CS420: Operating Systems

Operating System Structure

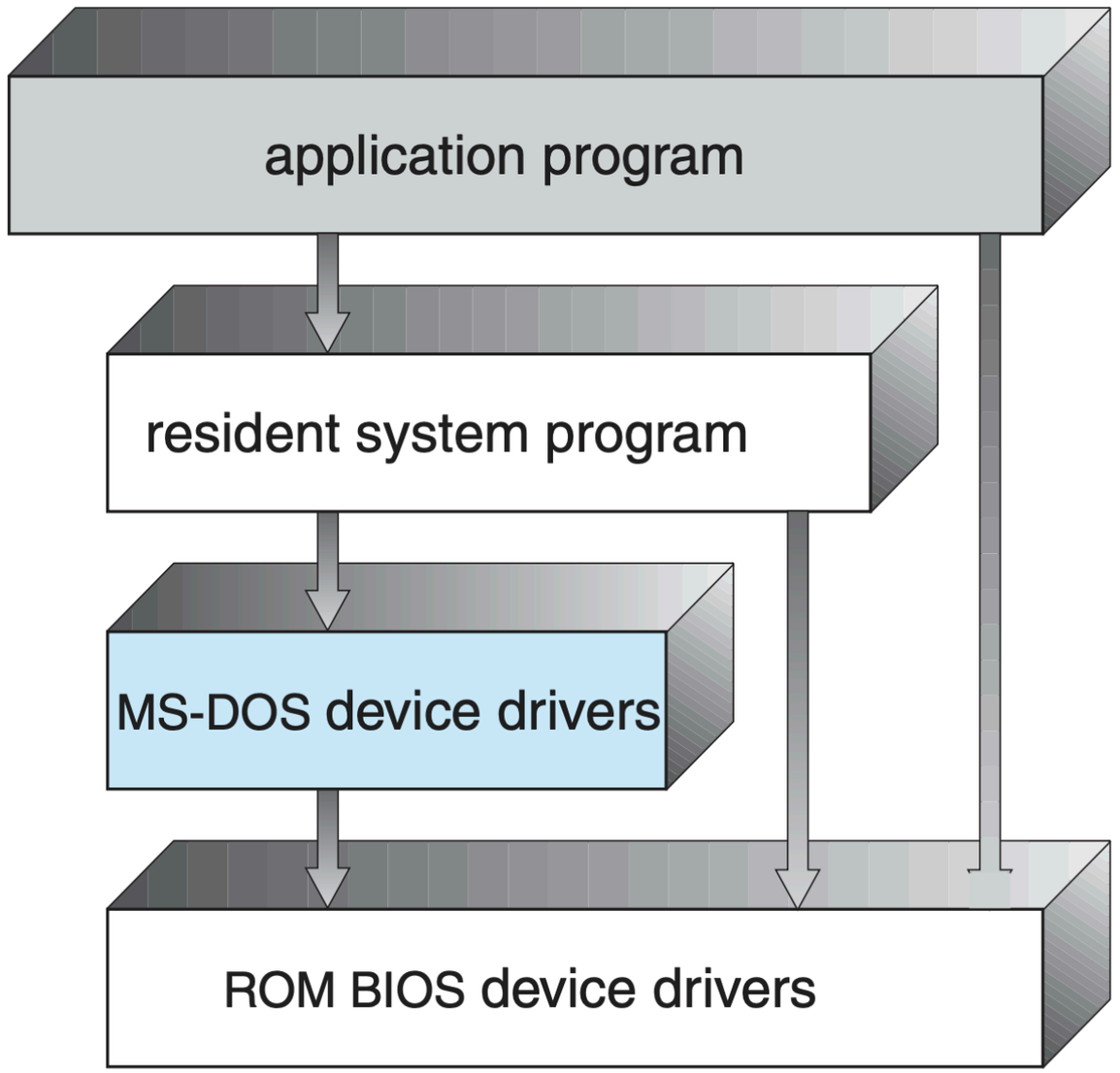
James Moscola
Department of Engineering & Computer Science
York College of Pennsylvania

