15B11CI412

Operating Systems & System Programming

Introduction

COURSE OUTCOMES

1. **CO1:**Describe and explain the fundamental components of operating systems and system programming.
2. **CO2:**Apply and compare various policies of scheduling in processes and threads in OS.
3. **CO3:**Describe and discuss various resource management techniques of operating systems and compare their performances.
4. **CO4:**Understand the concept of IPC and describe various process synchronization techniques in OS.
5. **CO5:**Discuss the working of IO management and apply various disk scheduling techniques.
6. **CO6:**Analyze and report appropriate OS design choices when building real-world systems.

Recommended Reading material

1. **William Stallings, "OPERATING SYSTEMS INTERNALS AND DESIGN PRINCIPLES".**
2. Andrew S. Tanenbaum, "Operating Systems Design and Implementation", Third Edition, Prentice Hall Publications 2006
3. A.S. Tanenbaum, "Modern Operating Systems", 2nd edition, Prentice Hall India.
4. **A.Silberschatz, P.Galvin, G. Gagne, "Operating systems concepts" Willey international company (sixth edition)**

Introduction

- What is an operating system?
- Operating Systems History
  - Simple Batch Systems
  - Multiprogrammed Batch Systems
  - Time-sharing Systems
  - Personal Computer Systems
- Parallel and Distributed Systems
- Real-time Systems

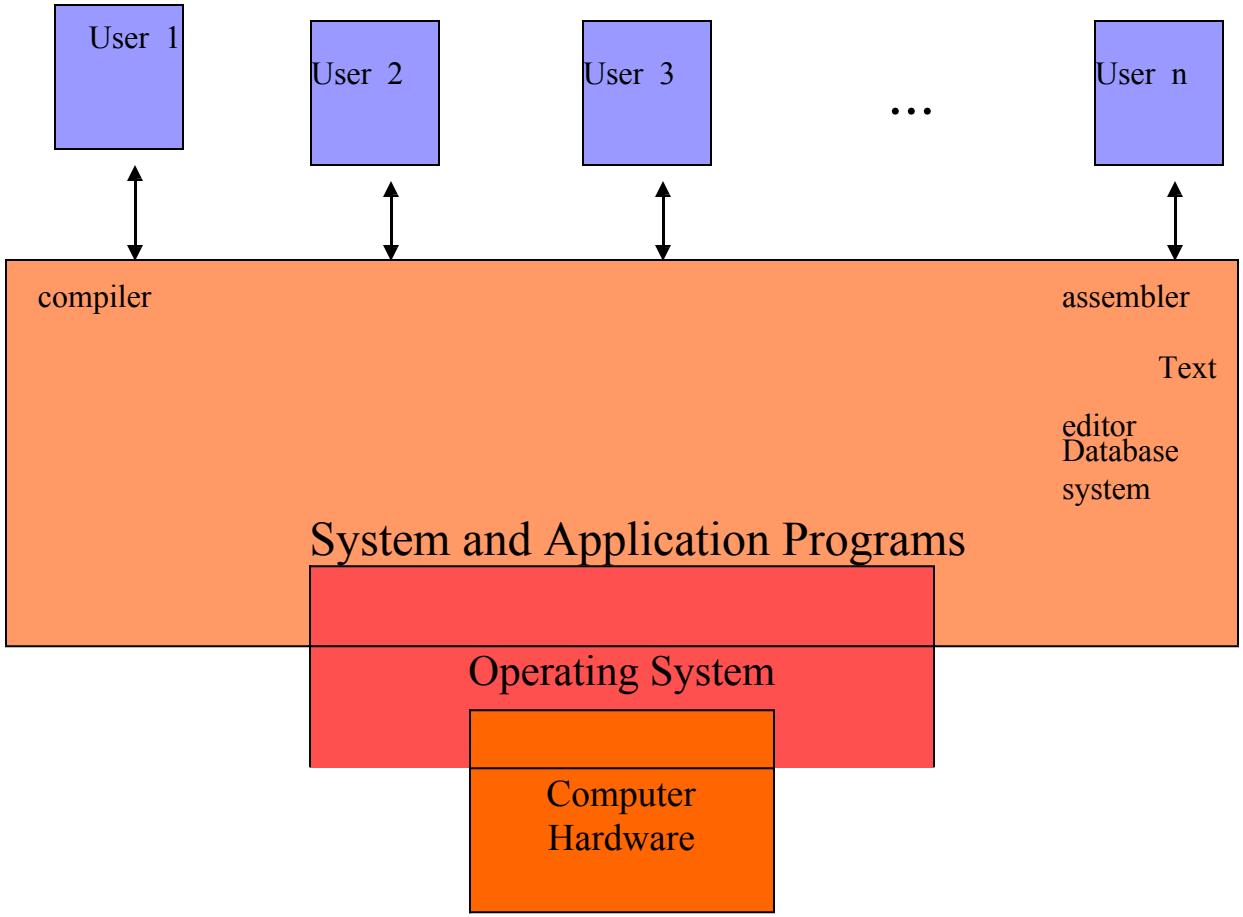
What is an Operating System?

- An OS is a program that acts an intermediary between the user of a computer and computer hardware.
- OS acts as control program as well as resource manager.
- Major cost of general purpose computing is software.
  - OS simplifies and manages the complexity of running application programs efficiently.

Computer System Components

- Hardware
  - Provides basic computing resources (CPU, memory, I/O devices).
- Operating System
  - Controls and coordinates the use of hardware among application programs.
- Application Programs
  - Solve computing problems of users (compilers, database systems, video games, business programs such as banking software).
- Users
  - People, machines, other computers

Abstract View of System



Operating System Views

- Resource allocator
  - to allocate resources (software and hardware) of the computer system and manage them efficiently.
- Control program
  - Controls execution of user programs and operation of I/O devices.
- Kernel
  - The program that executes forever (everything else is an application with respect to the kernel).

