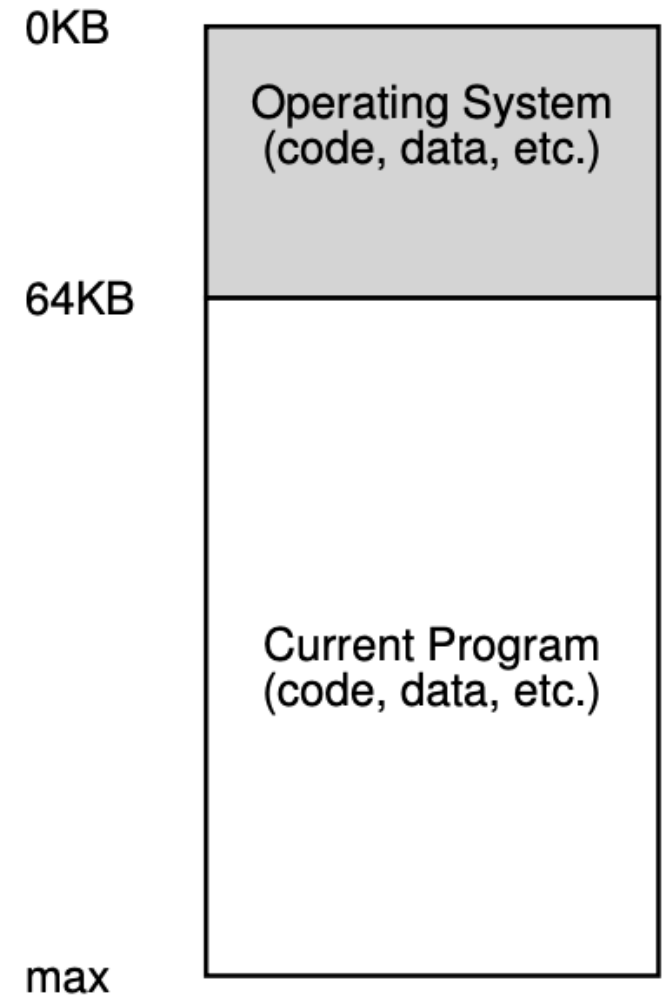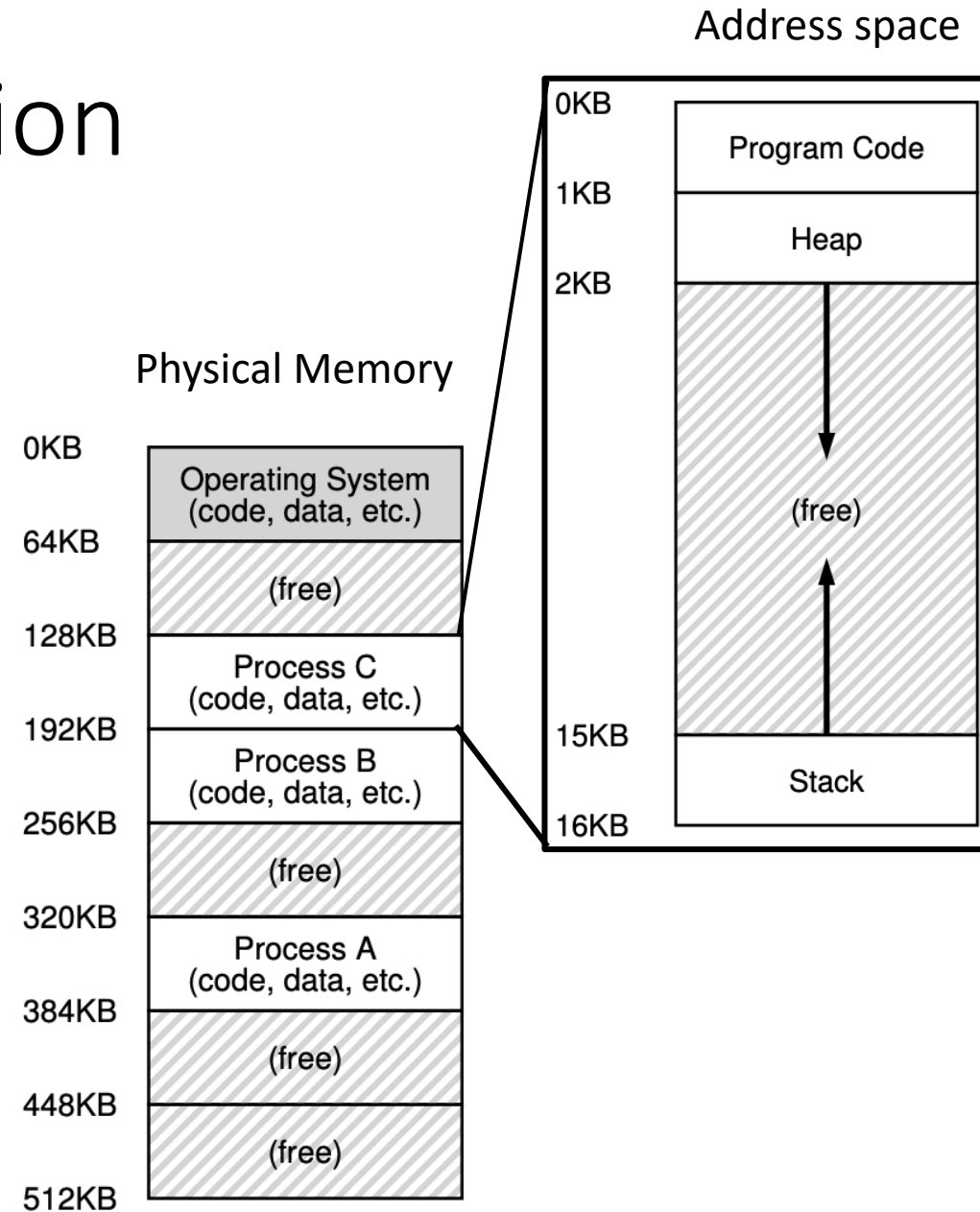# Address Spaces

Chapter 13

Background

- Initially the OS was just a set of libraries and primarily ran one process at a time

- Needs and technology changed, and the model was obsolete
  - Giving a process all of memory meant saving all of memory to disk on each swap
  - Secondary storage (disk) is orders of magnitude slower than memory!

- If only we could keep processes in memory….

0KB

Operating System
(code, data, etc.)

64KB

Current Program
(code, data, etc.)

max

The Address Space Abstraction

- The OS virtualizes physical memory into partitions and assigns spaces for each process

- Each process gets its own **address space**
  - From the perspective of a process, this is "all" the memory on the system
  - Memory locations in a process's address space are **virtual addresses**
  - OS ensures that processes stay within their address space
    - Provides security for each process

Address space

0KB	Program Code
1KB	
2KB	Heap
	↓ (free) ↑
15KB	
16KB	Stack

Physical Memory

0KB	Operating System (code, data, etc.)
64KB	(free)
128KB	Process C (code, data, etc.)
192KB	Process B (code, data, etc.)
256KB	(free)
320KB	Process A (code, data, etc.)
384KB	(free)
448KB	(free)
512KB	

Memory API Prelude

- Managing heap memory is not always easy
  - Heap memory can be requested manually
  - Sometimes functions provided by the C library will create heap memory for us and provide a pointer and size of the data

- Good news! There are tools that can help!

- Install valgrind to your VM (detects memory errors/leak detection)
  - sudo apt install valgrind