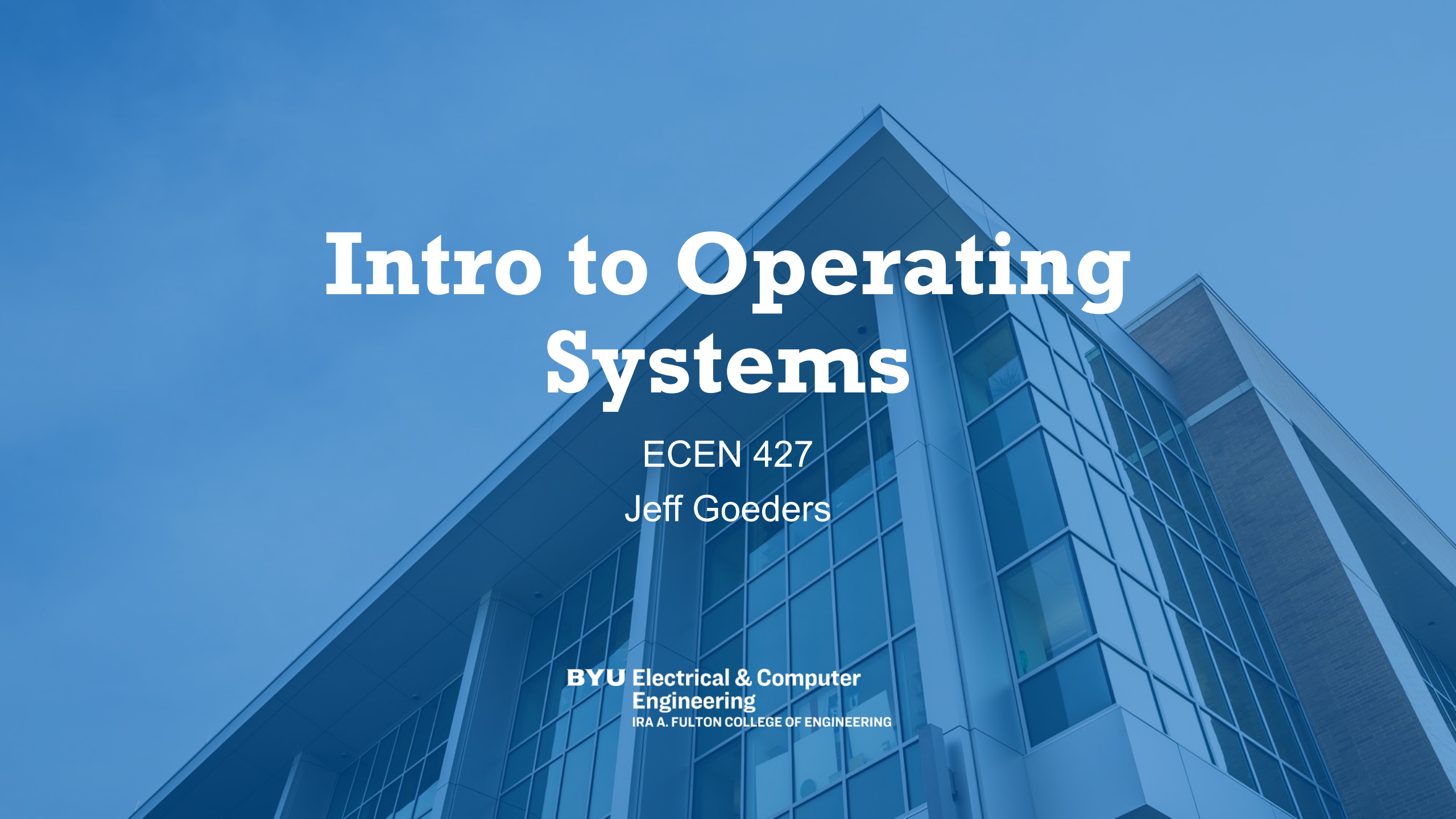


Intro to Operating Systems

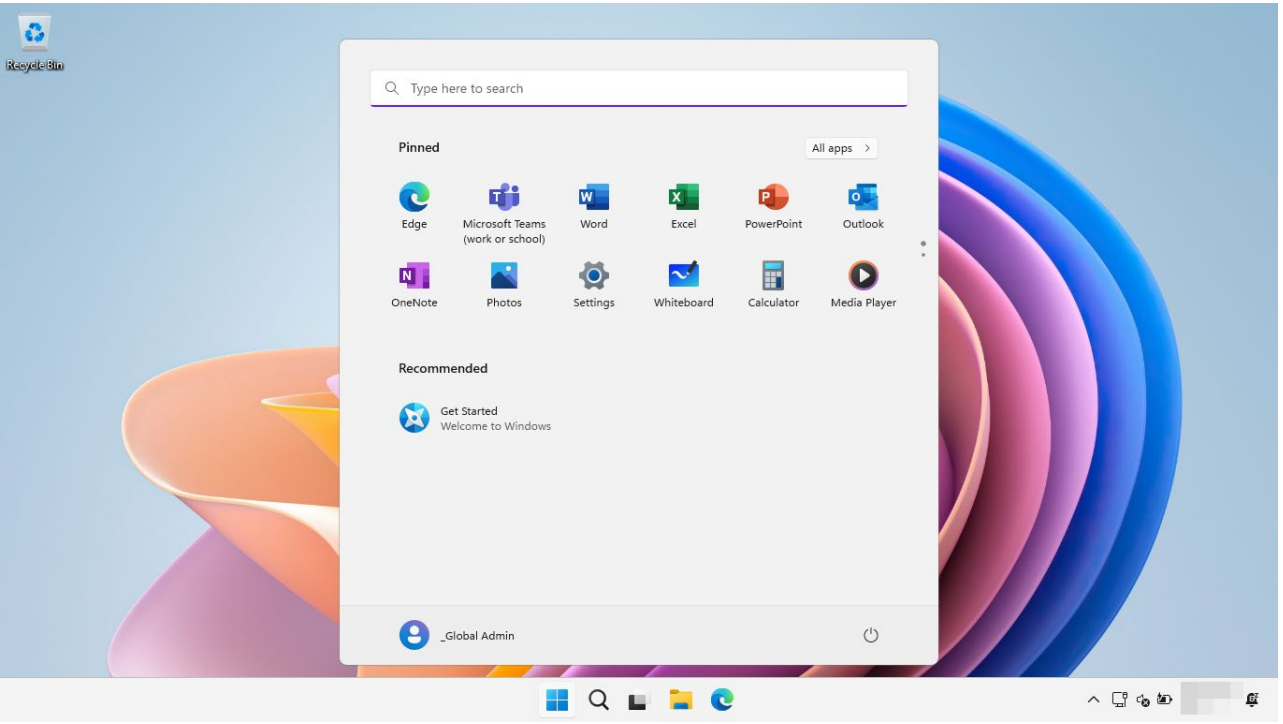
ECEN 427

Jeff Goeders

BYU Electrical & Computer Engineering
IRA A. FULTON COLLEGE OF ENGINEERING

What is an Operating System?

