

CIS 3207 - Operating Systems

Contiguous Memory Allocation

Professor Qiang Zeng

Spring 2018



Previous class...



- Uniprocessor policies
 - FCFS, Shortest Job First
 - Round Robin
 - Multilevel Feedback Queue
- Multiprocessor policies
 - A MFQ per processor
 - A process has affinity with a processor
 - Exception: idle cores can steal processes




Background

- Main memory and registers are the only storage that a CPU can access directly
- Program must be brought (from disk) into memory for it to be run




Outline

- Fixed partitions
- Dynamic partitions
- Buddy system



Contiguous allocation:
Each process occupies a contiguous
memory region in the physical memory

- Segmentation
- Paging

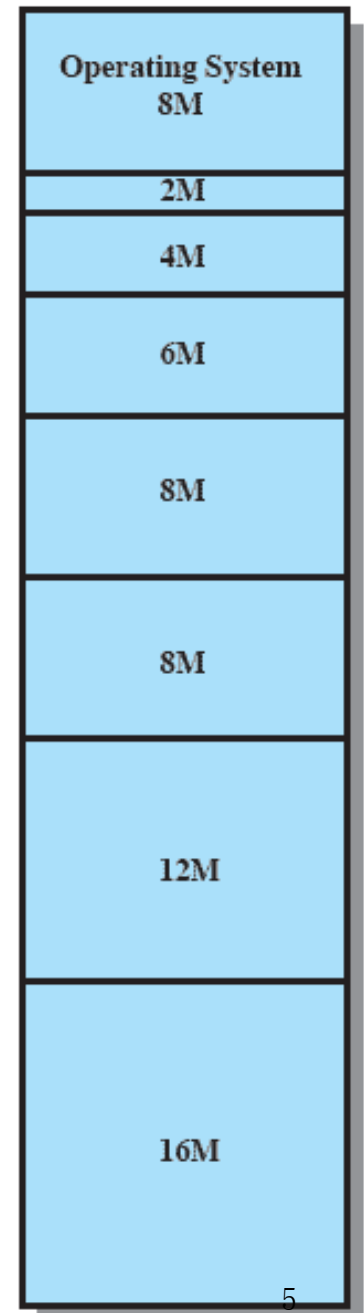


Non-contiguous allocation:
Each process occupies multiple memory regions
scattered in the physical memory.



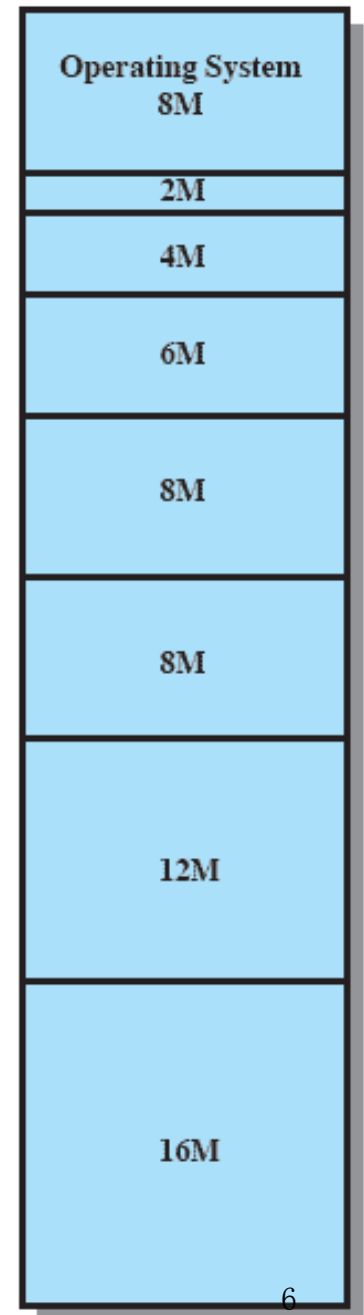
Fixed partitions

- The bounds of each partition are *fixed/predefined*
- Disadvantages:
 - Cause Internal Fragmentation when the allocated space is larger than the need
 - What is “**fragmentation**”? Small useless chunks
 - E.g., when you put a 13M process in the 16M partition, 3M space is wasted and it is called **internal fragmentation**
 - *Internal*: the wasted space is inside allocated space
 - The number of active processes is limited
- **Analogy**: street parking with meters



