EECS 111:

System Software

Lecture: Introduction

Prof. Mohammad Al Faruque

The Henry Samueli School of Engineering
Electrical Engineering & Computer Science
University of California Irvine (UCI)

Lecture: outline

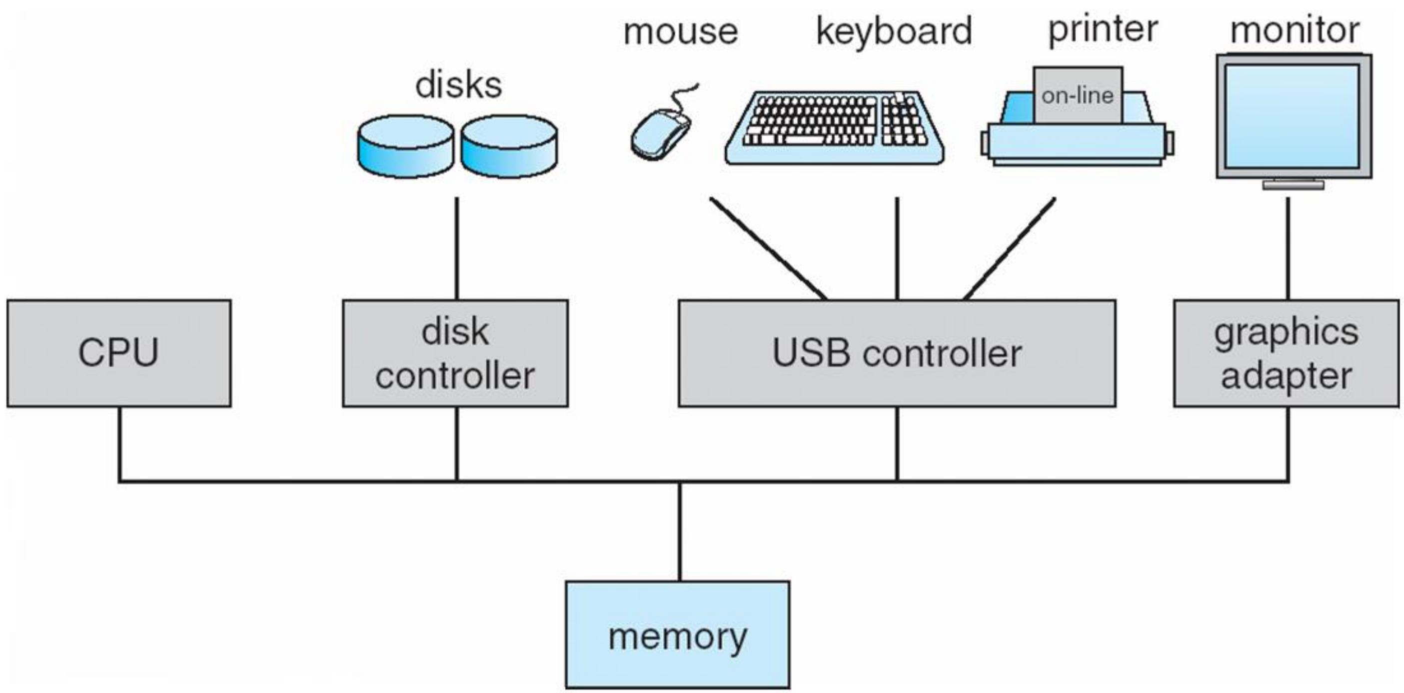
- Course Management
- What is System Software?
- Open Discussion
- Introduction to System Software (OS in general)

Computer System & OS Structures

- Computer System Operation
  - I/O Structure
  - Storage Structure, Storage Hierarchy
  - Hardware Protection
- Operating System Services, System calls, System Programs
- Structuring OS
  - Virtual Machine Structure and Organization
- OS Design and Implementation
  - Process Management, Memory Management, Secondary Storage Management, I/O System Management, File Management, Protection System, Networking, Command-Interpreter.

Computer System Organization

- **Computer-system operation**
  - **One or more CPUs, device controllers connect through common bus providing access to shared memory**
  - **Concurrent execution of CPUs and devices competing for memory cycles**



Computer-System Operation

- **I/O devices and the CPU can execute concurrently**
- **Each device controller is in charge of a particular device type**
- **Each device controller has a local buffer**
- **CPU moves data from/to main memory to/from local buffers**
- **I/O is from the device to local buffer of controller**
- **Device controller informs CPU that it has finished its operation by raising an interrupt**

Common Functions of Interrupts

- **Interrupt transfers control to the interrupt service routine, generally through the interrupt vector, which contains the addresses of all the service routines**
- **Interrupt architecture must save the address of the interrupted instruction**
- **Incoming interrupts are *disabled* while another interrupt is being processed to prevent a *lost interrupt***
- **An operating system is interrupt driven**
- **Note:**
  <u>A *trap* is a software-generated interrupt caused either by an error or a user request (system-call)</u>

Interrupt Handling

- **The operating system preserves the state of the CPU by storing registers and the program counter on the stack**
- **The OS then determines which type of interrupt has occurred by one of two schemes:**
  - **polling**
  - **vectored interrupt system**
- **Separate segments of code determine what action should be taken for each type of interrupt**

Vectored Interrupt

- **In a computer, a vectored interrupt is an I/O interrupt that tells the part of the computer that handles I/O interrupts at the hardware level that a request for attention from an I/O device has been received and also identifies the device that sent the request.**
  - 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.