Windows



MacOS

Operating Systems

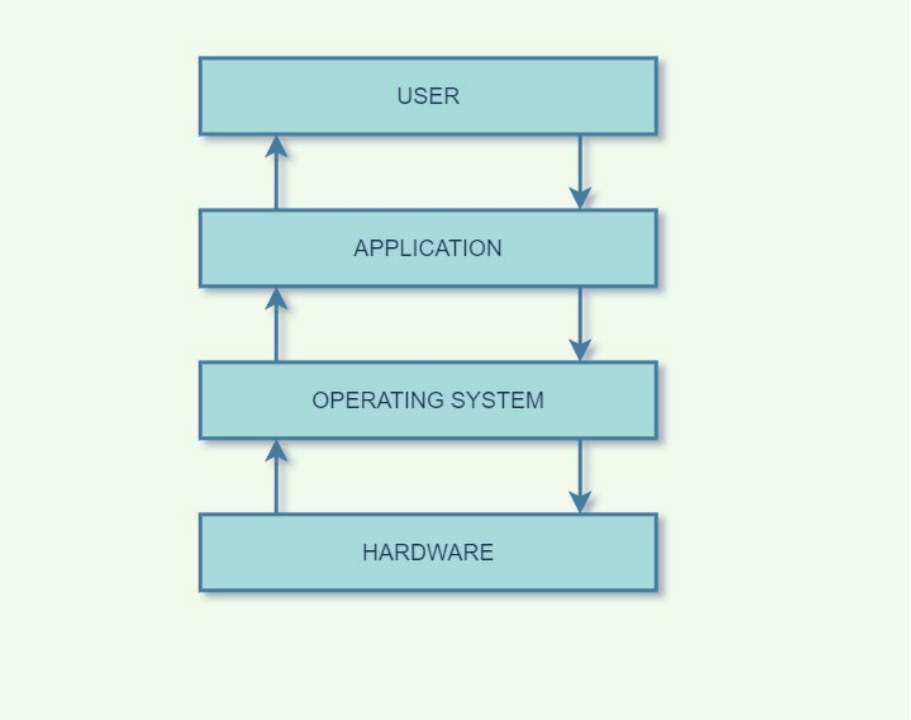
"Allows you to run multiple programs simultaneously, allowing programs to share memory, enabling programs to interact with devices, …"