Fixed partitions - questions

- If 8M partitions are all used, where do you place a 7M process, and what is the size of the internal fragmentation?
 - The 12M partition is the best choice
 - The internal fragmentation is 5M
- How to resolve the severe internal fragmentation?
 - Dynamic partitions

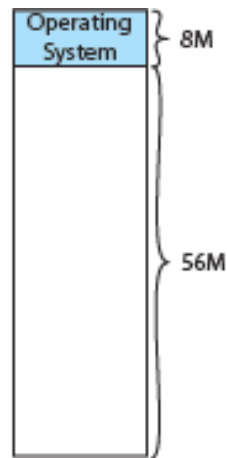


Dynamic partitions

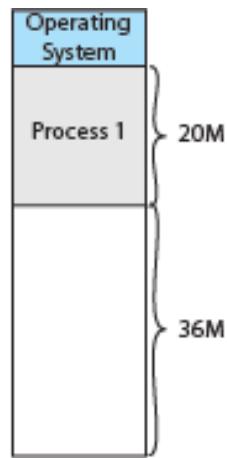
- Process is allocated exactly the memory it requires
- The partitions are dynamic: the number and locations of partitions are not fixed
- **Analogy:** street parking without meters



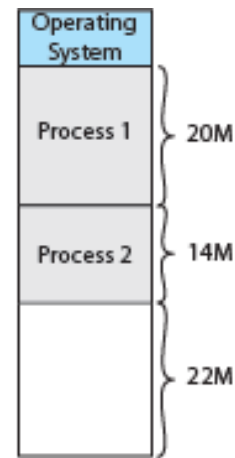
Dynamic partitions - example



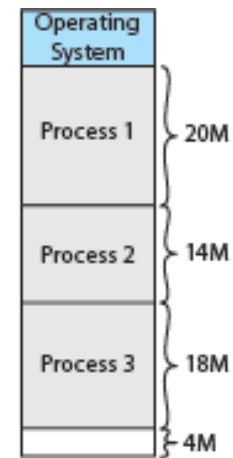
(a)



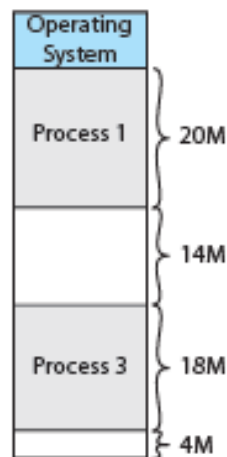
(b)



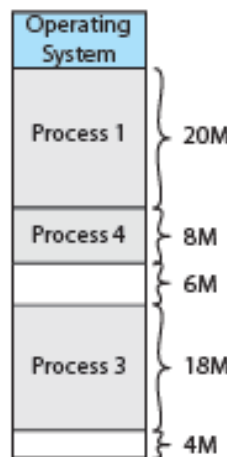
(c)



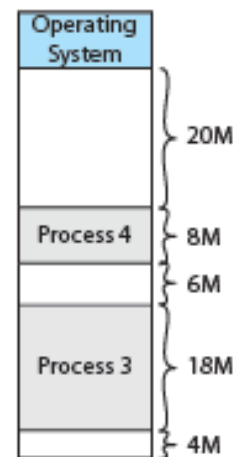
(d)



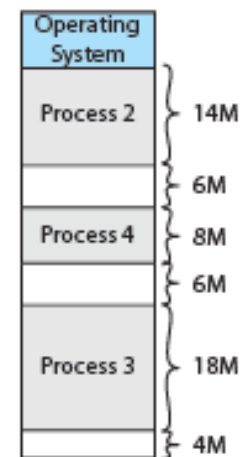
(e)



(f)



(g)



(h)



Dynamic partition - disadvantage

- External fragmentation
 - Their total size is large enough to satisfy a request, but they are not contiguous, so cannot be used to service the request
 - *External*: fragmentation is outside allocated space
- Solution: **Compaction**
 - OS shifts processes so that they are contiguous; thus, free memory is together in one block
 - But, program execution must be paused for relocation; waste CPU time



Placement Algorithms

- When there is more than one free block of memory of sufficient size, the system must decide which free block to allocate.