Interrupt Number	Address
0	0003h
1	000Bh
2	0013h
3	001Bh
4	0023h
5	002Bh
6	0033h
7	003Bh
8	0043h
9	004Bh
10	0053h
11	005Bh
12	0063h
13	006Bh
14	0073h
15	007Bh

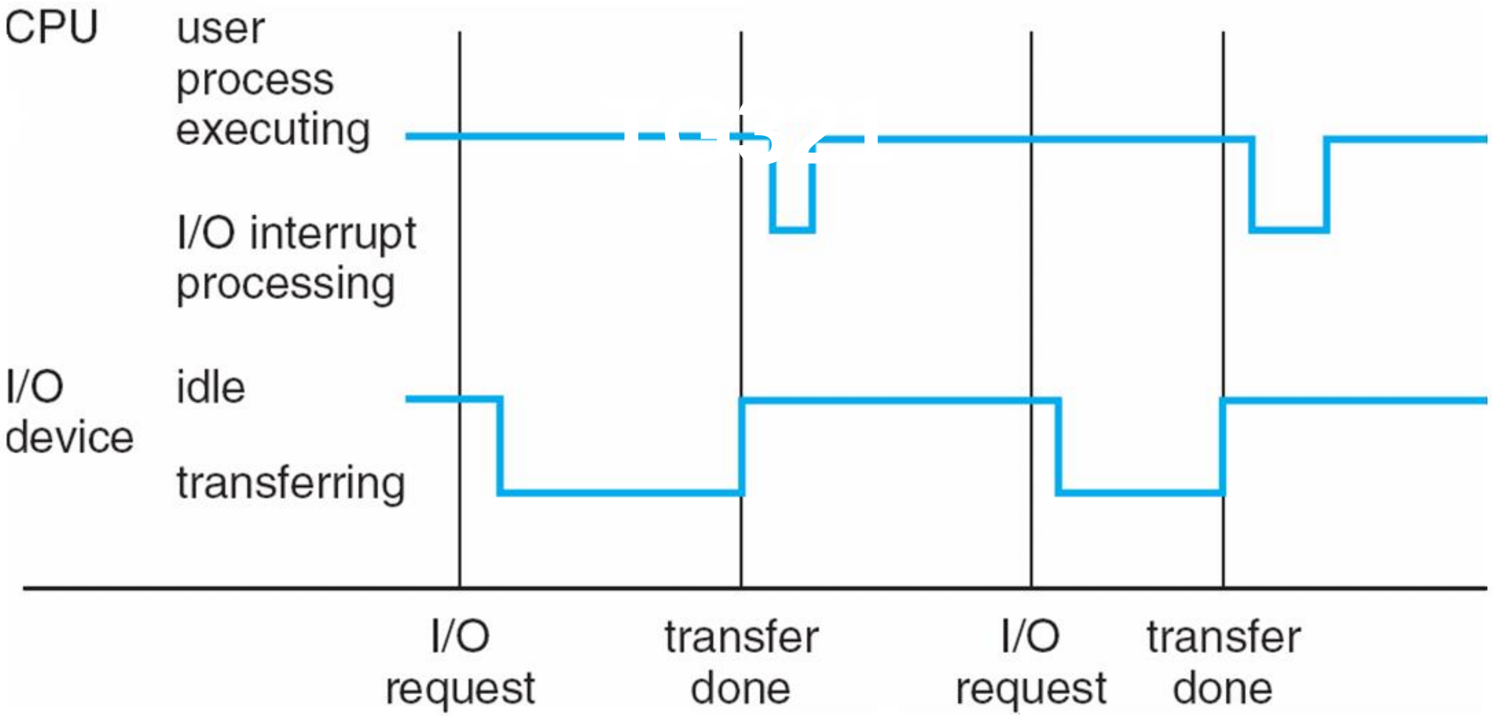
Interrupt Number	Address
16	0083h
17	008Bh
18	0093h
19	009Bh
20	00A3h
21	00ABh
22	00B3h
23	00BBh
24	00C3h
25	00CBh
26	00D3h
27	00DBh
28	00E3h
29	00EBh
30	00F3h
31	00FBh

Polling Interrupt

- **In a polled interrupt, it requires that the interrupt handler poll or send a signal to each device in turn in order to find out which one sent the interrupt request.**
  - Device sets a flag when it is busy.
  - Program tests the flag in a loop waiting for completion of I/O.

Interrupt Timeline

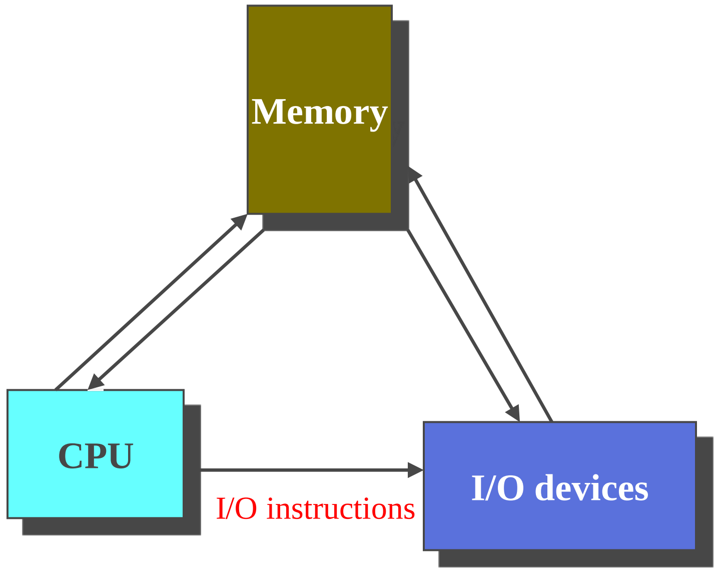
- Interrupt time line for a single process doing output
  - Process issues I/O request
  - I/O device signals "transfer done" via interrupt