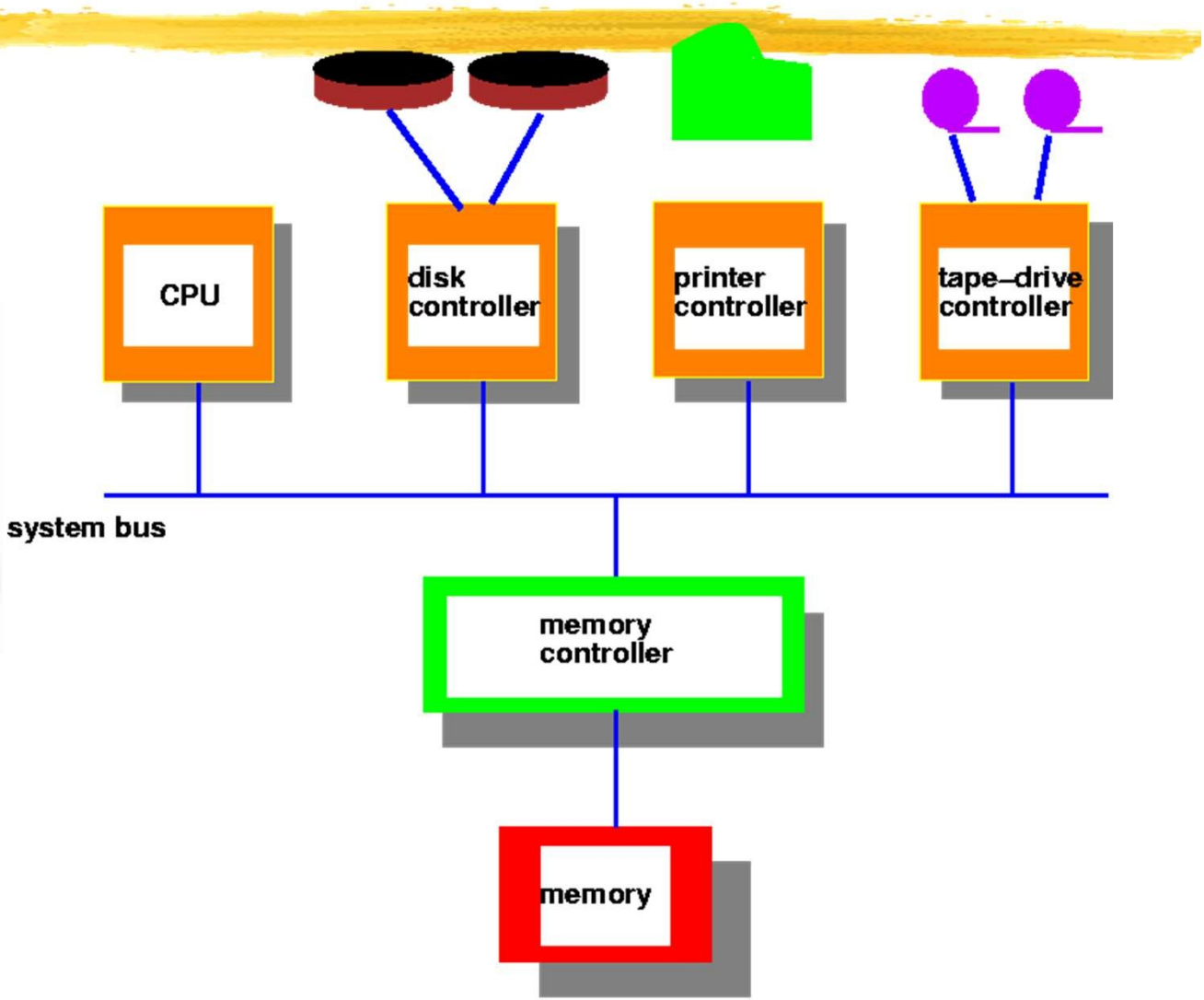
Goals of an Operating System

- Simplify the execution of user programs and make solving user problems easier.
- Use computer hardware efficiently.
  - Allow sharing of hardware and software resources.
- Make application software portable.
- Provide isolation, security and protection among user programs.
- Improve overall system reliability
  - error confinement, fault tolerance, reconfiguration.

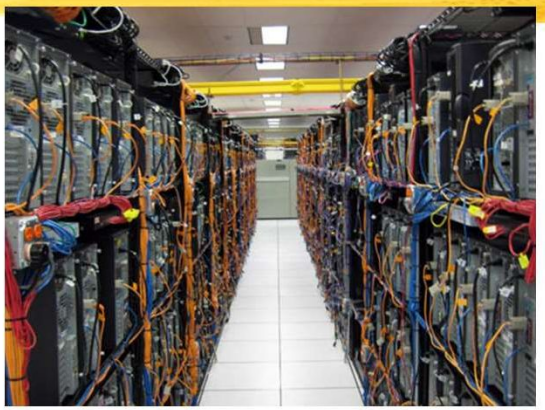
Why should I study Operating Systems?

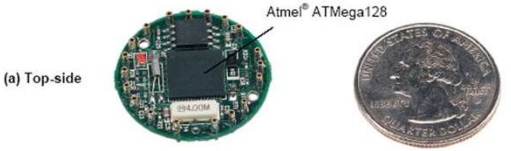
- Need to understand interaction between the hardware and applications
  - New applications, new hardware..
  - Inherent aspect of society today
- Need to understand basic principles in the design of computer systems
  - efficient resource management, security, flexibility
- Increasing need for specialized operating systems
  - e.g. embedded operating systems for devices - cell phones, sensors and controllers
  - real-time operating systems - aircraft control, multimedia services

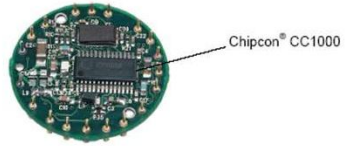
Computer System Architecture (traditional)

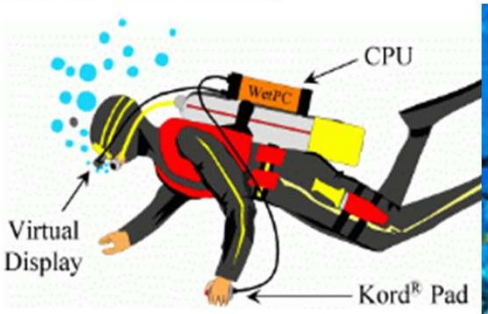


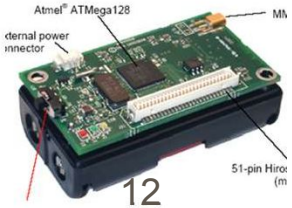
Systems Today











Operating System Spectrum

- Monitors and Small Kernels
  - special purpose and embedded systems, real-time systems
- Batch and multiprogramming
- Timesharing
  - workstations, servers, minicomputers, timeframes
- Transaction systems
- Personal Computing Systems
- Mobile Platforms, devices (of all sizes)

Early Systems - Bare Machine (1950s)

Hardware – *expensive* ; Human – *cheap*

- Structure
  - Large machines run from console
  - Single user system
    - Programmer/User as operator
  - Paper tape or punched cards
- Early software
  - Assemblers, compilers, linkers, loaders, device drivers, libraries of common subroutines.
  - Secure execution
  - Inefficient use of expensive resources
  - Low CPU utilization, high setup time.