Operating System Design and Implementation

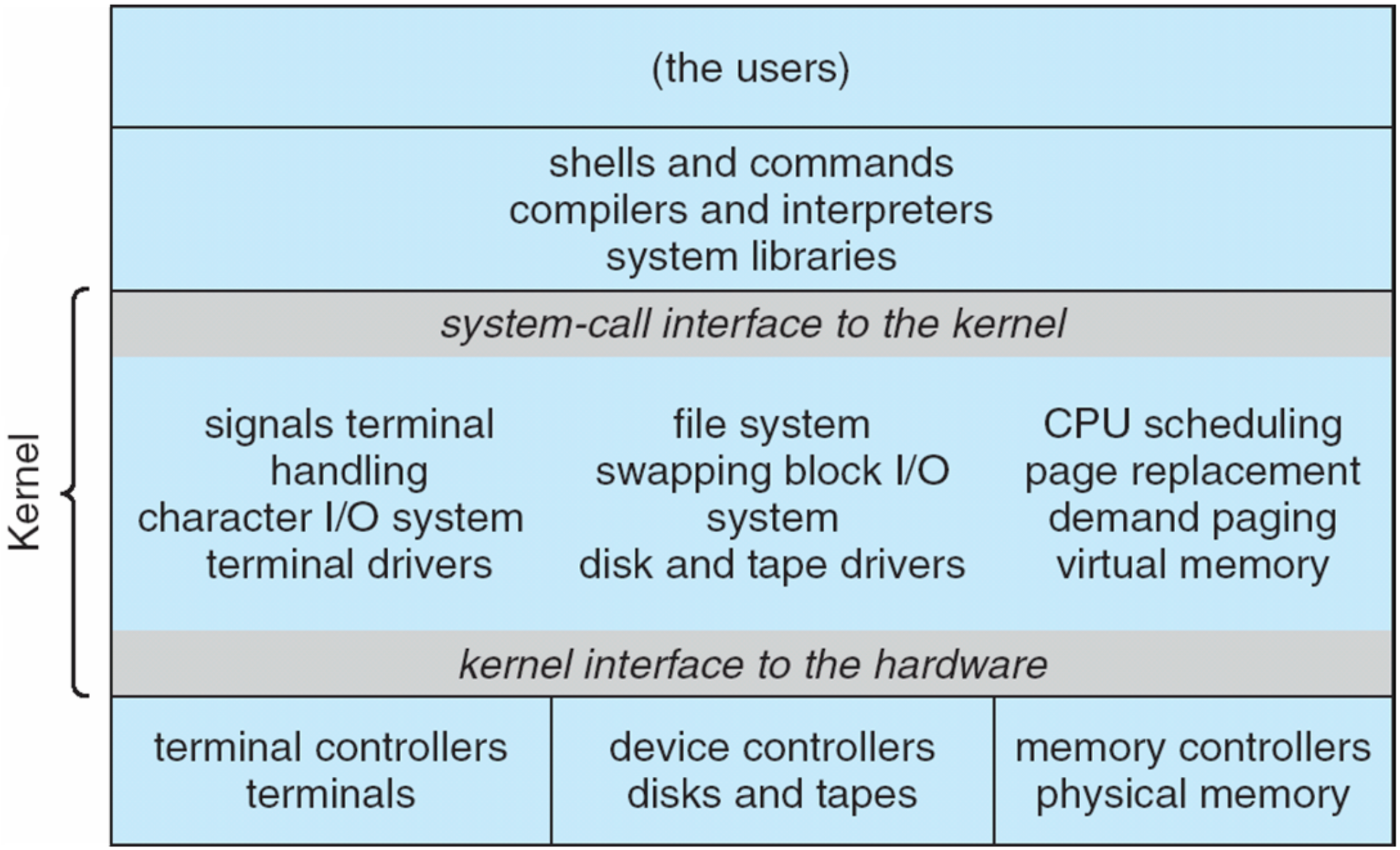
- **Design and Implementation of OS not "solvable"**
  - There is no single OS that works best for every use case
- **How does one design an operating system??**
  - Start by defining goals and specifications
  - Affected by choice of hardware, type of system
    - Desktop/laptop, mobile, real-time, distributed, etc.
  - User goals and System goals
    - User goals – operating system should be convenient to use, easy to learn, reliable, safe, and fast
    - System goals – operating system should be easy to design, implement, and maintain, as well as flexible, reliable, error-free, and efficient