I/O Structure

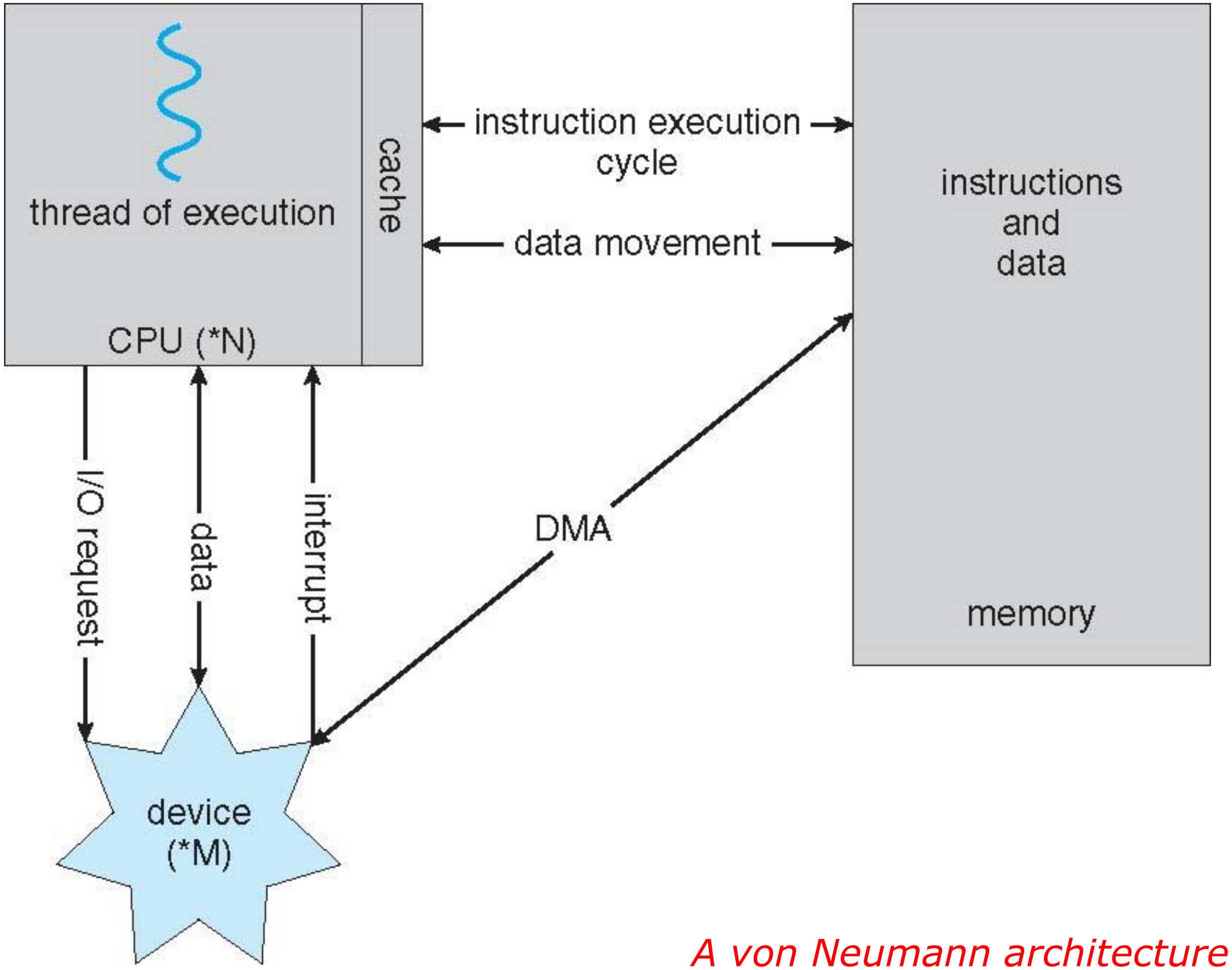
- ***Synchronous* I/O (blocking I/O):**
  after I/O starts, control returns to user program only upon I/O completion
  - *wait* instruction idles CPU until next interrupt
  - no simultaneous I/O processing, at most one outstanding I/O request at a time.
- ***Asynchronous* I/O (non-blocking I/O):**
  after I/O starts, control returns to user program without waiting for I/O completion
  - System call
  - Device Status table - holds type, address and state for each device
  - OS indexes into I/O device table to determine device status and modify table entry to include interrupt.

Direct Memory Access Structure



- **DMA: Direct Memory Access**
- **Used for high-speed I/O devices able to transmit information at close to memory speeds**
- **Device controller transfers blocks of data from buffer storage directly to main memory without CPU intervention**
- **Only one interrupt is generated per block, rather than the one interrupt per byte**

How a Modern Computer Works?



A von Neumann architecture

Computer System & OS Structures

- **Computer System Operation**
  - **I/O Structure**
  - **Storage Structure, Storage Hierarchy**
  - **Hardware Protection**
- **Operating System Services, System calls, System Programs**
- **Structuring OS**
  - **Virtual Machine Structure and Organization**
- **OS Design and Implementation**
  - Process Management, Memory Management, Secondary Storage Management, I/O System Management, File Management, Protection System, Networking, Command-Interpreter.

Storage Structure

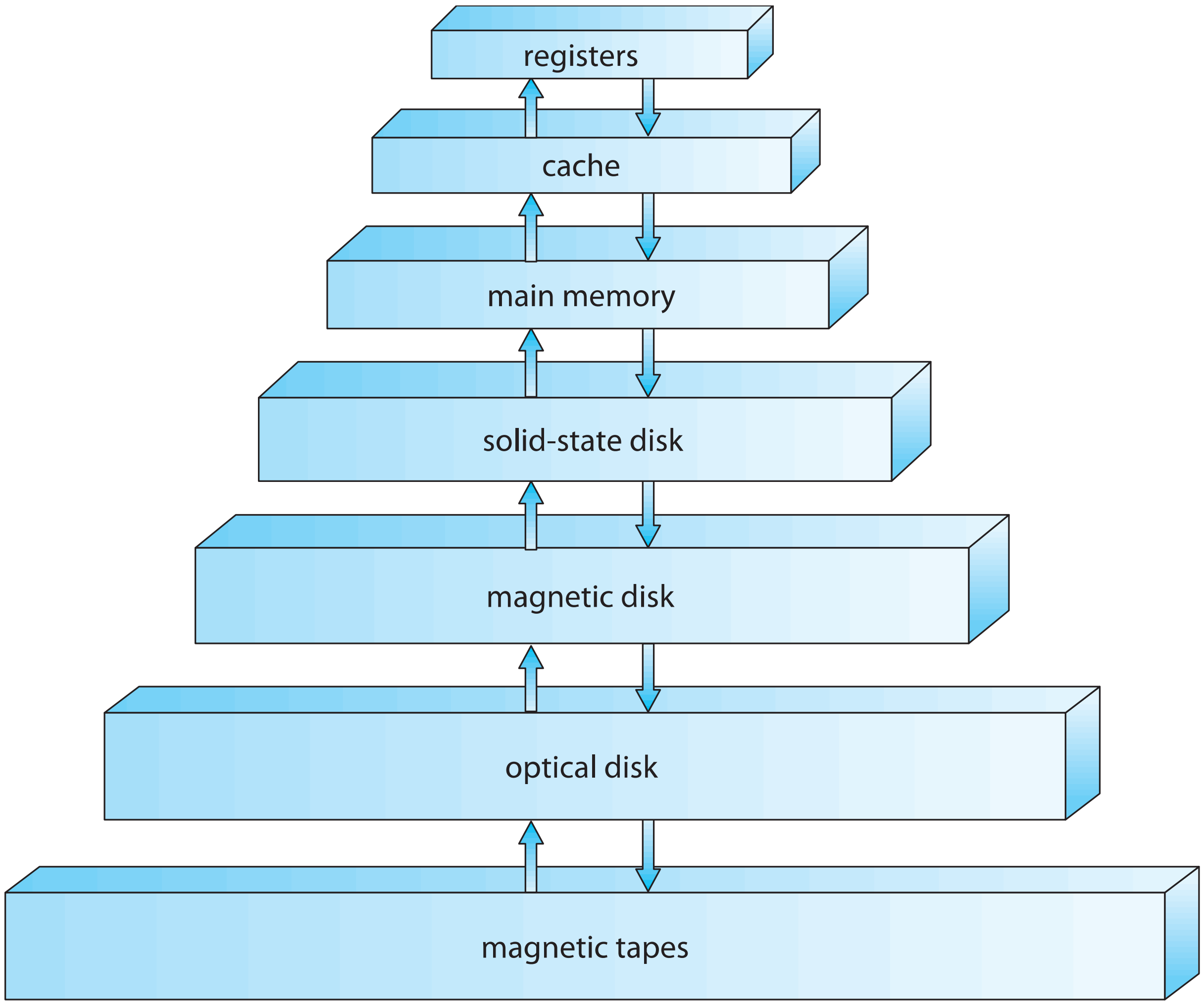
- **Main memory** – only large storage media that the CPU can access directly
  - Random access
  - Typically volatile
- **Secondary storage** – extension of main memory that provides large nonvolatile storage capacity
- **Magnetic disks** – rigid metal or glass platters covered with magnetic recording material
  - Disk surface is logically divided into tracks, which are subdivided into sectors
  - The disk controller determines the logical interaction between the device and the computer
- **Solid-state disks** – faster than magnetic disks, nonvolatile
  - Various technologies
  - Becoming more popular