From John Ousterhout slides

Simple Batch Systems (1960's)

- Reduce setup time by batching jobs with similar requirements.
- Add a card reader, Hire an operator
  - User is NOT the operator
  - Automatic job sequencing
    - Forms a rudimentary OS.
  - Resident Monitor
    - Holds initial control, control transfers to job and then back to monitor.
  - Problem
    - Need to distinguish job from job and data from program.

From John Ousterhout slides

Supervisor/Operator Control

- Secure monitor that controls job processing
  - Special cards indicate what to do.
  - User program prevented from performing I/O
- Separate user from computer
  - User submits card deck
  - cards put on tape
  - tape processed by operator
  - output written to tape
  - tape printed on printer
- Problems
  - Long turnaround time - up to 2 DAYS!!!
  - Low CPU utilization
    - I/O and CPU could not overlap; slow mechanical devices.

IBM 7094

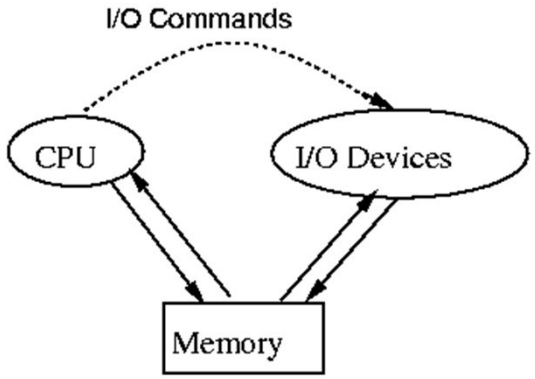
From John Ousterhout slides

Batch Systems - Issues

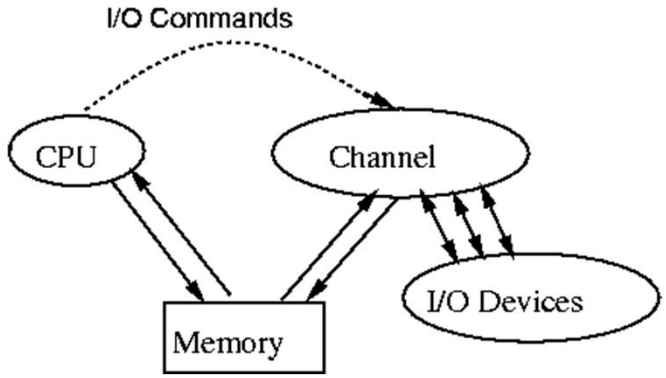
- Solutions to speed up I/O:
- Offline Processing
  - load jobs into memory from tapes, card reading and line printing are done offline.
- Spooling
  - Use disk (random access device) as large storage for reading as many input files as possible and storing output files until output devices are ready to accept them.
  - Allows overlap - I/O of one job with computation of another.
  - Introduces notion of a job pool that allows OS choose next job to run so as to increase CPU utilization.

Speeding up I/O

- Direct Memory Access (DMA)



- Channels



Batch Systems - I/O completion

- How do we know that I/O is complete?
  - Polling:
    - Device sets a flag when it is busy.
    - Program tests the flag in a loop waiting for completion of I/O.
  - Interrupts:
    - On completion of I/O, device forces CPU to jump to a specific instruction address that contains the interrupt service routine.
    - After the interrupt has been processed, CPU returns to code it was executing prior to servicing the interrupt.

Multiprogramming

- Use interrupts to run multiple programs simultaneously
  - When a program performs I/O, instead of polling, execute another program till interrupt is received.
- Requires secure memory, I/O for each program.
- Requires intervention if program loops indefinitely.
- Requires CPU scheduling to choose the next job to run.

Timesharing

Hardware – *getting cheaper*; Human – *getting expensive*

- Programs queued for execution in FIFO order.
- Like multiprogramming, but timer device interrupts after a quantum (timeslice).
  - Interrupted program is returned to end of FIFO
  - Next program is taken from head of FIFO
- Control card interpreter replaced by command language interpreter.

Timesharing (cont.)

- Interactive (action/response)
  - when OS finishes execution of one command, it seeks the next control statement from user.
- File systems
  - online filesystem is required for users to access data and code.
- Virtual memory
  - Job is swapped in and out of memory to disk.

Personal Computing Systems

Hardware – *cheap* ; Human – *expensive*

- Single user systems, portable.
- I/O devices - keyboards, mice, display screens, small printers.
- Laptops and palmtops, Smart cards, Wireless devices.
- Single user systems may not need advanced CPU utilization or protection features.
- Advantages:
  - user convenience, responsiveness, ubiquitous

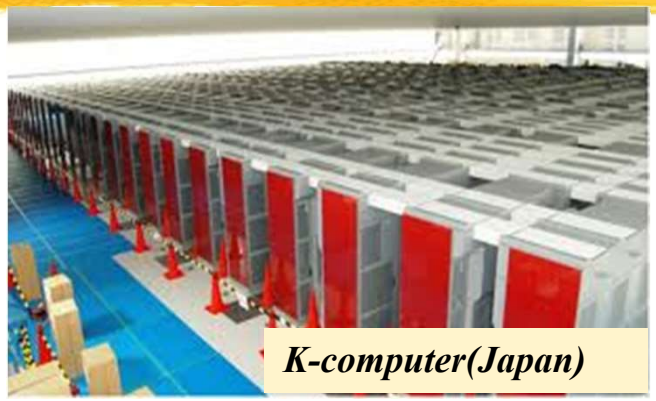
Parallel Systems

- Multiprocessor systems with more than one CPU in close communication.
- Improved Throughput, economical, increased reliability.
- Kinds:
  - Vector and pipelined
  - Symmetric and asymmetric multiprocessing
  - Distributed memory vs. shared memory
- Programming models:
  - Tightly coupled vs. loosely coupled ,message-based vs. shared variable

Parallel Computing Systems

ILLIAC 2 (UIllinois)

Climate modeling, earthquake simulations, genome analysis, protein folding, nuclear fusion research, …..