Best-fit

- chooses the block that is closest in size to the request

First-fit

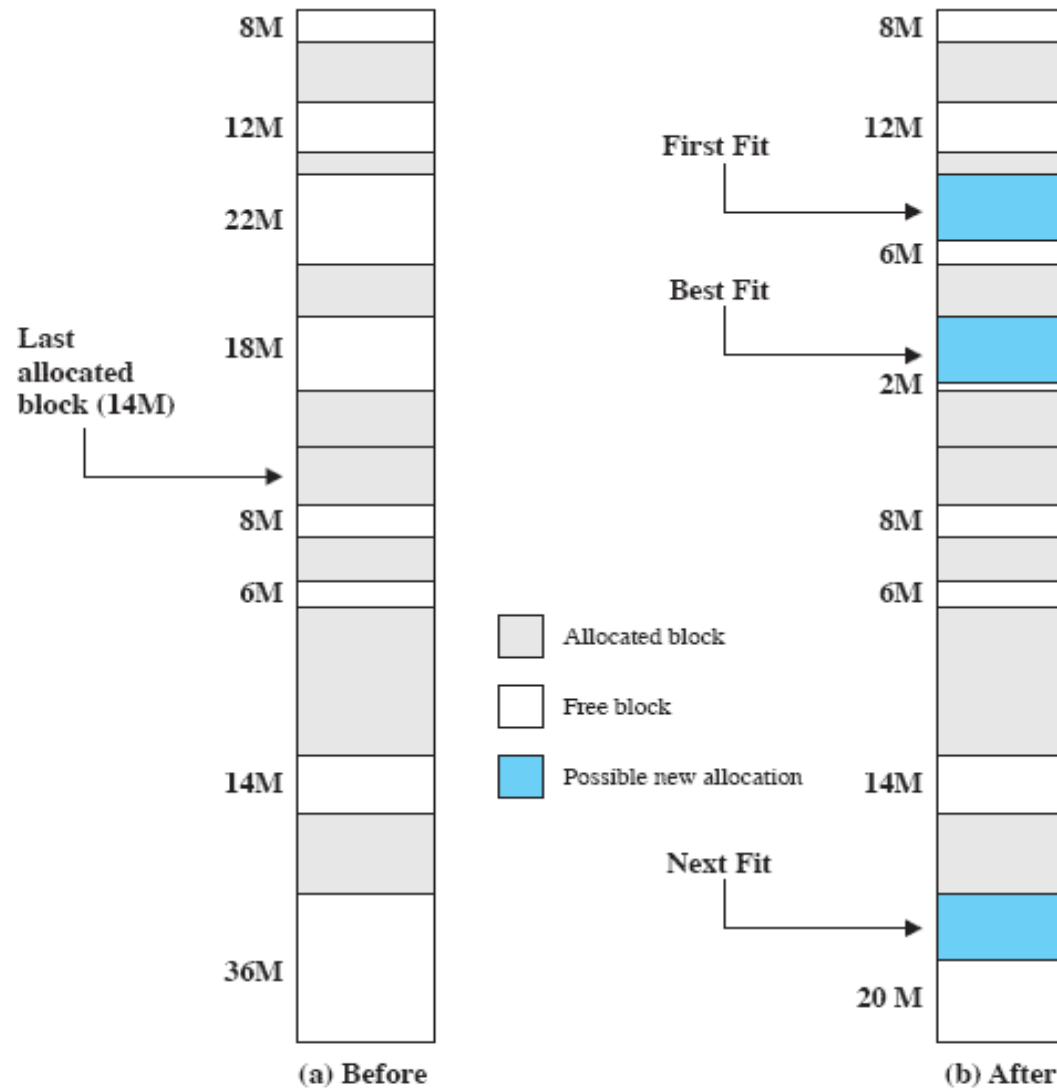
- scan memory from the beginning and chooses the first available block that is large enough