Storage Hierarchy

- **Storage systems organized in hierarchy based on**
  - **Speed**
  - **Cost**
  - **Volatility**

- **Caching is often used between storage systems**
  - **transparently copying information into faster storage system (e.g. CPU cache holds most-recently used data from main memory)**
  - **main memory can be viewed as a *cache* for secondary storage**

Storage-Device Hierarchy



Computer System & OS Structures

- Computer System Operation
  - I/O Structure
  - Storage Structure, Storage Hierarchy
  - Hardware Protection
- Operating System Services, System calls, System Programs
- Structuring OS
  - Virtual Machine Structure and Organization
- OS Design and Implementation
  - Process Management, Memory Management, Secondary Storage Management, I/O System Management, File Management, Protection System, Networking, Command-Interpreter.

Hardware Protection

- Dual Mode Operation
- I/O Protection
- Memory Protection
- CPU Protection

Dual-mode operation

- **Sharing system resources requires operating system to ensure that an incorrect program cannot cause other programs to execute incorrectly.**

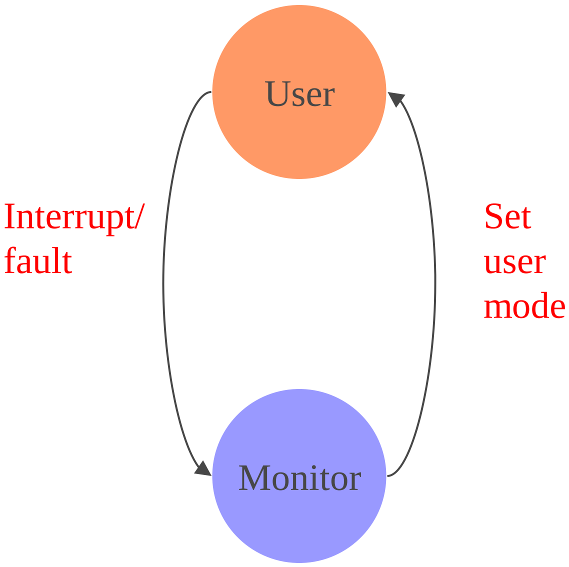
- **Provide hardware support to differentiate between at least two modes of operation:**

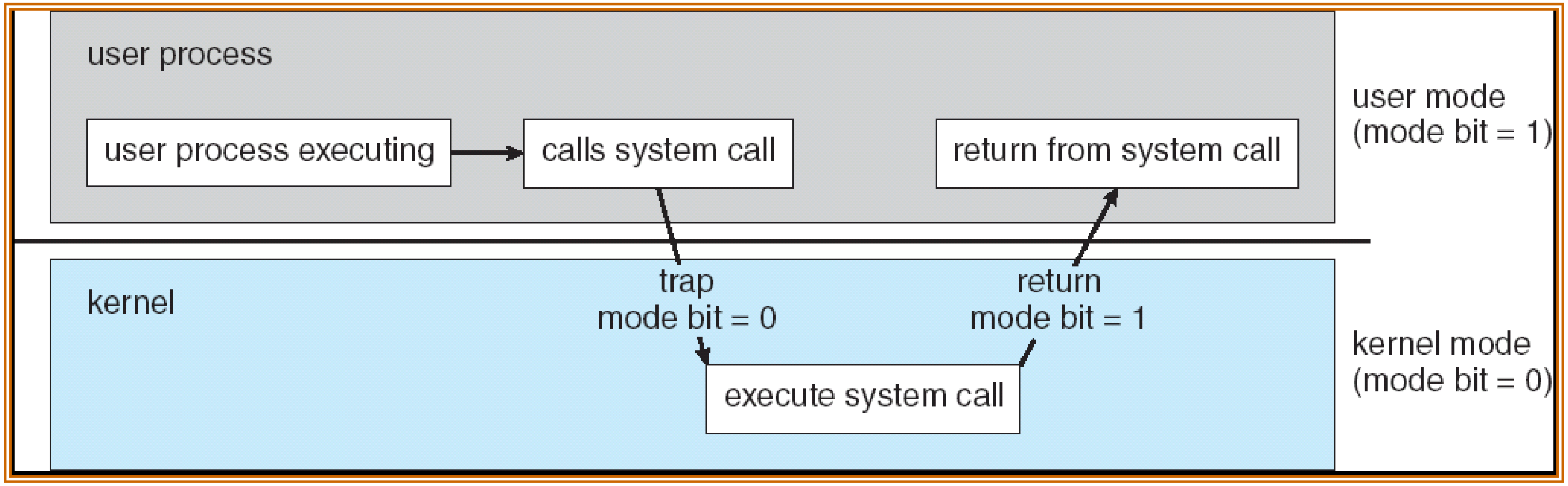
  1. User mode -- execution done on behalf of a user.