K-computer(Japan)

Connection Machine (MIT)

Tianhe-1(China)

IBM Blue Gene

Distributed Systems

Hardware – *very cheap* ; Human – *very expensive*

- Distribute computation among many processors.
- Loosely coupled -
  - no shared memory, various communication lines
- client/server architectures
- Advantages:
  - resource sharing
  - computation speed-up
  - reliability
  - communication - e.g. email
- Applications - digital libraries, digital multimedia

Real-time systems

- Correct system function depends on timeliness
- Feedback/control loops
- Sensors and actuators
- Hard real-time systems -
  - Failure if response time too long.
  - Secondary storage is limited
- Soft real-time systems -
  - Less accurate if response time is too long.
  - Useful in applications such as multimedia, virtual reality.

Storage Structure

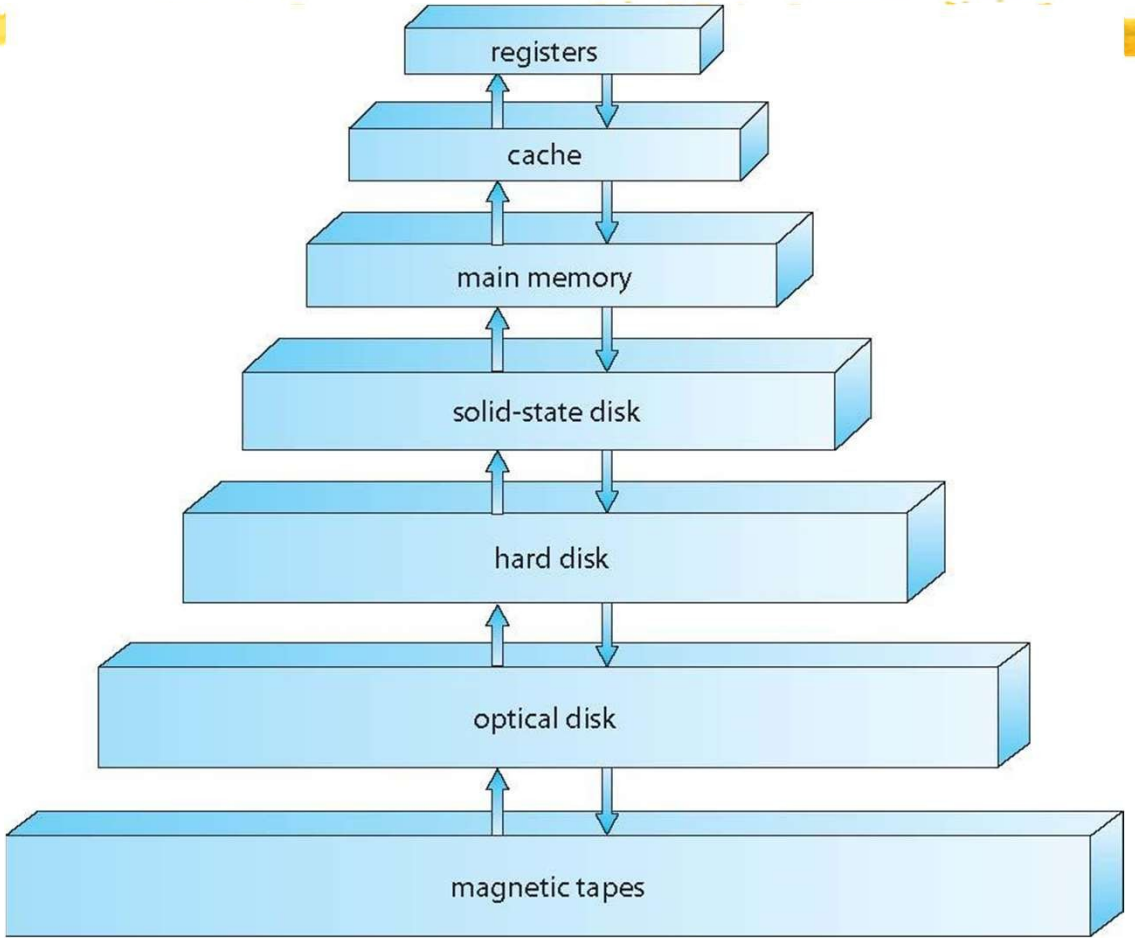
- Main memory - only large storage media that the CPU can access directly.
- Secondary storage - has large nonvolatile storage capacity.
  - Magnetic disks - rigid metal or glass platters covered with magnetic recording material.
    - Disk surface is logically divided into tracks, subdivided into sectors.
    - Disk controller determines logical interaction between device and computer.

Storage Hierarchy

- Storage systems are organized in a hierarchy based on
  - Speed
  - Cost
  - Volatility

- Caching - process of copying information into faster storage system; main memory can be viewed as fast cache for secondary storage.