"The primary way the OS does this is through a general technique that we call **virtualization**. That is, the OS takes a **physical** resource (such as the processor, or memory, or a disk) and transforms it into a more general, powerful, and easy-to-use **virtual** form of itself. Thus, we sometimes refer to the operating system as a **virtual machine**.

What does an operating system do?

1. Process management (runs programs)
2. Memory management
3. File system management
4. Device management
5. Network management

Sometimes we call this the “**kernel**”



https://www.mygreatlearning.com/blog/what-is-operating-system/

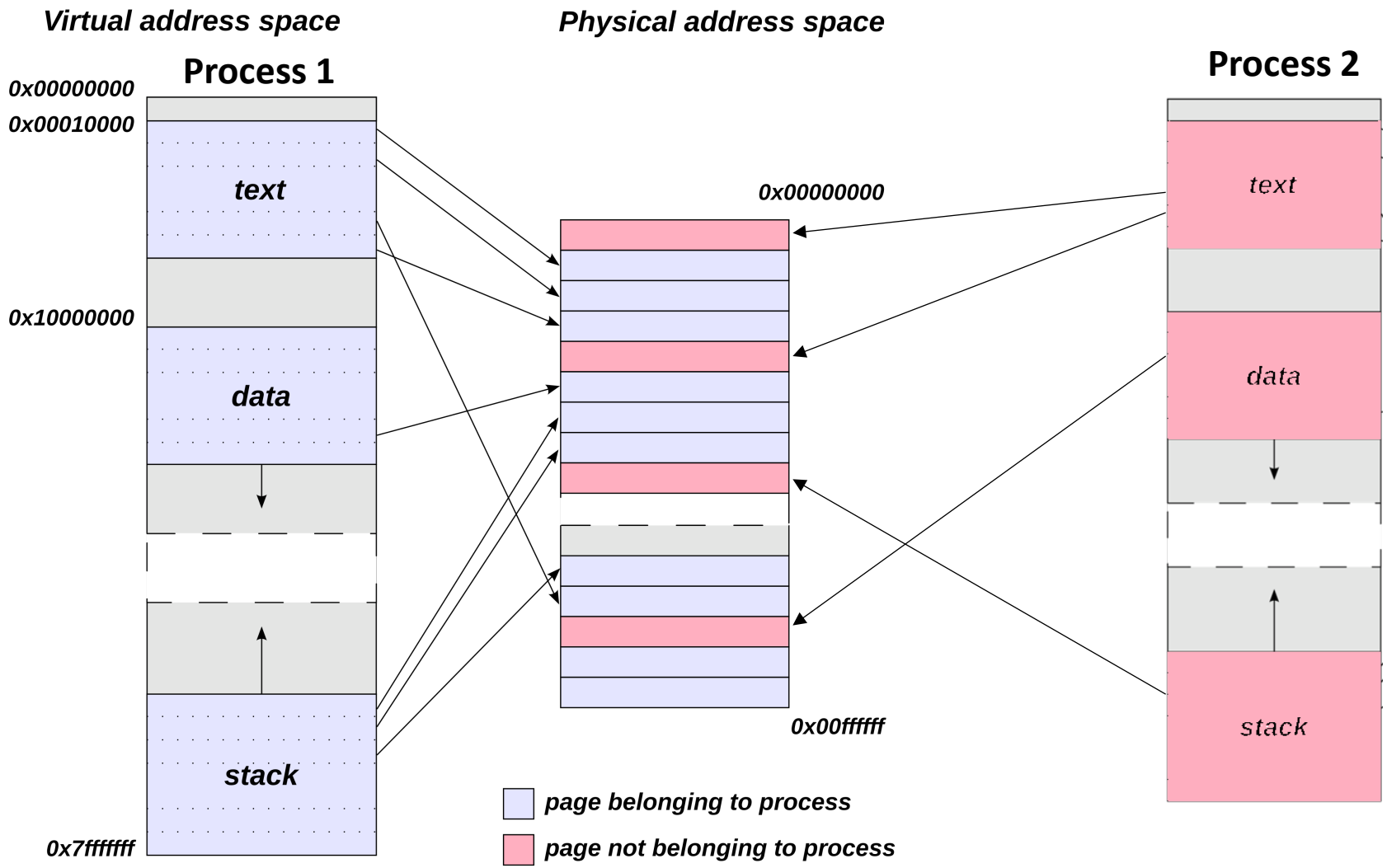
Virtualizing the CPU

- Example

Virtualizing Memory

- Example

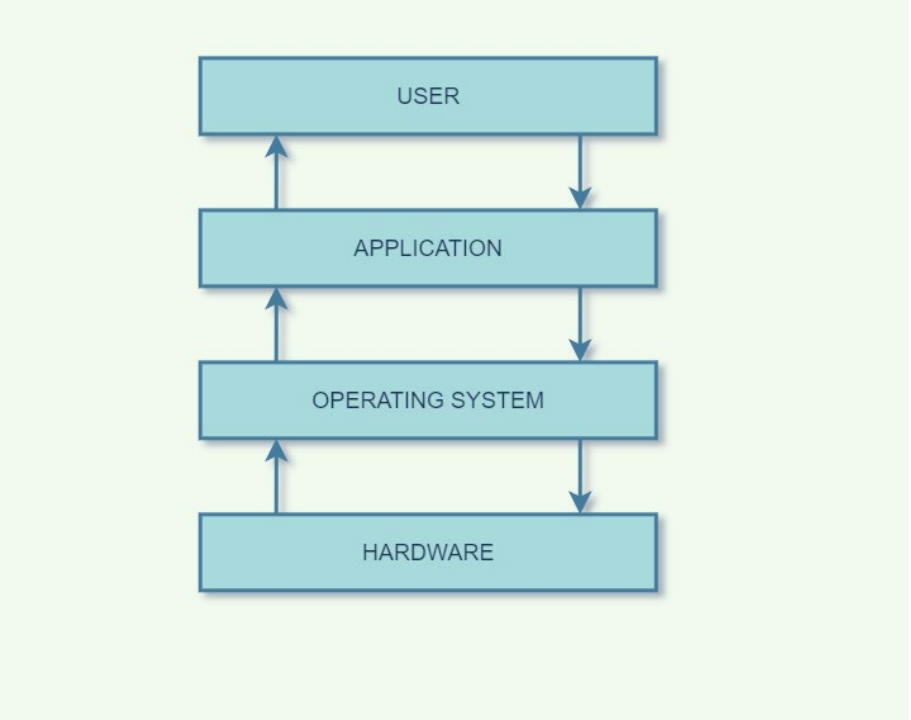
Virtualizing Memory



What does an operating system do?

1. Process management (runs programs)
2. Memory management
3. File system management
4. Device management
5. Network management

Sometimes we call this the “**kernel**”



https://www.mygreatlearning.com/blog/what-is-operating-system/

Interacting with the Operating System

Often we need to “interact” with the operating system:

1. *Process management* -> Run a program
2. *Memory management* -> Allocate memory
3. *File system management* -> Open/read/write files
4. *Device management* -> Read from a USB device
5. *Network management* -> Open a network socket

Early Operating Systems:

- Were very simplistic; just a set of available functions in a library.

- Problem: No security.
- Imagine if any program could read from any memory location, or anywhere on the disk.