An OS With Simple Structure

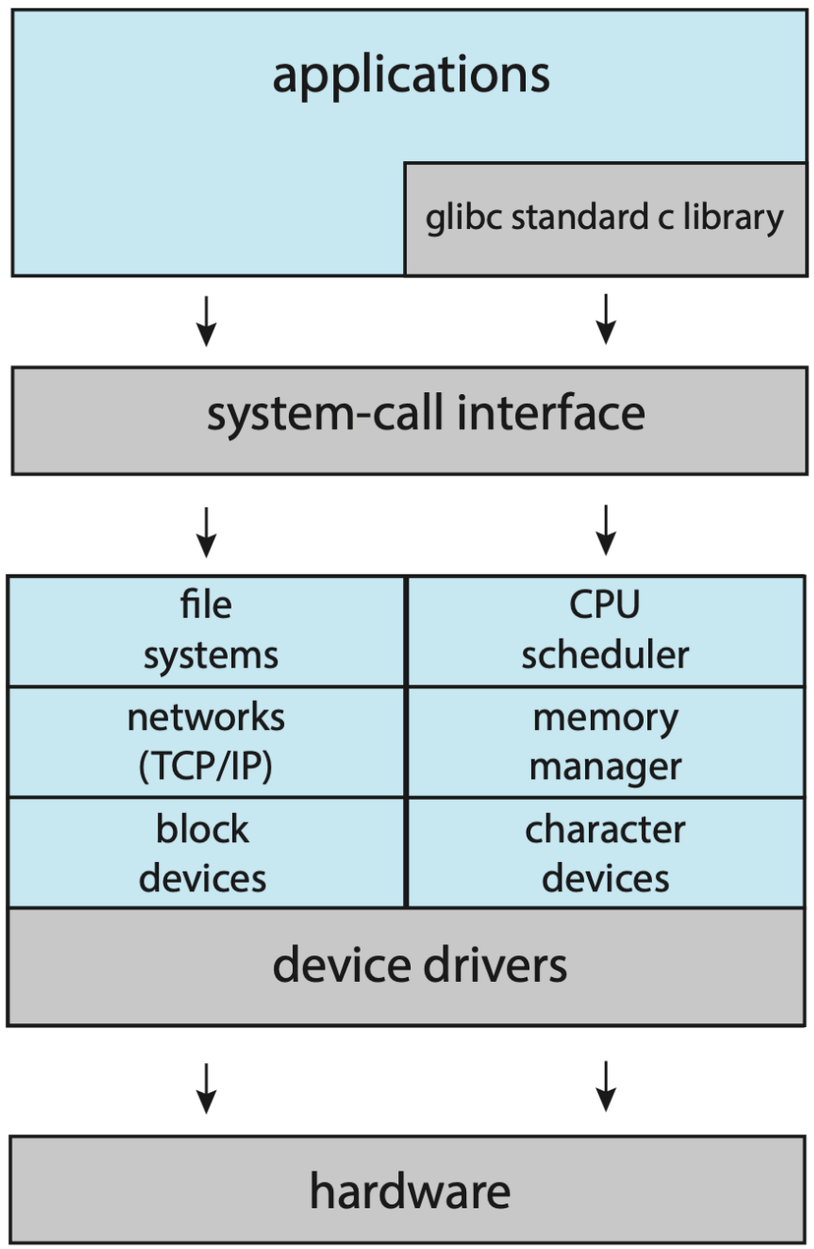
- **MS-DOS – written to provide the most functionality in the least space**
  - Not divided into modules
- **Interfaces and levels of functionality not well separated**
  - Applications able to write directly to hardware like disks
  - Vulnerable to malicious programs that could crash entire system
- **No dual-mode operation available at the time as hardware didn't support it**