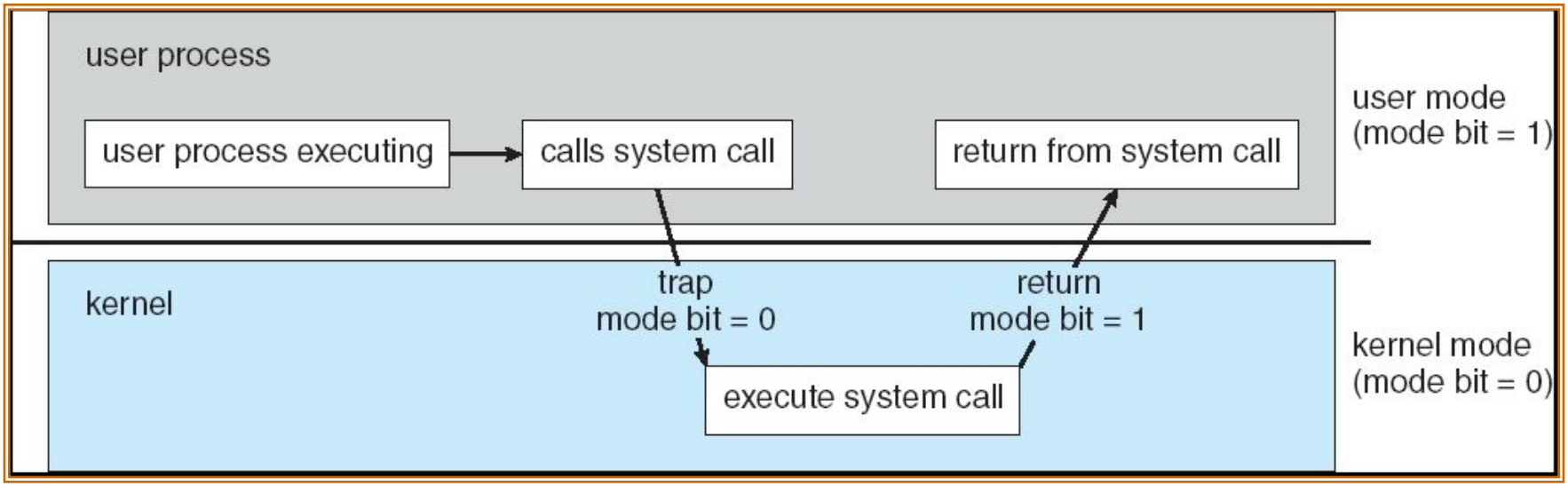
Storage Device Hierarchy



System Calls

We use **system calls** to provide an interface between a process and **OS**.
A **system call** helps a program request services from the kernel.

- User code can issue a syscall, which causes a trap
- Kernel handles the syscall



Traps are occurred by the user program to invoke the functionality of the OS

Image Source : Operating Systems Concepts By Silberschatz & Galvin

System Calls

- Interface between applications and the OS.
  - Application uses an assembly instruction to trap into the kernel
  - Some higher level languages provide wrappers for system calls (e.g., C)
- System calls pass parameters between an and OS via registers or memory
  .
- Linux has about 300 system calls
  - read(), write(), open(), close(), fork(), exec(), ioctl(),…..

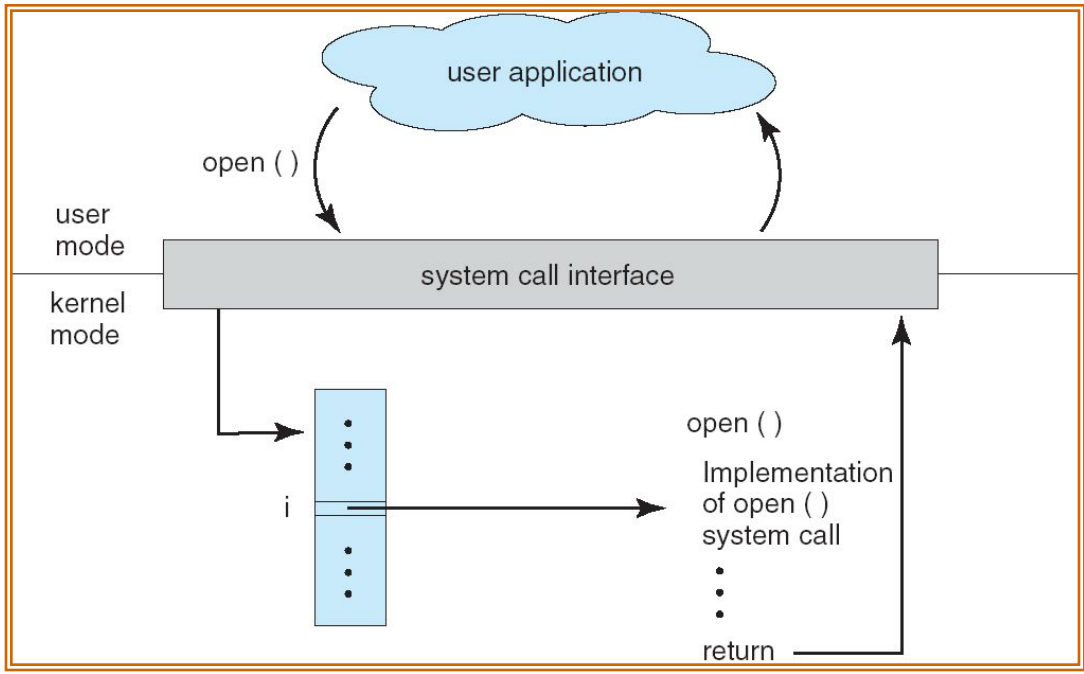


Image Source : Operating Systems Concepts By Silberschatz & Galvin

OS Task: Process Management

- Process - fundamental concept in OS
  - Process is an instance of a program in execution.
  - Process needs resources - CPU time, memory, files/data and I/O devices.
- OS is responsible for the following process management activities.
  - Process creation and deletion
  - Process suspension and resumption
  - Process synchronization and interprocess communication
  - Process interactions - deadlock detection, avoidance and correction

OS Task: Memory Management

- Main Memory is an array of addressable words or bytes that is quickly accessible.
- Main Memory is volatile.
- OS is responsible for:
  - Allocate and deallocate memory to processes.
  - Managing multiple processes within memory - keep track of which parts of memory are used by which processes. Manage the sharing of memory between processes.
  - Determining which processes to load when memory becomes available.

OS Task: Secondary Storage and I/O Management

- Since primary storage (i.e., main memory) is expensive and volatile, secondary storage is required for backup.
- Disk is the primary form of secondary storage.
  - OS performs storage allocation, free-space management, etc.
- I/O system in the OS consists of
  - Device driver interface that abstracts device details
  - Drivers for specific hardware devices

OS Task: File System Management

- File is a collection of related information - represents programs and data.