  2. Monitor mode (supervisor/kernel/system mode) -- execution done on behalf of operating system.

Dual-mode operation (cont.)

- Mode bit added to computer hardware to indicate the current mode: monitor(0) or user(1).
- When an interrupt or fault occurs, hardware switches to monitor mode.
- Privileged instructions only in monitor mode.





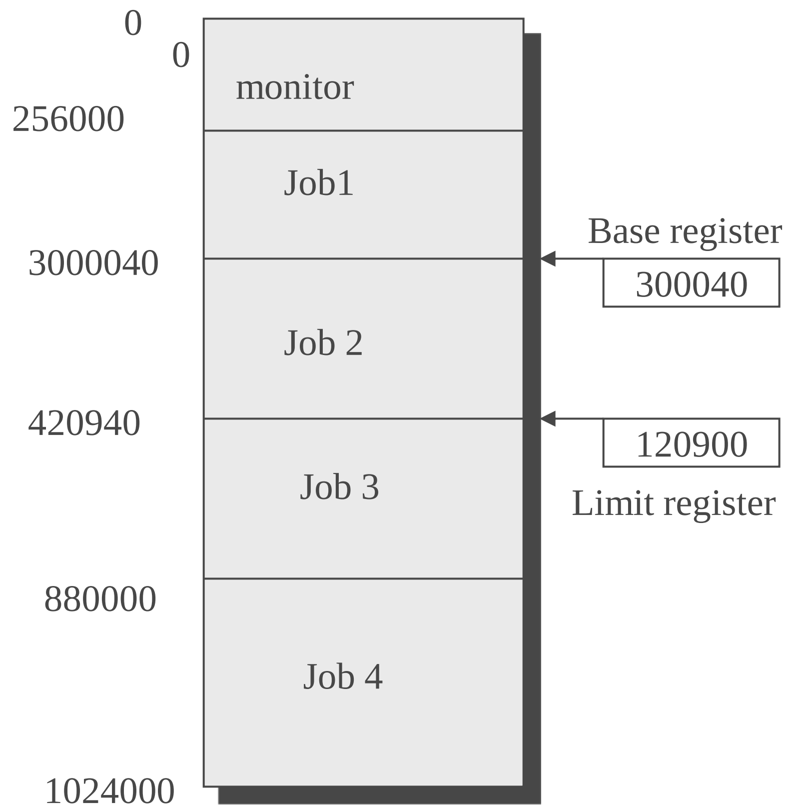
I/O Protection

- **All I/O instructions are privileged instructions.**

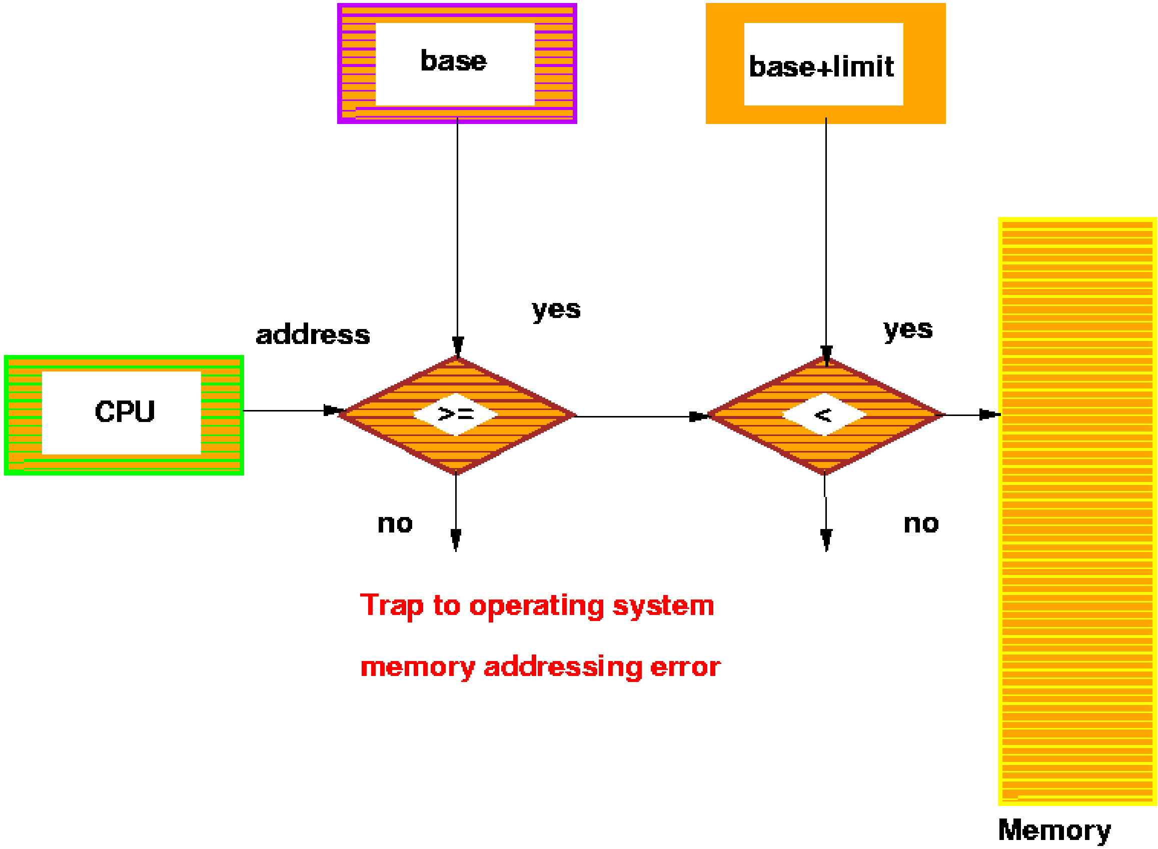
- **Must ensure that a user program could never gain control of the computer in monitor mode, for e.g. a user program that as part of its execution, stores a new address in the interrupt vector.**

Memory Protection

- Must provide memory protection at least for the interrupt vector and the interrupt service routines.
- To provide memory protection, add two registers that determine the range of legal addresses a program may address.
  - Base Register - holds smallest legal physical memory address.
  - Limit register - contains the size of the range.
- Memory outside the defined range is protected.
- When executing in monitor mode, the OS has unrestricted access to both monitor and users' memory.



