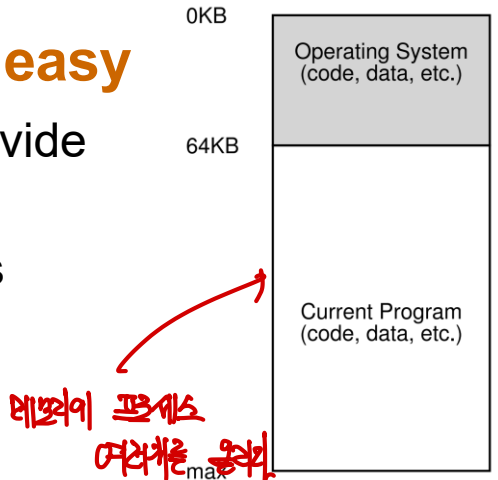
The Abstraction: Address Spaces

Prof. Yongtae Kim

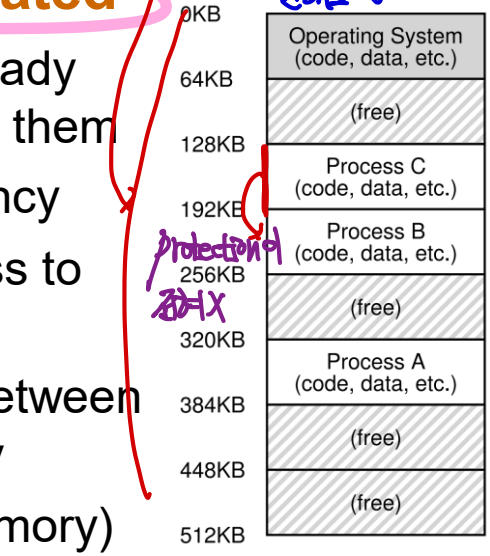
Computer Science and Engineering
Kyungpook National University

Early Systems

- **In early days, building computer systems was easy**
  - From the memory perspective, early machines didn't provide much of an abstraction to users
  - The physical memory of the machine contains the OS as a set of routines (library) and one running process
  - Life was sure easy for OS developers in those days

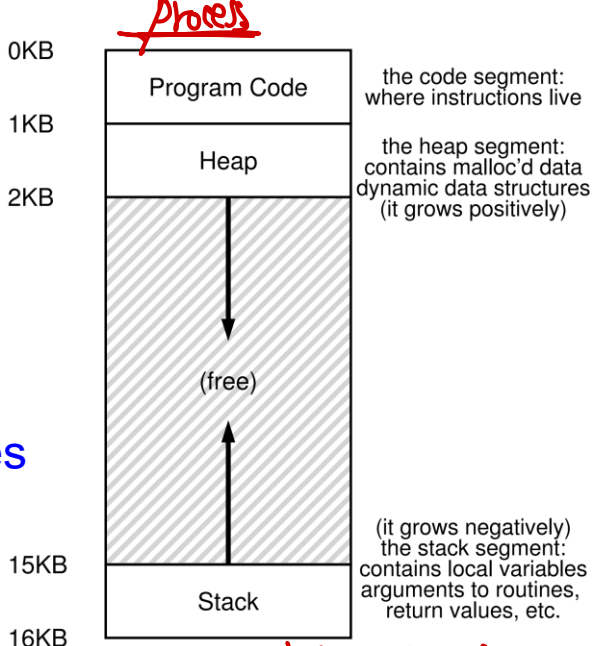


- **Computer systems were getting more complicated**
  - In the multiprogramming era, multiple processes were ready to run at a given time, and the OS would switch between them
  - Doing so increased the utilization of the CPU and efficiency
  - For time sharing era, running one process with full access to all memory causes a big problem (memory swapping)
  - So, processes were stayed in memory while switching between them, allowing OS to implement timing sharing efficiently
  - This leads to protection issue (e.g. A can access B's memory)



The Address Space

- **The OS creates an easy to use abstraction of physical memory**
  - This memory abstraction is called the address space
  - It is the running program's view of memory in the system
- **Understanding this fundamental OS abstraction of memory is key to understanding how memory is virtualized**
  - The address space of a process contains all of the memory state of running program
  - Code: where instructions live
  - Stack: local variables, parameters, return address
  - Heap: dynamically allocated memory (`malloc()`)
  - The program isn't in memory in physical addresses 0 ~ 16KB; rather at some arbitrary physical addresses
  - How can the OS build this abstraction of a private, potentially large address space for multiple running processes on top of a single, physical memory?