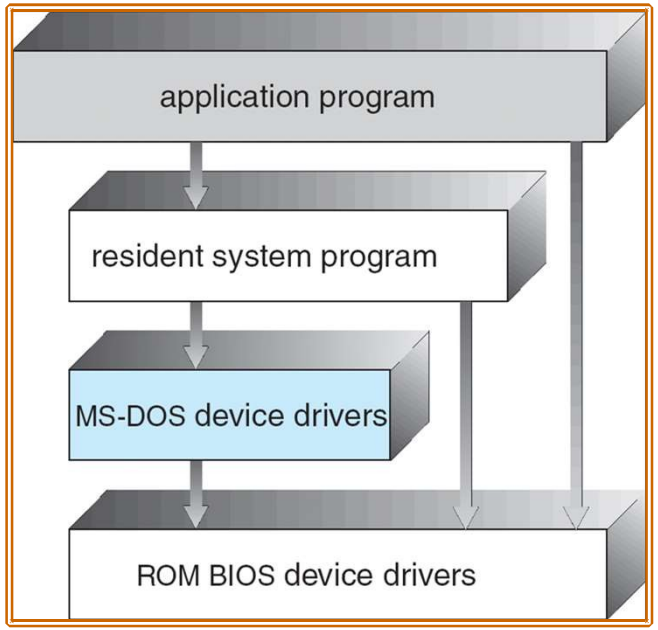
- OS is responsible for
  - File creation and deletion
  - Directory creation and deletion
  - Supporting primitives for file/directory manipulation.
  - Mapping files to disks (secondary storage).

OS Task: Protection and Security

- Protection mechanisms control access of processes to user and system resources.
- Protection mechanisms must:
  - Distinguish between authorized and unauthorized use.
  - Specify access controls to be imposed on use.
  - Provide mechanisms for enforcement of access control.

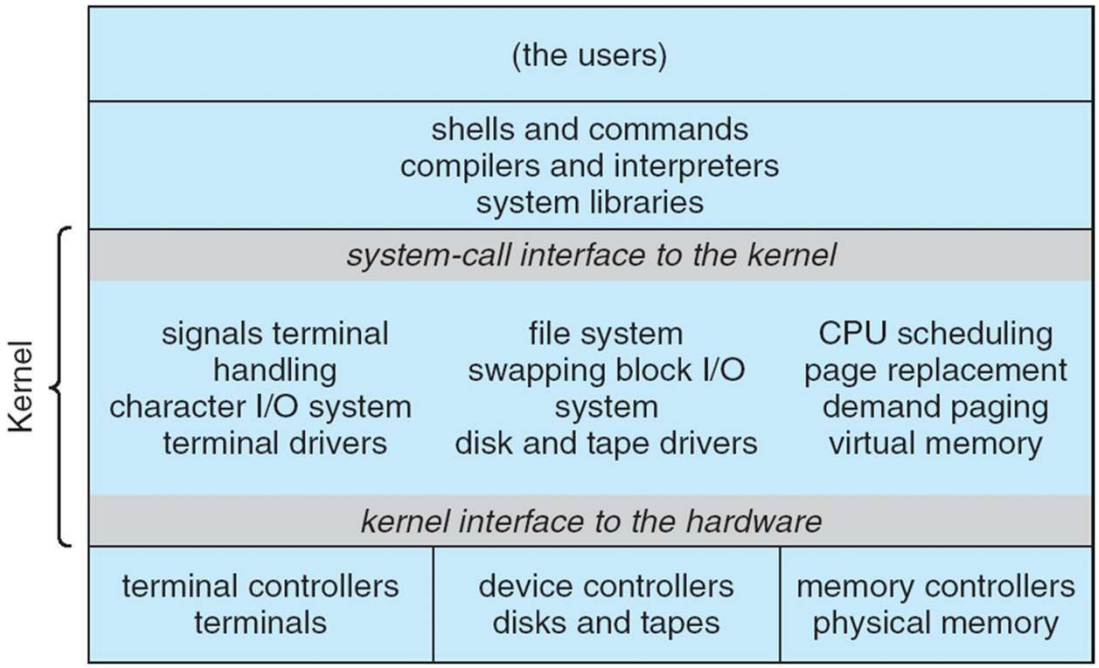
OS Structure - Simple Approach

- MS-DOS - provides a lot of functionality in little space.
  - Not divided into modules, Interfaces and levels of functionality are not well separated



Original UNIX System Structure

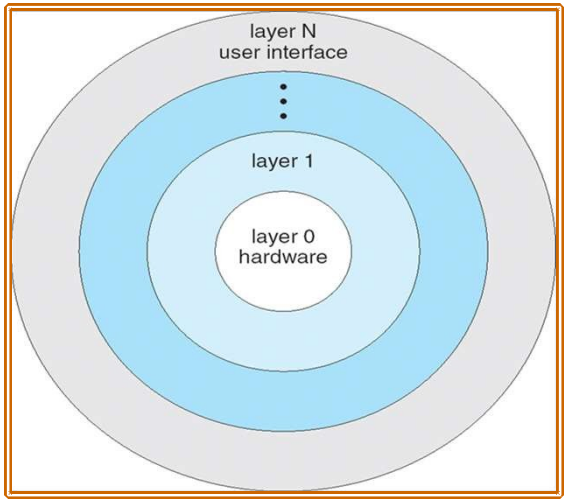
- Limited structuring, has 2 separable parts
  - Systems programs
  - Kernel
    - everything below system call interface and above physical hardware.
    - Filesystem, CPU scheduling, memory management



Layered OS Structure

- OS divided into number of layers - bottom layer is hardware, highest layer is the user interface.
- Each layer uses functions and services of only lower-level layers.
- THE Operating System and Linux Kernel has successive layers of abstraction.

User Programs
Interface Primitives
Device Drivers and Schedulers
Virtual Memory
I/O
CPU Scheduling
Hardware



Monolithic Kernel

In monolithic kernel, both user services and kernel services are kept in the same address space.

Monolithic kernel is larger than microkernel.

It executes quickly in comparison to microkernel.

It is difficult to extend a monolithic kernel.

If a service crashes, the entire system crashes when a monolithic kernel is used.
Less code is required to build a monolithic kernel.

Examples of monolithic kernel include: Linux, BSDs (FreeBSD, OpenBSD, NetBSD), OS-9, AIX, HP-UX, DOS, OpenVMS, XTS-400, Microsoft Windows (95,98,Me), and Solaris.

Monolithic Kernel (Contd)..