Hardware Address Protection

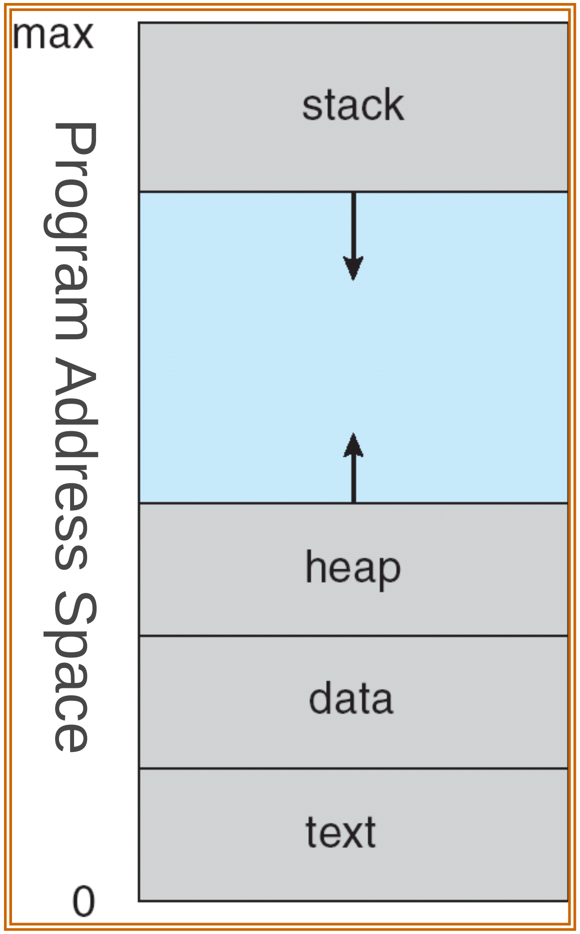


•

The load instructions for the base and limit registers are privileged instructions.

More detail: A Program's Address Space

- Address space ⇒ the set of accessible addresses + state associated with them:
  - For a 32-bit processor there are 2^{32} = 4 billion addresses
- What happens when you read or write to an address?
  - Perhaps Nothing
  - Perhaps acts like regular memory
  - Perhaps ignores writes
  - Perhaps causes I/O operation
    - (Memory-mapped I/O)
  - Perhaps causes exception (fault)



CPU Protection

- **Timer - interrupts computer after specified period to ensure that OS maintains control.**
  - **Timer is decremented every clock tick.**
  - **When timer reaches a value of 0, an interrupt occurs.**
- **Timer is commonly used to implement time sharing.**
- **Timer is also used to compute the current time.**
- **Load timer is a privileged instruction.**

Computer System & OS Structures

- **Computer System Operation**
  - **I/O Structure**
  - **Storage Structure, Storage Hierarchy**
  - **Hardware Protection**
- **Operating System Services, System calls, System Programs**
- **Structuring OS**
  - **Virtual Machine Structure and Organization**
- **OS Design and Implementation**
  - Process Management, Memory Management, Secondary Storage Management, I/O System Management, File Management, Protection System, Networking, Command-Interpreter.

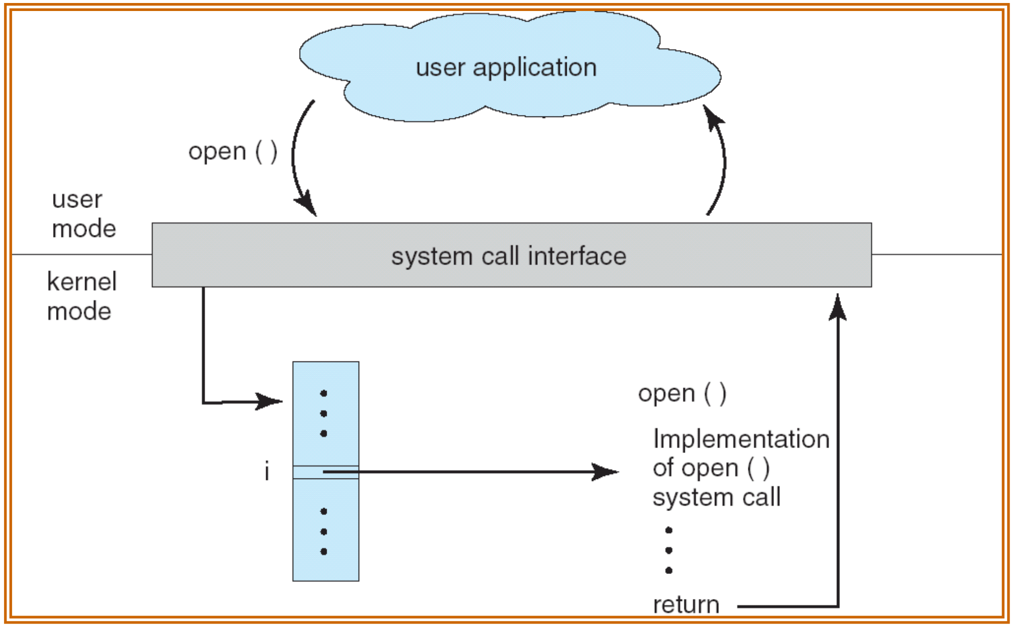
General System Architecture

- **Given the I/O instructions are privileged, how do users perform I/O?**

- **Via system calls - the method used by a process to request action by the operating system.**

System Calls

- **Interface between running program and the OS.**
  - **Assembly language instructions (macros and subroutines)**
  - **Some higher level languages allow system calls to be made directly (e.g. C)**
- **Passing parameters between a running program and OS via registers, memory tables or stack.**
- **Unix has about 32 system calls**
  - **read(), write(), open(), close(), fork(), exec(), ioctl(),…..**



Operating System Services

- **Services that provide user-interfaces to OS**
  - Program execution - load program into memory and run it
  - I/O Operations - since users cannot execute I/O operations directly
  - File System Manipulation - read, write, create, delete files
  - Communications - interprocess and intersystem
  - Error Detection - in hardware, I/O devices, user programs

- **Services for providing efficient system operation**
  - Resource Allocation - for simultaneously executing jobs
  - Accounting - for account billing and usage statistics
  - Protection - ensure access to system resources is controlled

System Programs

- **Convenient environment for program development and execution. User view of OS is defined by system programs, not system calls.**
  - Command Interpreter (sh, csh, ksh) - parses/executes other system programs
  - File manipulation - copy (cp), print (lpr), compare(cmp, diff)
  - File modification - editing (ed, vi, emacs)
  - Application programs - send mail (mail), read news (rn)
  - Programming language support (cc)
  - Status information, communication
  - etc….