Traditional UNIX System Structure w/ Monolithic Kernel



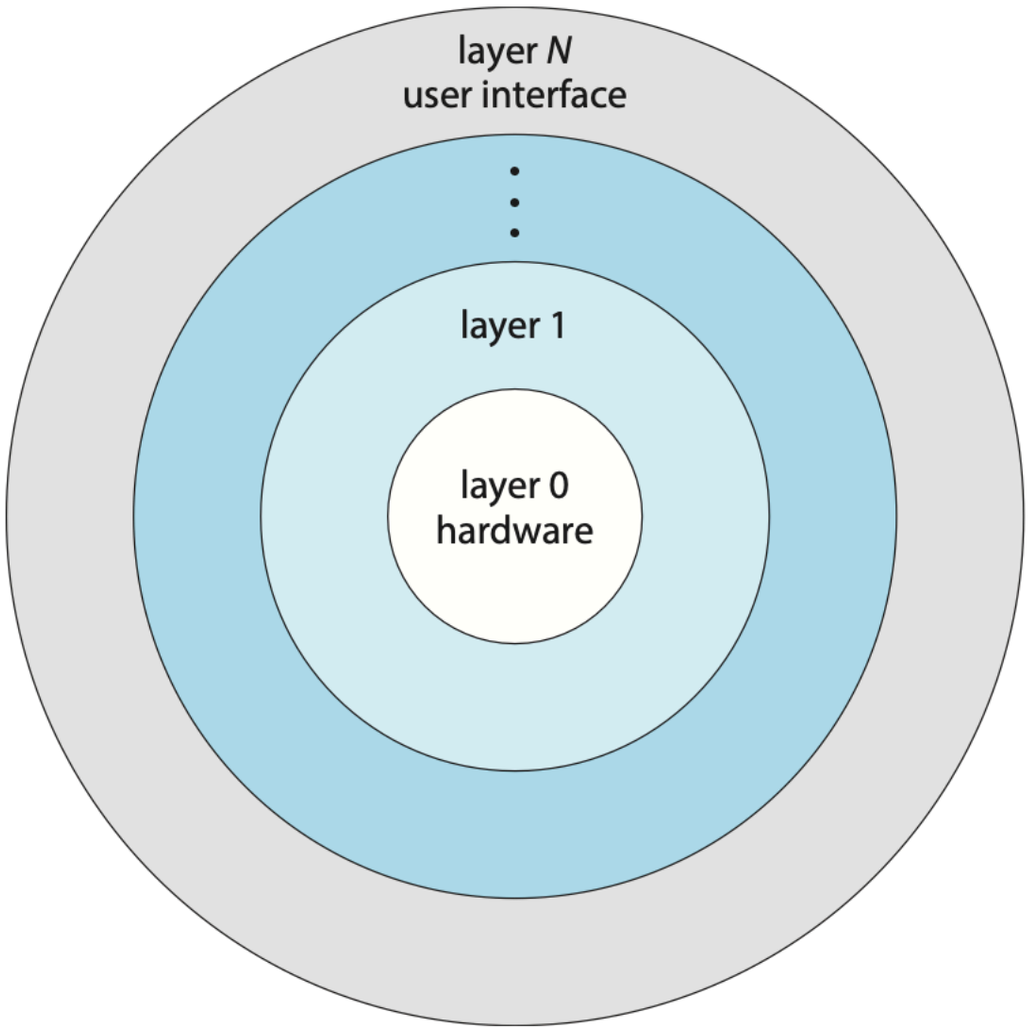
Linux System Structure



- **Developers can interact with OS in multiple ways**
  - Make system calls directly
  - Use common library like the glibc standard c library
    - Includes functions like: **`printf`**, **`fopen`**, **`fclose`**, **`strlen`**, **`strncat`**, **`time`**, **`toupper`**, and hundreds more!