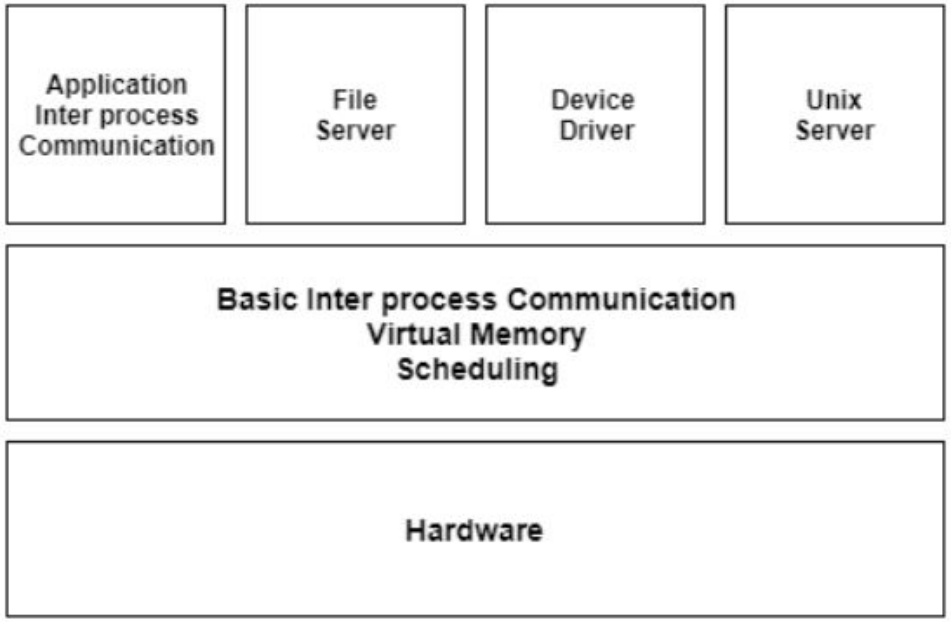
- Benefits:
  - Easier to extend a microkernel
  - Easier to port OS to new architectures
  - More reliable and more secure (less code is running in kernel mode)

- Detriments:
  - Performance overhead severe for naïve implementation

MicroKernel

A microkernel is the minimum software that is required to correctly implement an operating system.

This includes memory, process scheduling mechanisms and basic inter-process communication.



Microkernel Based Operating System

MicroKernel (Contd) ..

- The microkernel contains basic requirements such as memory, process scheduling mechanisms and basic interprocess communication.

- The only software executing at the privileged level i.e. kernel mode is the microkernel. The other functions of the operating system are removed from the kernel mode and run in the user mode. These functions may be device drivers, file servers, application interprocess communication etc.

- The microkernel makes sure that the code can be easily managed because the services are divided in the user space. This means that there is less code running in the kernel mode which results in increased security and stability.

MicroKernel (Contd) ..

Benefits of Microkernels

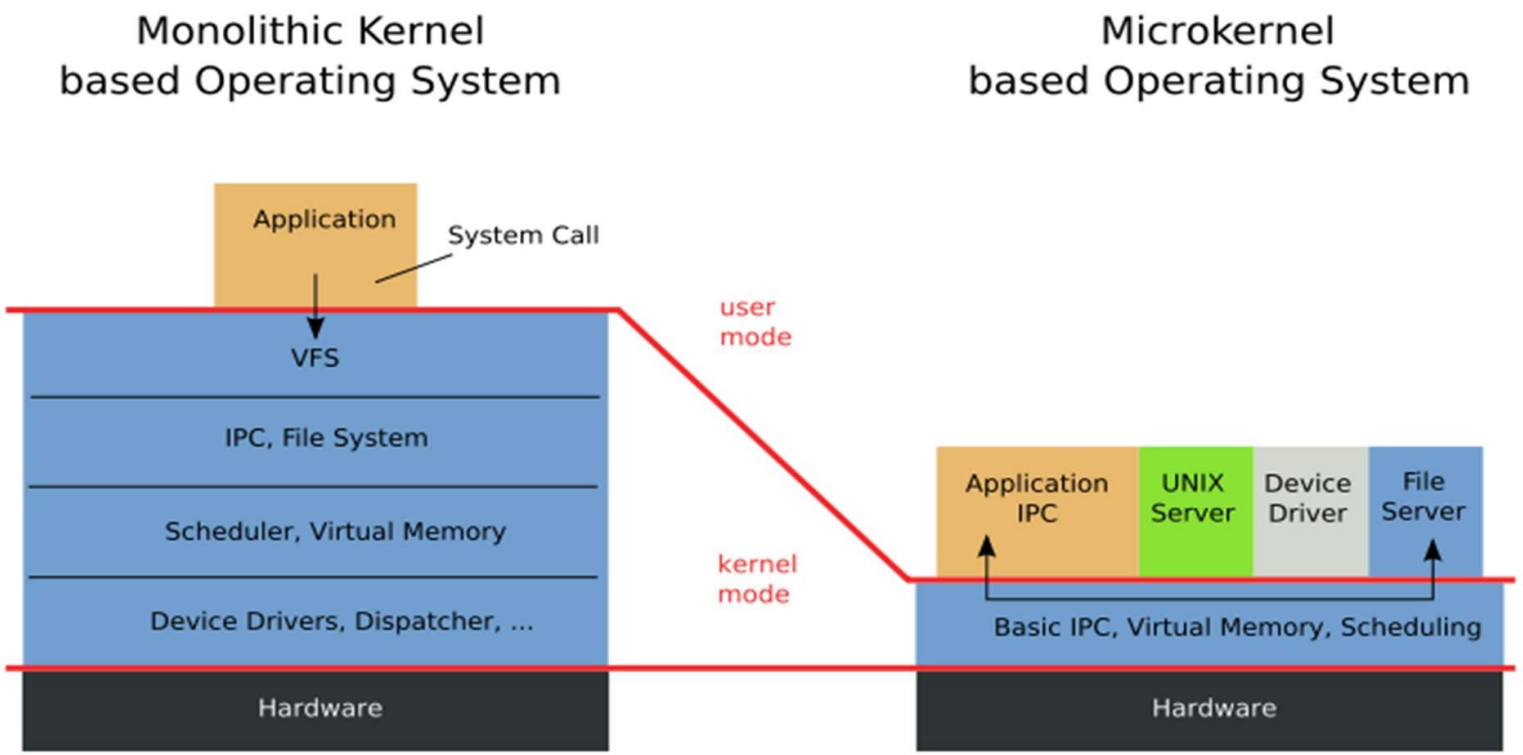
- Microkernels are modular and the different modules can be replaced, reloaded, modified, changed etc. as required. This can be done without even touching the kernel.