- **The OS is virtualizing memory with some hardware support**

Goals

- **One major goal of a virtual memory (VM) system is transparency**
  - The program shouldn't be aware of the fact that memory is virtualized
  - The program behaves as if it has its own private physical memory

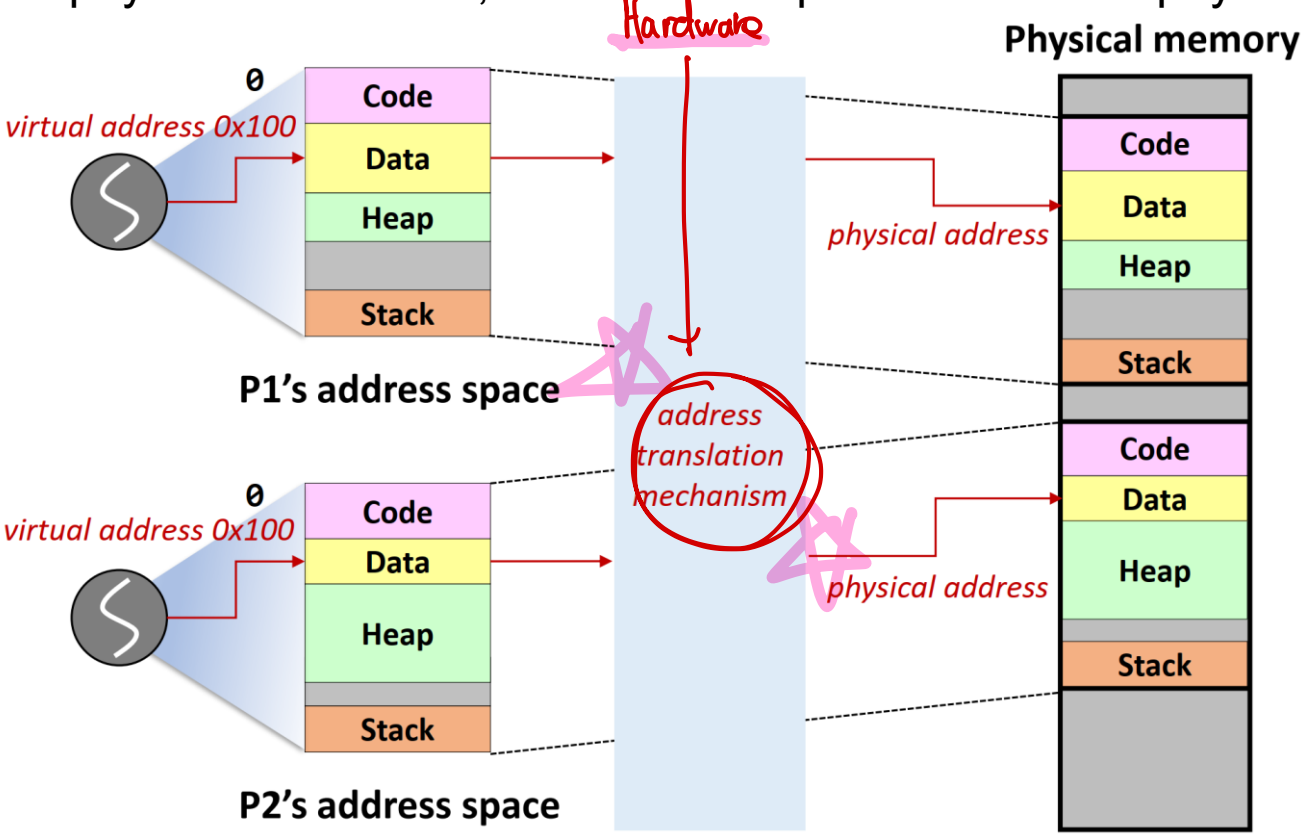
- **Another goal of VM is efficiency**
  - The OS should strive to make the virtualization as efficient as possible both in terms of time and space
  - The OS will have to rely on hardware support (e.g. TLBs)

- **Finally, a third VM goal is protection**
  - The OS should make sure to protect processes from on another as well as the OS itself from process
  - Protection thus enables us to deliver the property of isolation among processes

- **We will focus our exploration on the basic mechanisms needed to virtualize memory, including both OS and hardware support**

Summary

- **The virtual memory system is responsible for providing the illusion of a large, sparse, private address space to programs**
  - The OS, with some hardware help, will take virtual memory references, and turn them into physical addresses, which can be presented to the physical memory



Courtesy of Prof. Jin-Soo Kim @ SNU