Layered Operating System Approach

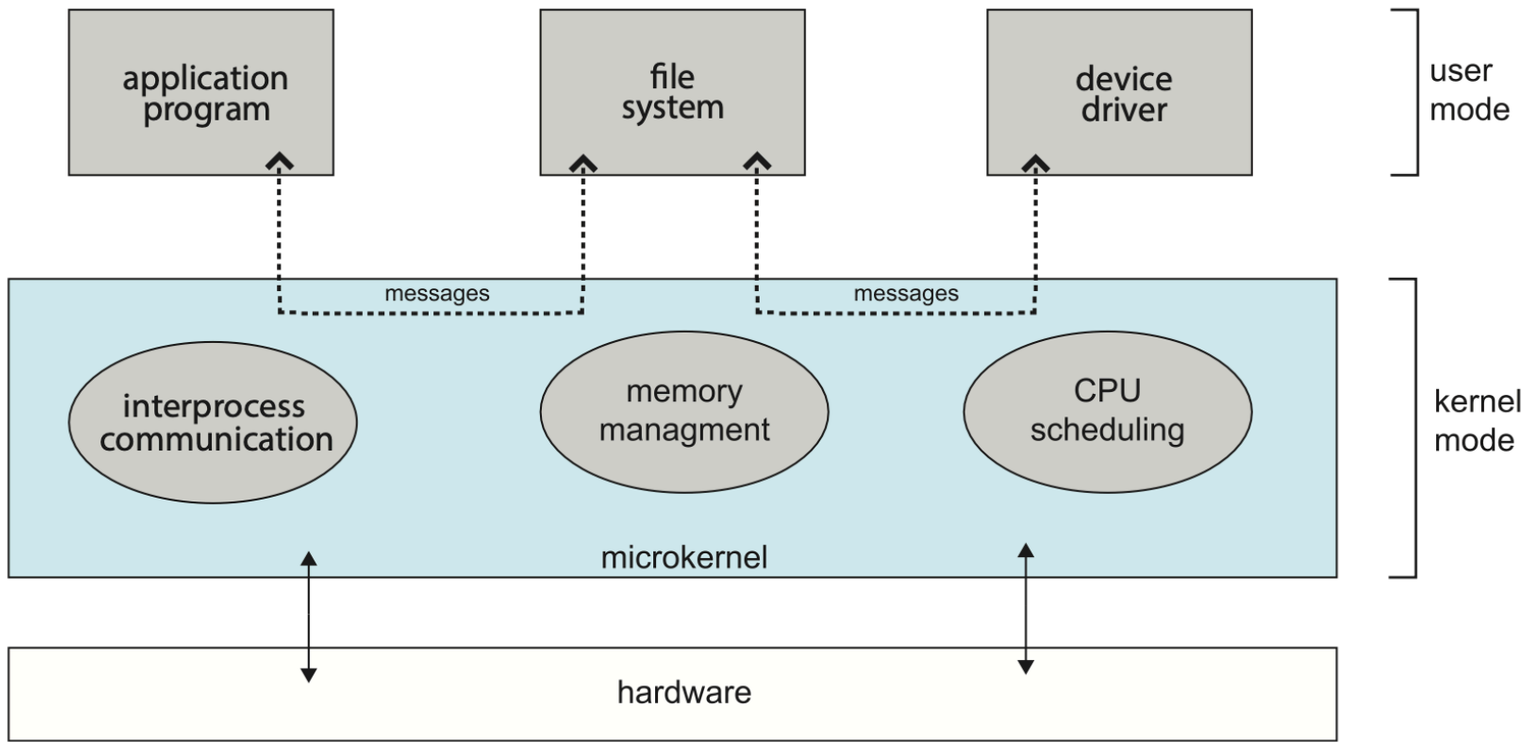
- **The operating system is divided into a number of layers (levels), each built on top of lower layers. The bottom layer (layer 0), is the hardware; the highest (layer N) is the user interface.**



- **With modularity, layers are selected such that each uses functions (operations) and services of only lower-level layers**
  - An OS can be designed from the lower layers up; ensure that lower layers work before moving to higher layers

Microkernel System Structure

- **Moves all nonessential components of the kernel into 'user' space modules**
- **Communication between user modules uses message passing**
- **Pros:**
  - Smaller kernel
  - Easier to extend
  - Easier to port to new HW architectures
  - More secure & reliable as services run in user mode



- **Cons:**
  - Performance overhead of user space to kernel space communication

Loadable Kernel Modules

- **Link in additional services via modules at boot or during runtime**
  - No need to recompile kernel to add support for new devices / file systems
- **Most modern operating systems implement kernel modules**
  - Each module is loadable as needed within the kernel
  - Each module has well-defined interfaces
  - Any module can call any other module
    - No need to utilize message passing
- **Overall, similar to layers but more flexible**