- Microkernels are quite secure as only those components are included that would disrupt the functionality of the system otherwise.

- Microkernels contain fewer system crashes as compared to monolithic systems. Also, the crashes that do occur can be handled quite easily due to the modular structure of microkernels.

Monolithic Vs Microkernel OS

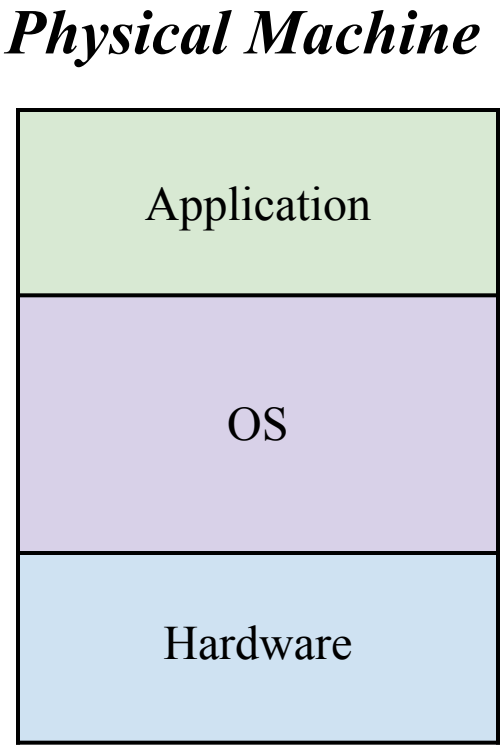
A microkernel is a kernel type that implements an operating system by providing methods, including low-level address space management, IPC, and thread management. On the other hand, a monolithic kernel is a type of kernel in which the complete OS runs in the kernel space.



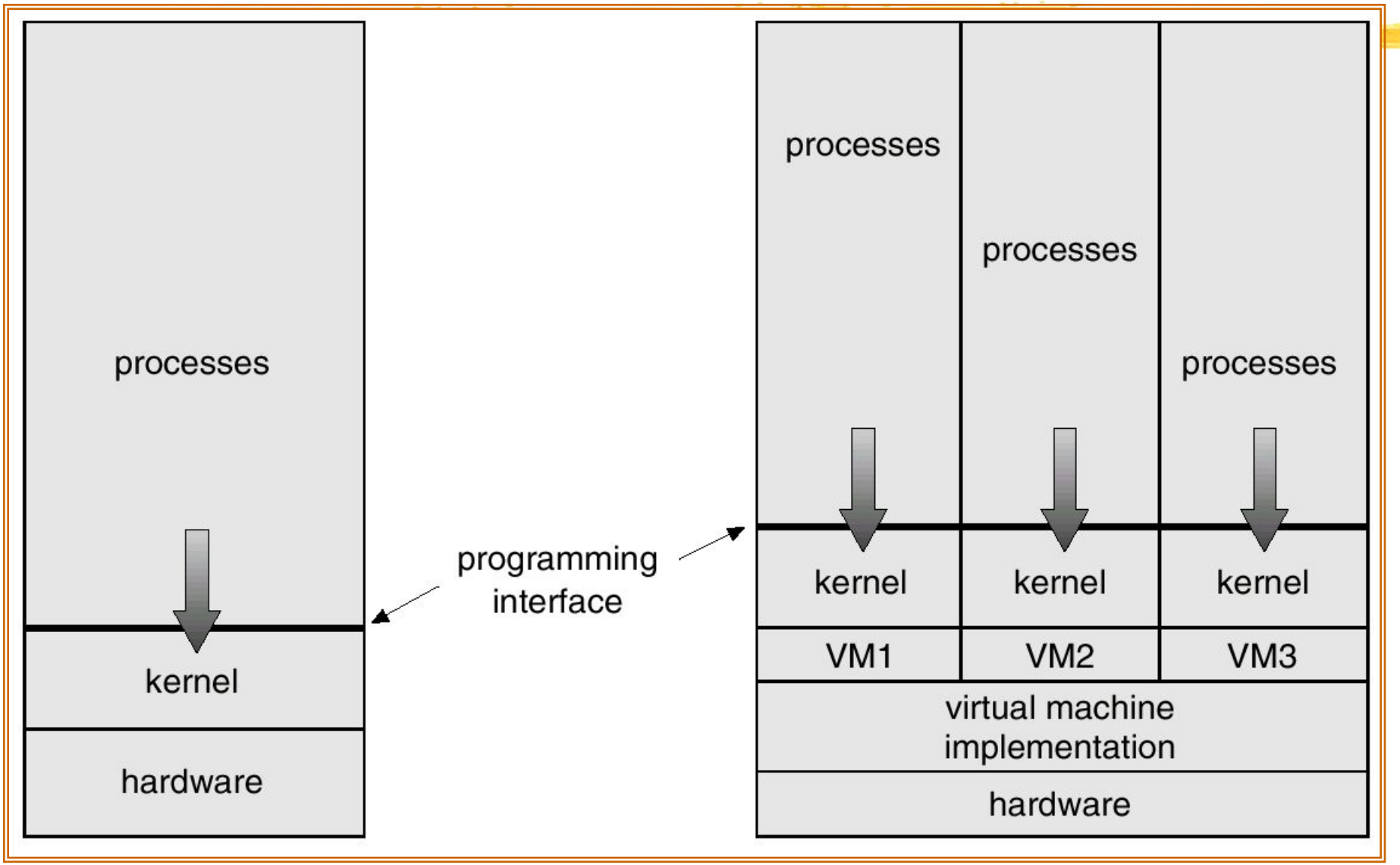
Slide adapted from http://web.cecs.pdx.edu/~walpole/class/cs533/fall2015/home.html

Virtualization

- Virtualization is the creation of a virtual (rather than actual) version of something, such as an **operating system**, a **server**, a **storage device** or **network resources**.



System Models



Virtual Machines