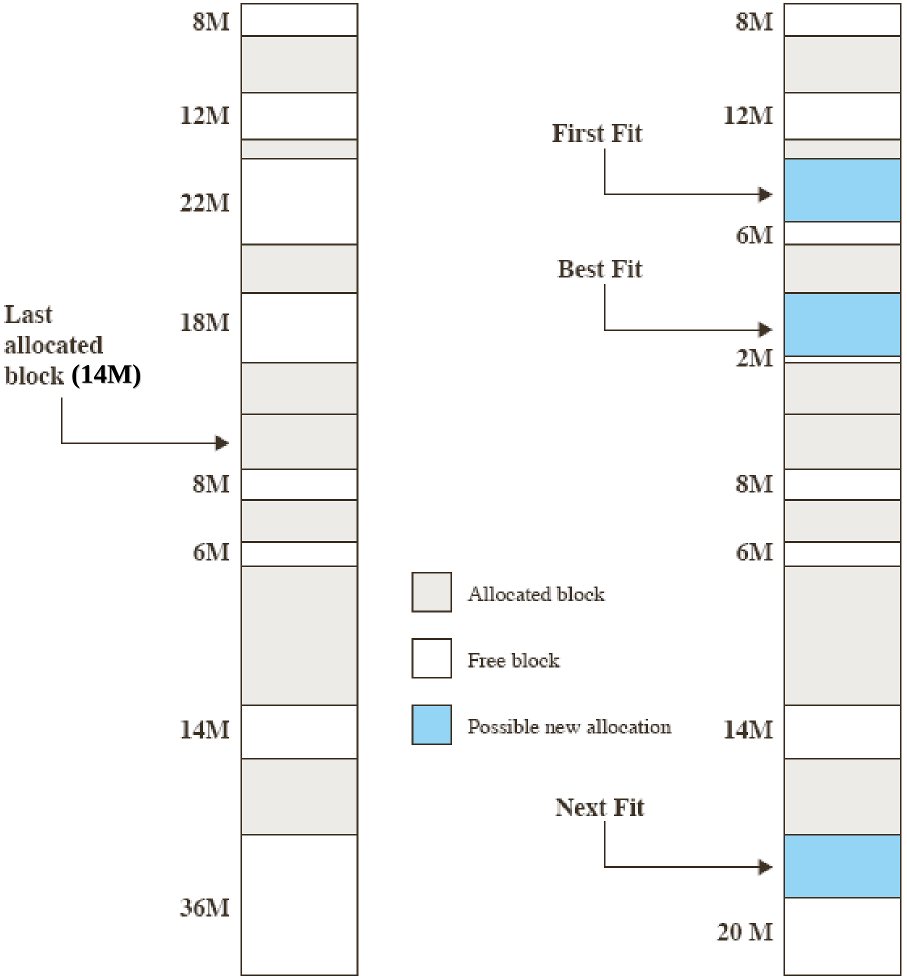
3.b Partitioning

Dynamic partitioning: stacking the boxes

- ➢ Dynamic partitioning stacks processes contiguously
  - 👍 also an old and relatively simple allocation scheme
    - partitions are now of variable length and number
    - a process is allocated exactly as much memory as required
  - 👎 but also inadequate for today’s standards
    - stacking processes will not prevent gaps as processes are continuously swapped in and out of memory
    - → this is called **external fragmentation**
    - O/S **compaction** routines can shift processes from time to time, but this is time-consuming in read/write operations and relocation (re-translating references)

3.b Partitioning

Dynamic partitioning: stacking the boxes



(a) Before

(b) After

Stallings, W. (2004) *Operating Systems: Internals and Design Principles (5th Edition)*.

Before and after allocation of a 16M block under dynamic partitioning

3.b Partitioning

Dynamic partitioning: stacking the boxes

- Placement algorithms minimize external fragmentation
  - in order to minimize costly compaction (“garbage-collection” of fragmentation), position the processes cleverly; alternatives are:
  - **best-fit** placement
    - chooses the block that is closest in size to the request, so as to leave the smallest amount of fragmentation
  - **first-fit** placement
    - scans the memory from the beginning and chooses the first available block that is large enough
  - **next-fit** placement
    - scans the memory from the last placement and chooses the next available block that is large enough

3.b Partitioning

Dynamic partitioning: stacking the boxes

- ➢ Placement algorithms: comparative results
  - ✓ while performance depends on the exact sequence of process requests and sizes, statistical conclusions can be reached:
  - 👎 **best-fit** placement
    - ▪ paradoxically, the worst performer! it quickly litters memory with small fragments and requires compaction frequently
  - 👍 **first-fit** placement
    - ▪ the best and fastest
  - ✌ **next-fit** placement
    - ▪ the runner-up: slightly worse than first-fit, because it spreads fragmentation more evenly (whereas first-fit has a tendency to preserve big blocks at the end of memory)