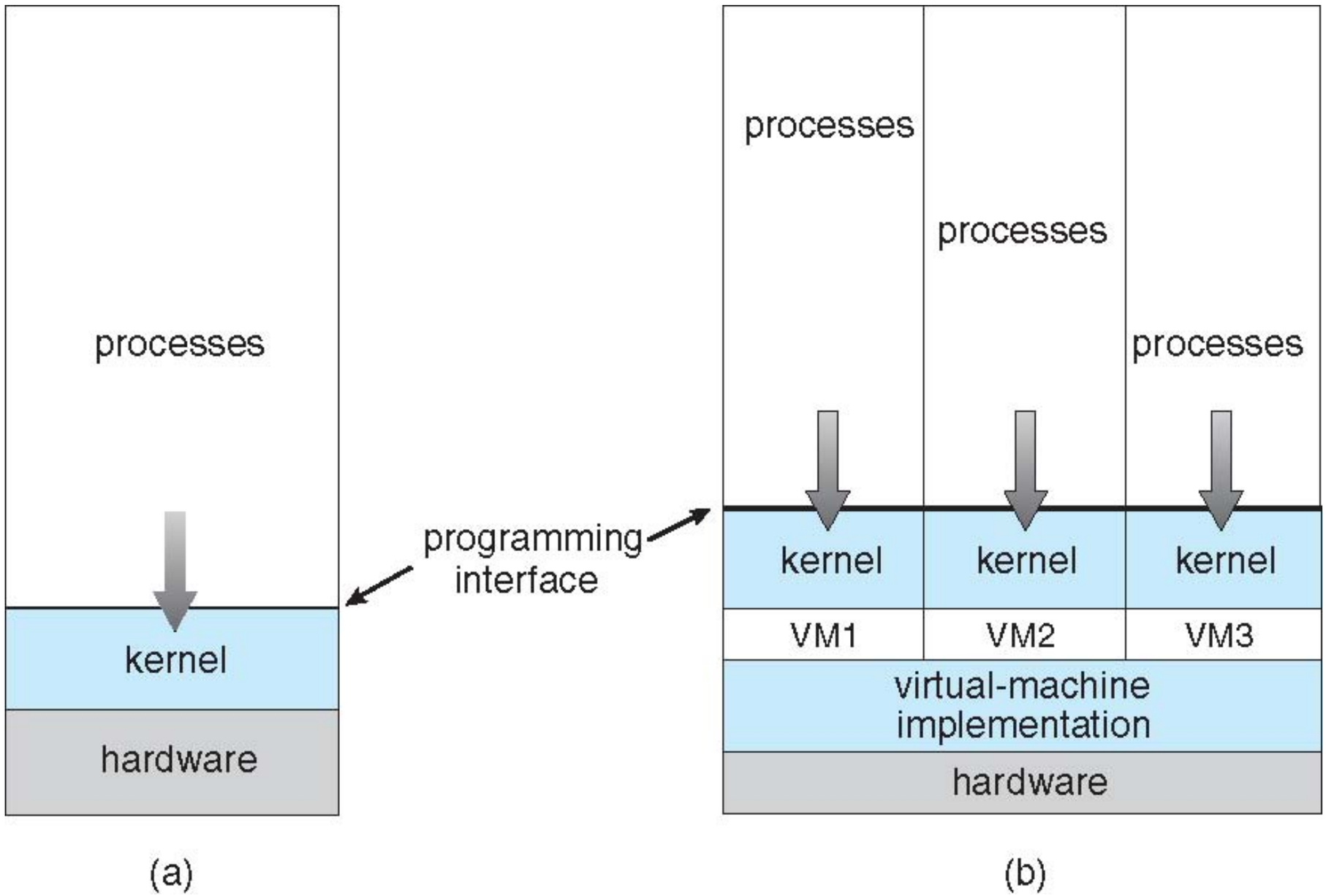
Virtual Machines

- **Main idea — abstract the hardware of a single computer into several different execution environments**
  - Actual computer is called the 'host'
  - Create the illusion that each separate environment is running on its own private computer with own processor, memory, etc.
  - Virtual Machine Manager creates and runs 'guest' VMs by providing an interface that is identical to the host
  - Each guest provided with a (virtual) copy of underlying computer.