Next-fit

- scan memory from the location of the last placement and chooses the next available block that is large enough



Placement Algorithms - Example



Assume a 16M partition is requested

Questions

- Does Fixed Partitions Allocation have external fragmentation?
 - Zero
 - There are no small useless chunks
- Does Dynamic Partitions Allocation have internal fragmentation?
 - Zero
 - The allocated partition size is as needed



Buddy System (or, Buddy Memory Management)

- **Fixed partitions** cause zero external fragmentation but severe internal fragmentation, while **dynamic partitions** cause zero internal fragmentation but severe external fragmentation
- Buddy System is between the two
 - It causes acceptable internal and external fragmentation, and has good overall performance
- Three properties
 - Split-based allocation
 - **Freelists**-based implementation
 - Coalescing-buddy-based deallocation

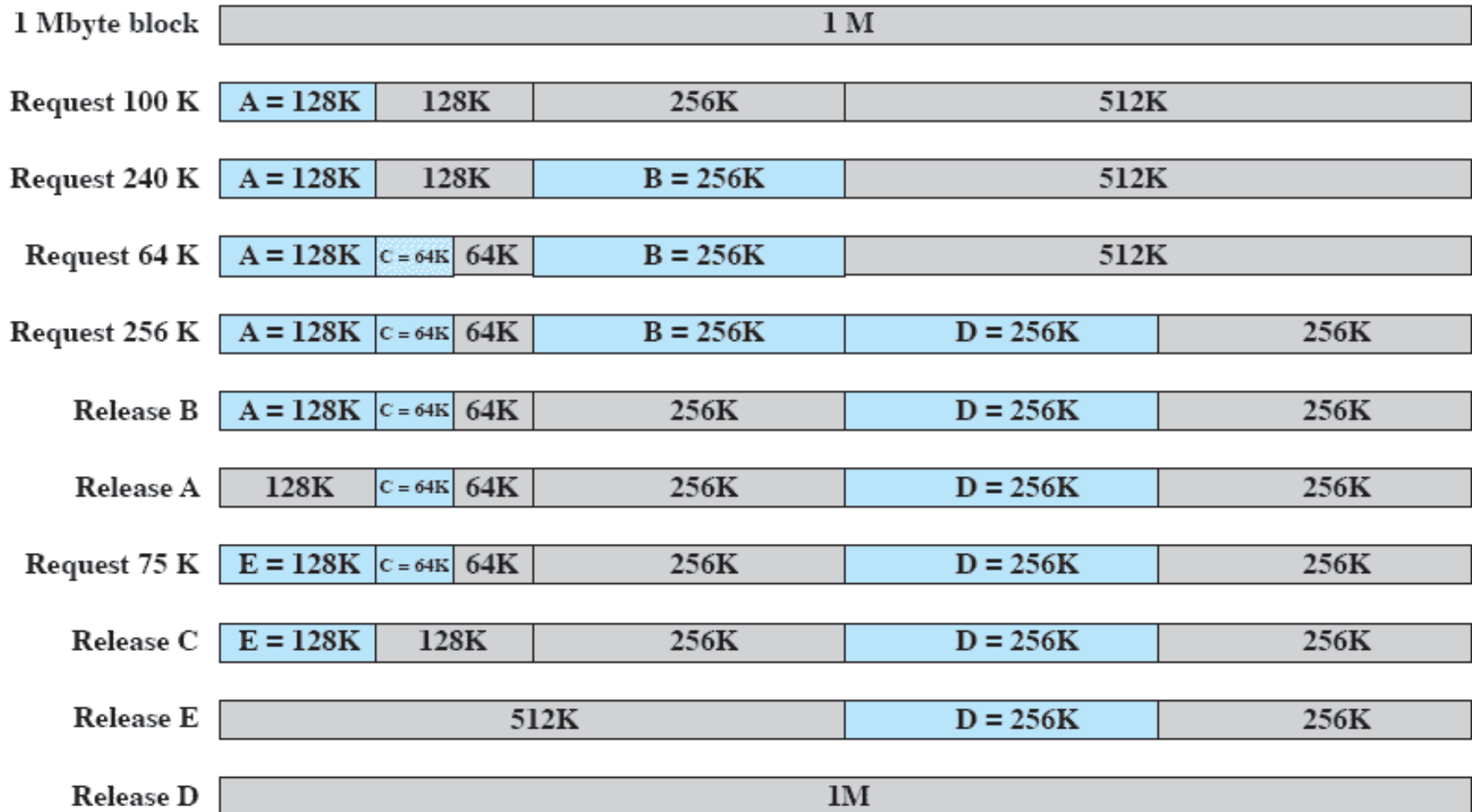


Buddy System – allocate the first block

- To begin, the entire space available for allocation is treated as a single block of size 2^U
- The request size is first rounded up as power of 2, denoted as S
 - E.g., 55 $\rightarrow 2^6=64$; 120 $\rightarrow 2^7=128$
- If $S = 2^U$, then the entire block is allocated. Otherwise, the block is split into two equal *buddies* of size 2^{U-1}
- If $S = 2^{U-1}$, then the request is allocated to one of the two buddies. Otherwise, one of the buddies is split into halves again. This process continues until the split block is equal to S and it is allocated to the request