3.b Partitioning

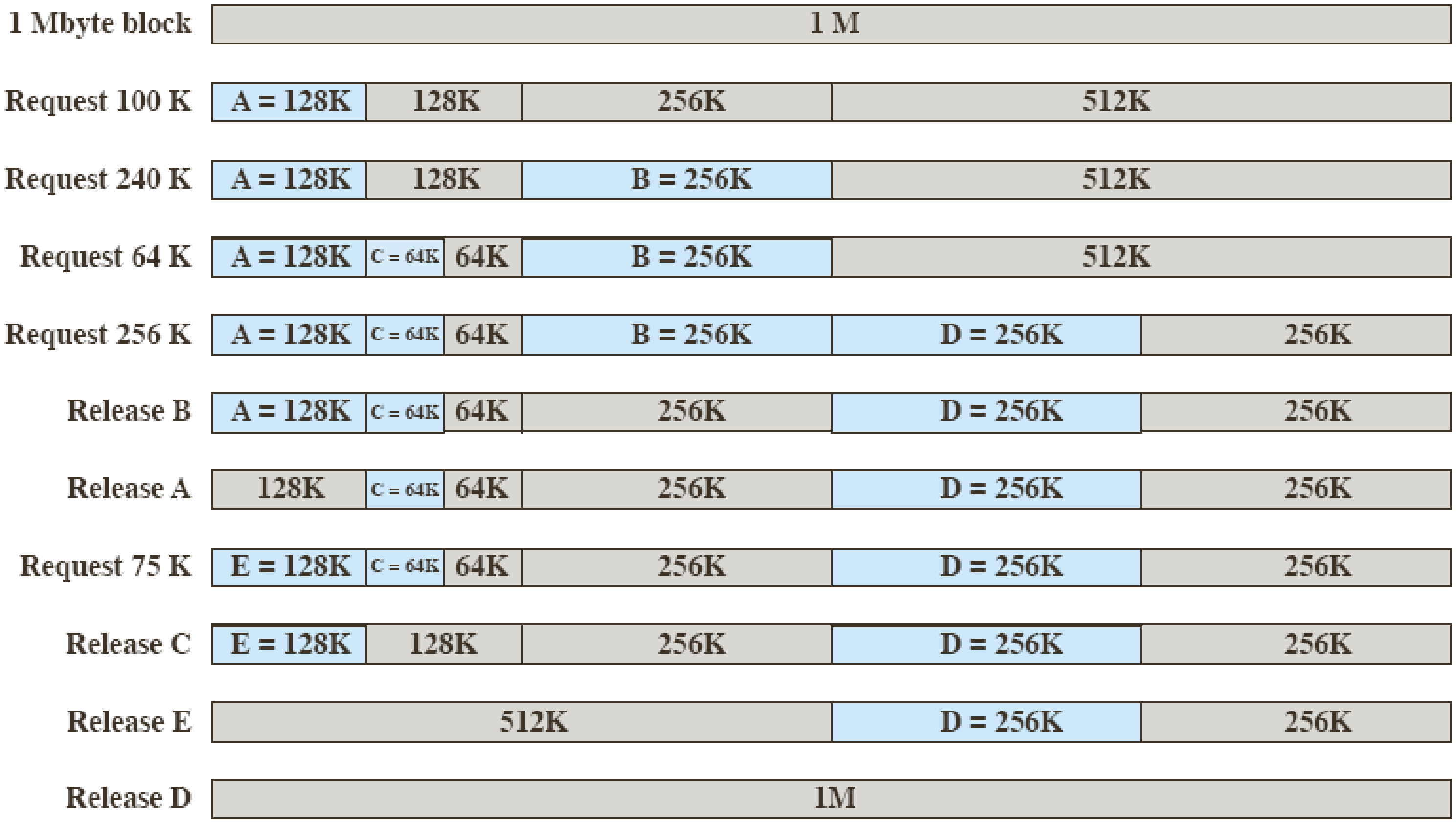
Dynamic partitioning: stacking the boxes

- Summary: fixed vs. dynamic partitioning
  - ✓ both fixed and dynamic partitioning schemes have drawbacks
  - ✓ fixed partitioning
    - limits the number of active processes
    - wastes space through internal fragmentation
  - ✓ dynamic partitioning
    - more complex to maintain
    - wastes space through external fragmentation
    - requires the overhead of compaction

3.b Partitioning

“Buddy system”: splitting & merging the shelves

- The buddy system: splitting and coalescing



Example of buddy system

Stallings, W. (2004) *Operating Systems: Internals and Design Principles (5th Edition).*

Principles of Operating Systems

CS 446/646

3. Memory Management

a. Goals of Memory Management

b. **Partitioning**

- ✓ Fixed partitioning: shelving the boxes
- ✓ Dynamic partitioning: stacking the boxes
- ✓ “Buddy system”: splitting & merging the shelves

c. **Linking & Loading**

d. **Simple Paging & Segmentation**

e. **Virtual Memory**

f. **Page Replacement Algorithms**