Command Interpreter System

- Commands that are given to the operating system via command statements that execute
  - Process creation and deletion, I/O handling, Secondary Storage Management, Main Memory Management, File System Access, Protection, Networking.
- Obtains the next command and executes it.
- Programs that read and interpret control statements also called -
  - Control card interpreter, command-line interpreter, shell (in UNIX)

Computer System & OS Structures

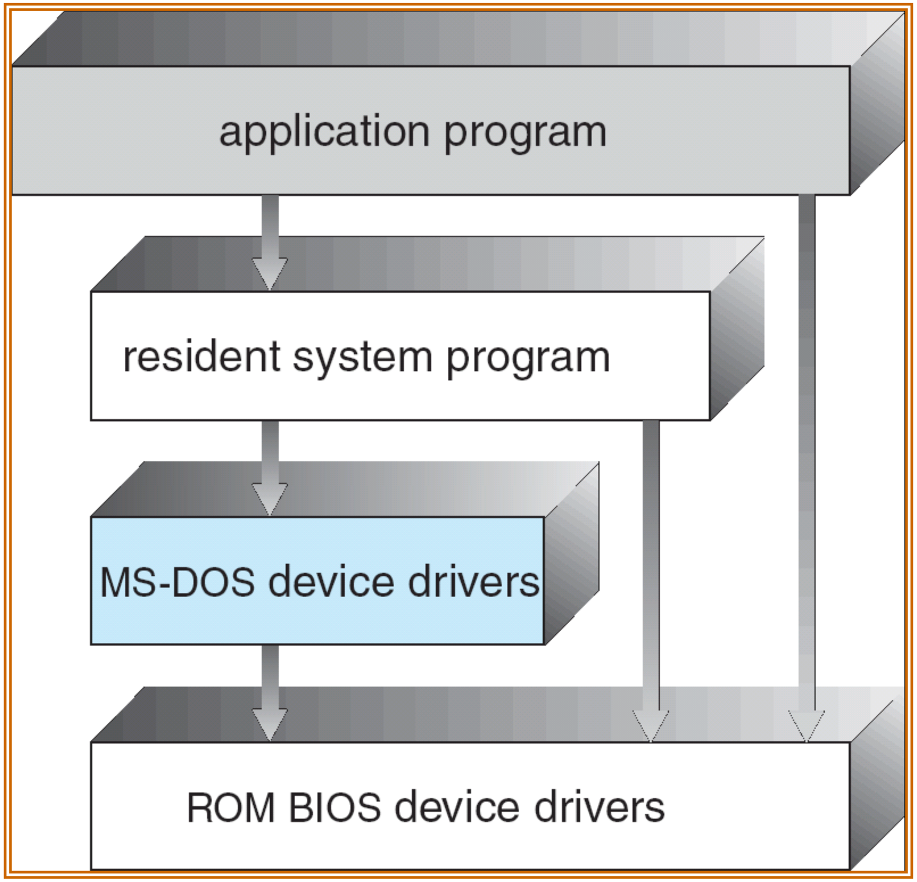
- **Computer System Operation**
  - **I/O Structure**
  - **Storage Structure, Storage Hierarchy**
  - **Hardware Protection**
- **Operating System Services, System calls, System Programs**
- **Structuring OS**
  - **Virtual Machine Structure and Organization**
- **OS Design and Implementation**
  - Process Management, Memory Management, Secondary Storage Management, I/O System Management, File Management, Protection System, Networking, Command-Interpreter.

Operating Systems: How are they organized?

- **Simple**
  - **Only one or two levels of code**
- **Layered**
  - **Lower levels independent of upper levels**
- **Microkernel**
  - **OS built from many user-level processes**
- **Modular**
  - **Core kernel with Dynamically loadable modules**