System Calls

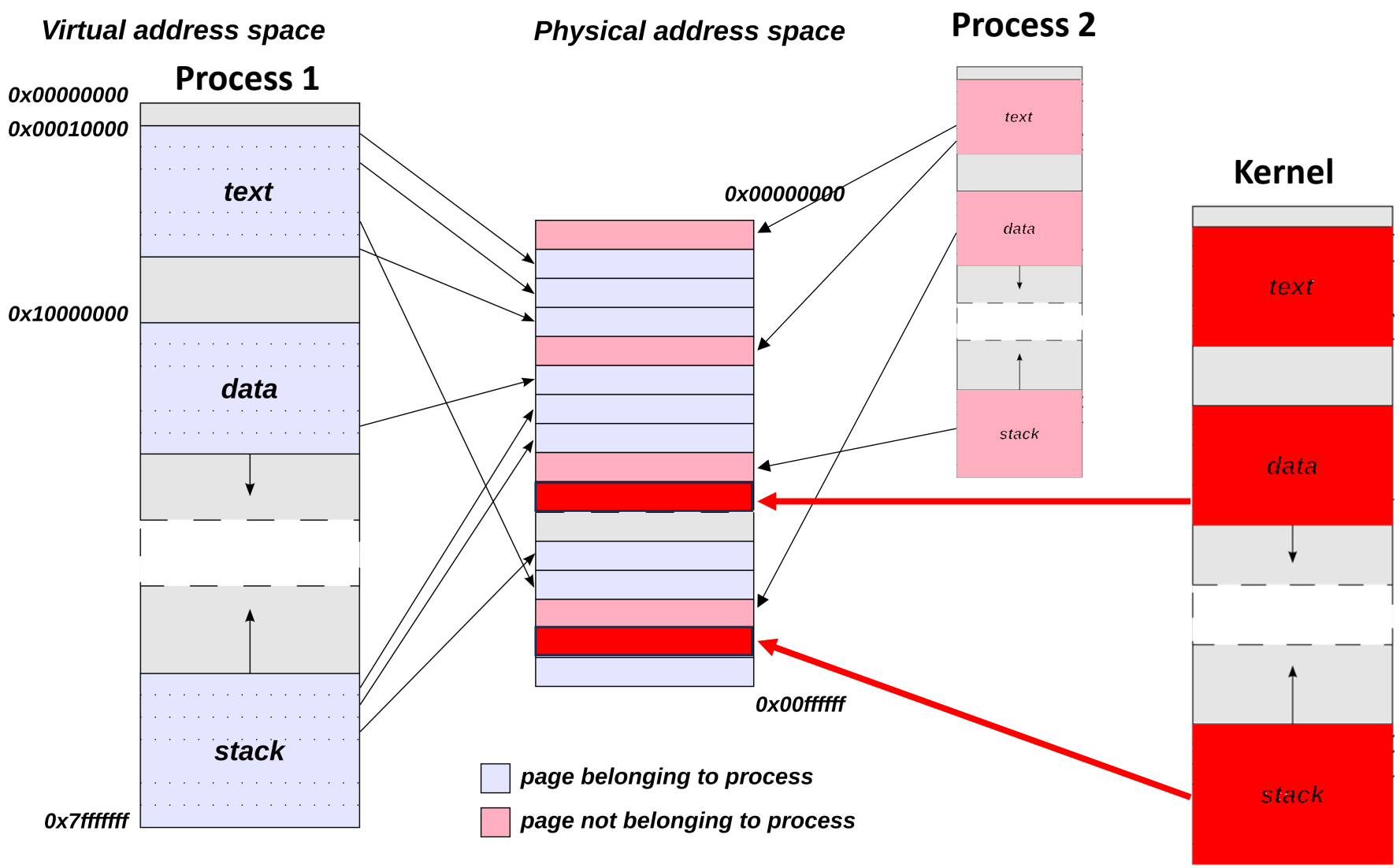
- To allow for security and separation/privacy between programs, **system calls**, were introduced.

- User applications run in what is referred to as **user mode (or userspace)** which means the hardware restricts what applications can do
  - For example, an application running in user mode can't typically initiate an I/O request to the disk, access any physical memory page, or send a packet on the network.

- A system call transfers control (i.e., jumps) into the OS while simultaneously raising the hardware privilege level (this is done via a **trap** instruction)
  - When a system call is initiated, the hardware transfers program control to a system call handler and simultaneously raises the privilege level to **kernel mode**.
  - In kernel mode, the OS has full access to the hardware of the system and thus can do things like initiate an I/O request or make more memory available to a program.

- When the OS is done servicing the request, it passes control back to the user via a special **return-from-trap** instruction, which reverts to user mode while simultaneously passing control back to where the application left off.

Virtualizing Memory



Linux System Calls

- https://www.chromium.org/chromium-os/developer-library/reference/linux-constants/syscalls/

- strace ./a.out
- strace --summary-only ./a.out