Virtual Machine Benefits

- **Fundamentally, multiple execution environments (different operating systems) can share the same hardware**
- **Protected from each other**
- **Some sharing of file can be permitted, but controlled**
- **Communicate with each other, other physical systems via networking**
- **Useful for development, testing**
- **Consolidation of many low-resource use systems onto fewer busier systems**
- **“Open Virtual Machine Format”, standard format of virtual machines, allows a VM to run within many different virtual machine (host) platforms**

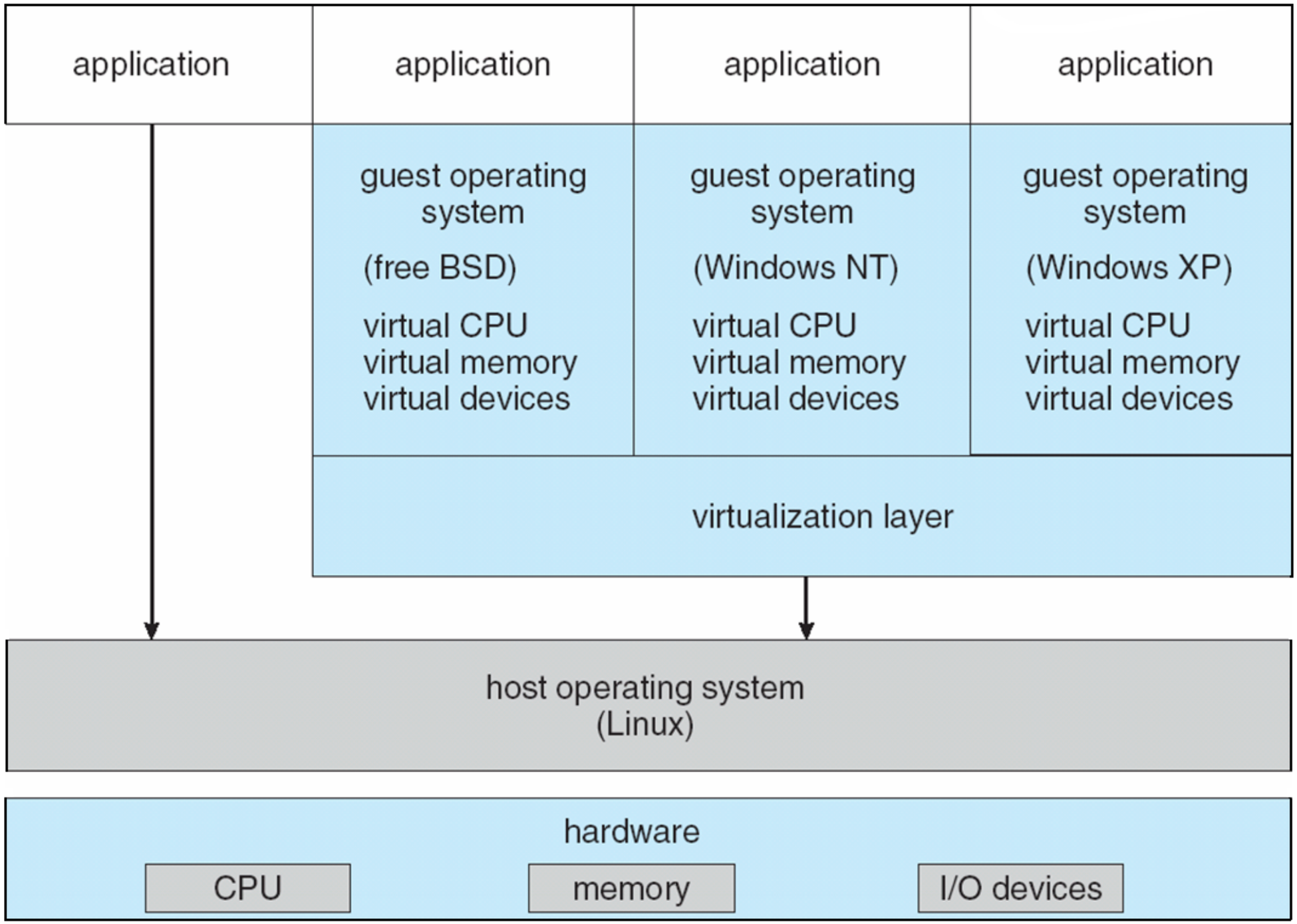
Virtual Machines



Non-virtual Machine

Virtual Machine w/ type 0 hypervisor (VMM runs directly on hardware)

VMware Architecture (Type 2 Hypervisor)



The Java Virtual Machine

- **Programming-Environment Virtualization**
  - JVM is compiled to be a native program on systems on which it runs
  - Converts Java bytecode into native machine code for host system