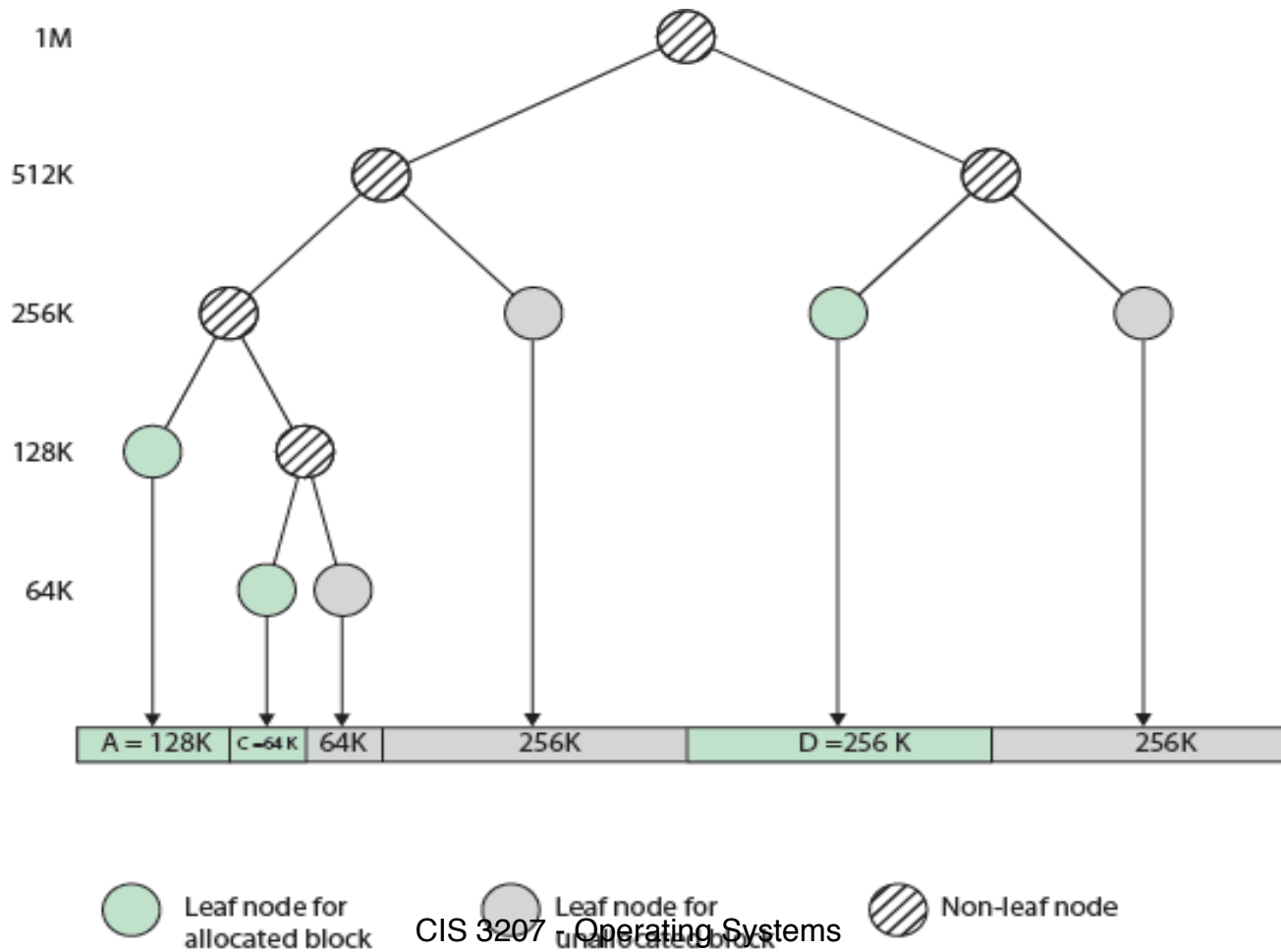


Buddy System – split-based allocation



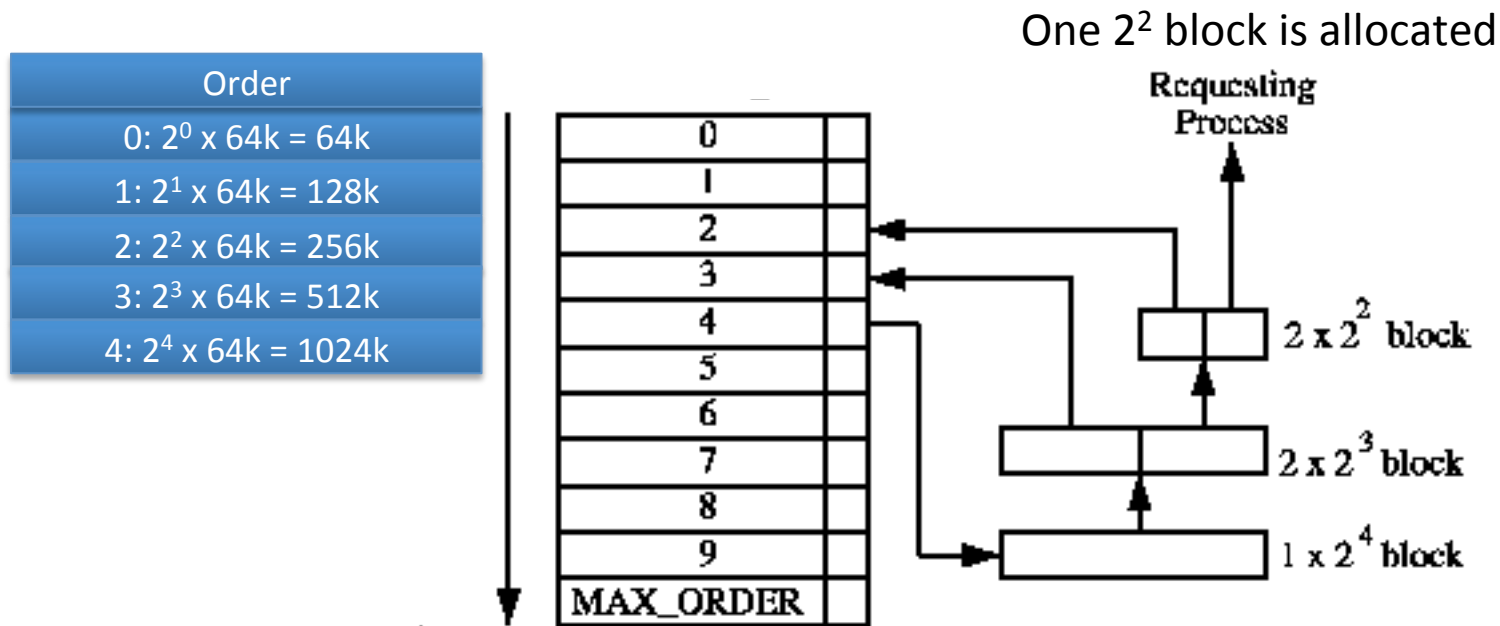
A Binary Tree Representation

- Buddies: sibling nodes, i.e., two nodes that share the parent node in the binary tree representation



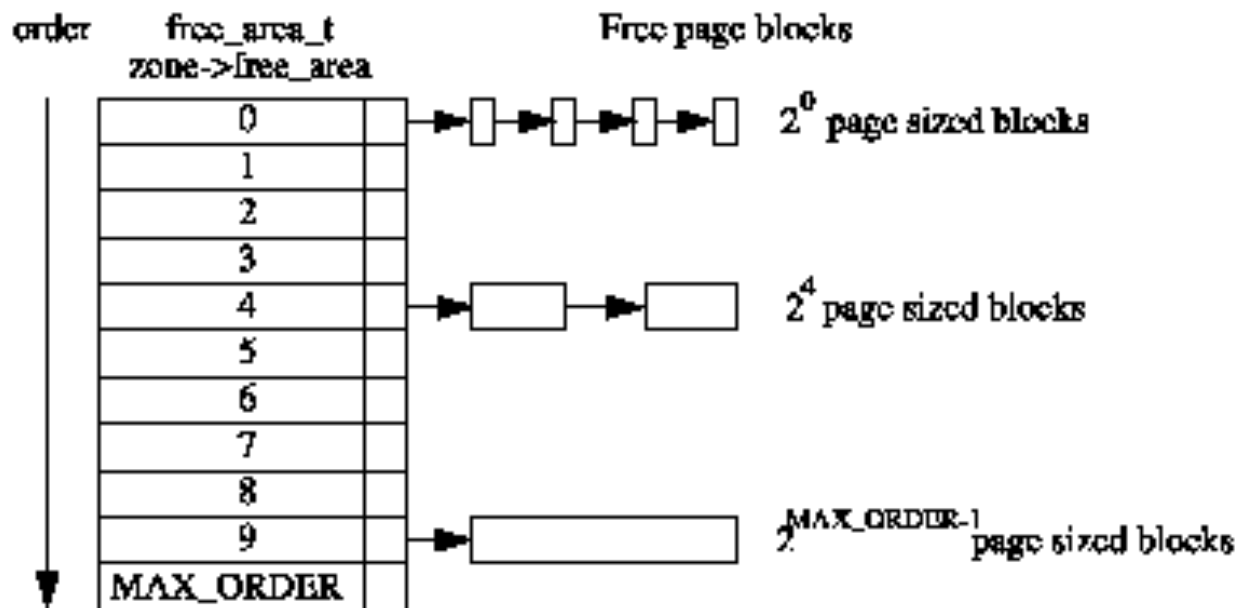
Buddy System – free-lists-based implementation

A free list is a linked list data structure used to connect all unallocated storage chunks. It is quick and easy to allocate and unallocate a storage chunk



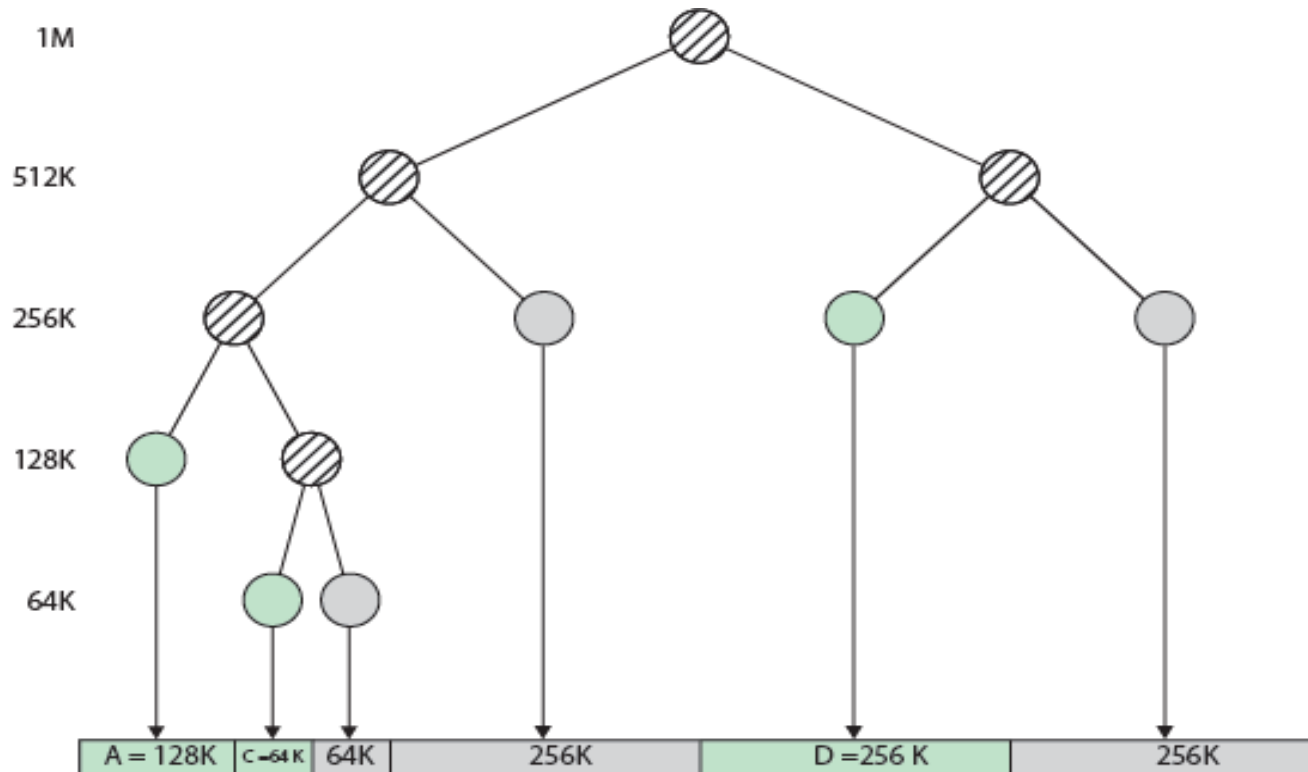
Buddy System used In Linux

- The Buddy System is used in Linux (and many other OSes) to manage the physical memory



Buddy System – coalescing-buddy-based deallocation

- Whenever a block is freed, it tries to coalesce with its buddy. The coalescing procedure is recursive




Buddy System - fragmentation

- Internal fragmentation?
 - Worst case $\sim 50\%$
 - E.g., request 128.01K, which is rounded up to 128x2K
- External fragmentation
 - Still exists, as coalescing can only occur between buddies
 - Best-fit is always used: freelists-based implementation allows you to find the best-fit chunk *quickly*
 - By applying restriction on splitting, you will not see small mini tiny useless holes; e.g., previously we limit the smallest block as 64k




Big picture

- Fixed partitions
- Dynamic partitions
- Buddy system



Contiguous allocation:
Each process occupies a contiguous
memory region in the physical memory

- Segmentation
- Paging



Non-contiguous allocation:
Each process comprises multiple memory regions
scattered in the physical memory.