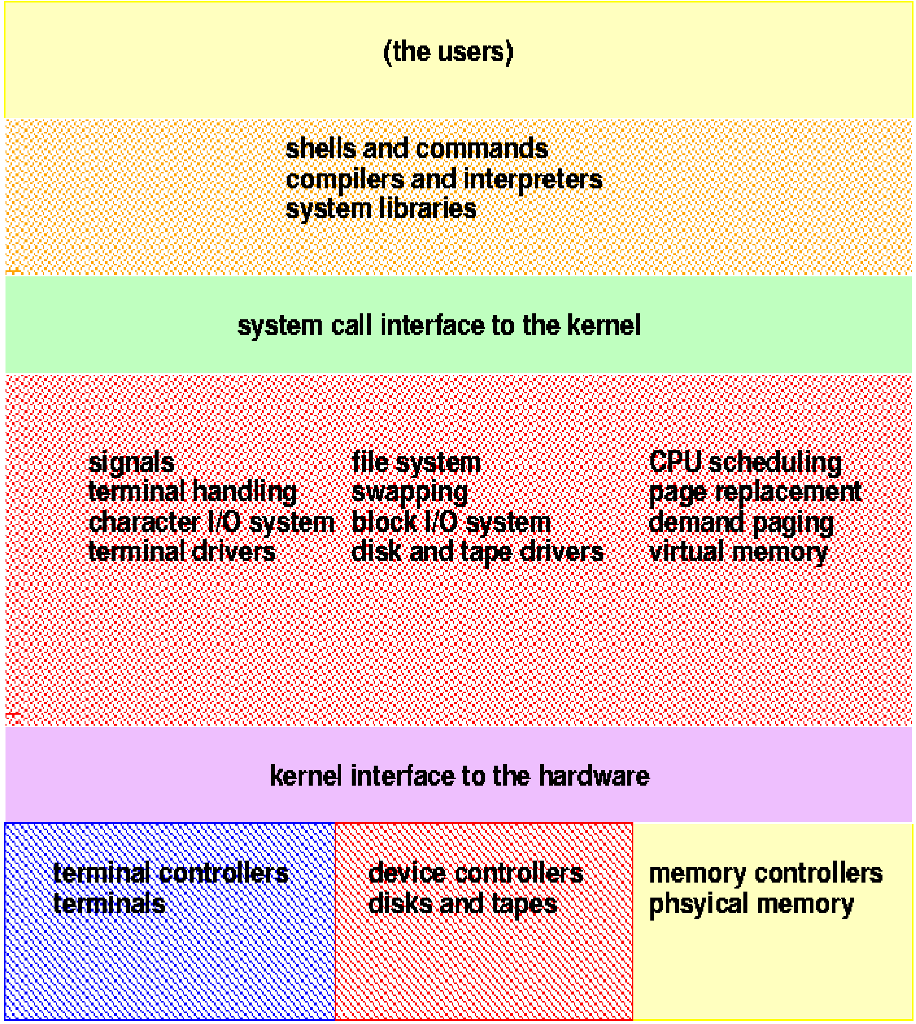
OS Structure - Simple Approach

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



UNIX System Structure

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

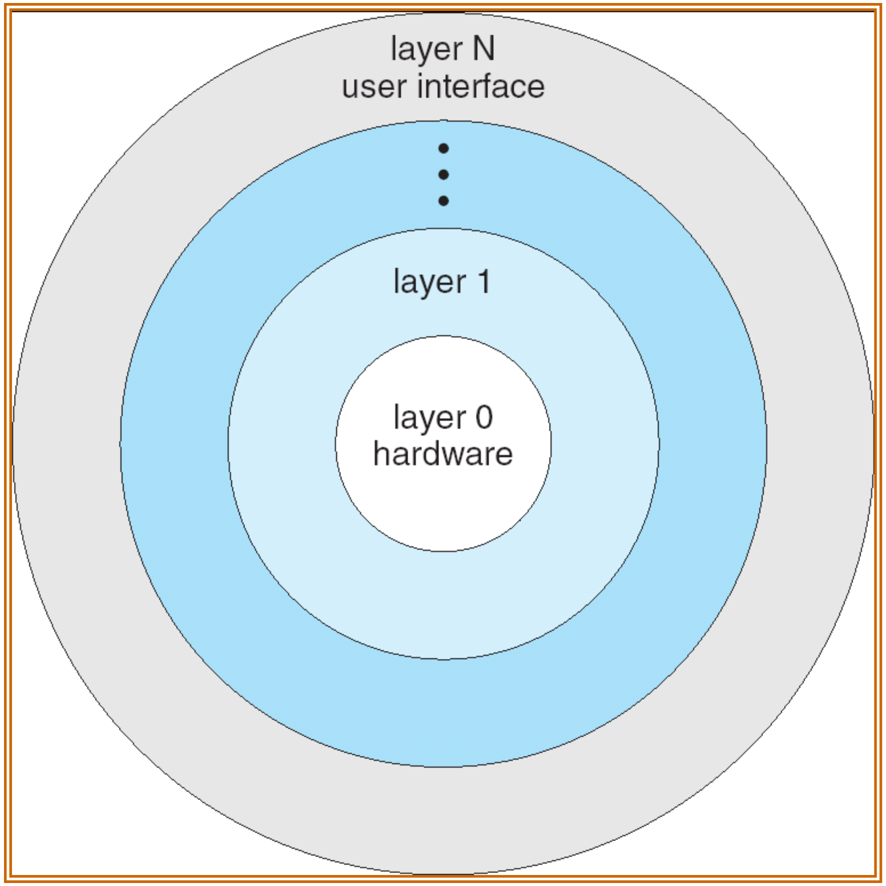


Layered OS Structure

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

- **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 Kernel has successive layers of abstraction.**

Layered Operating System

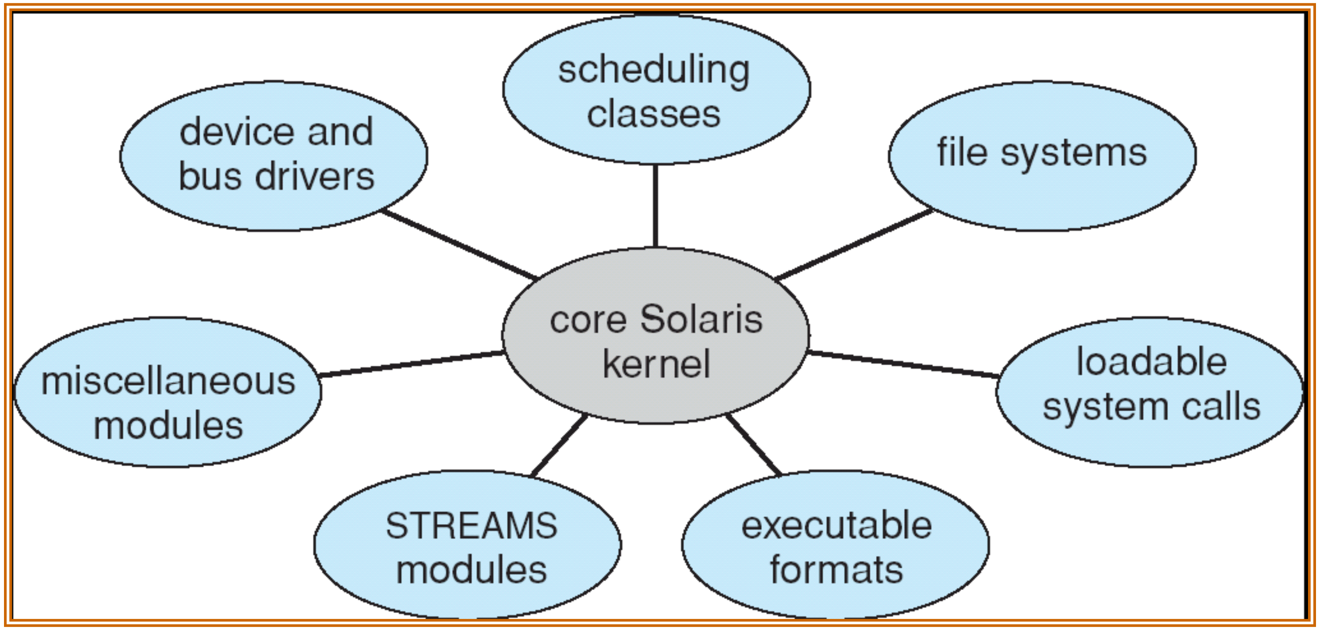


Microkernel Structure

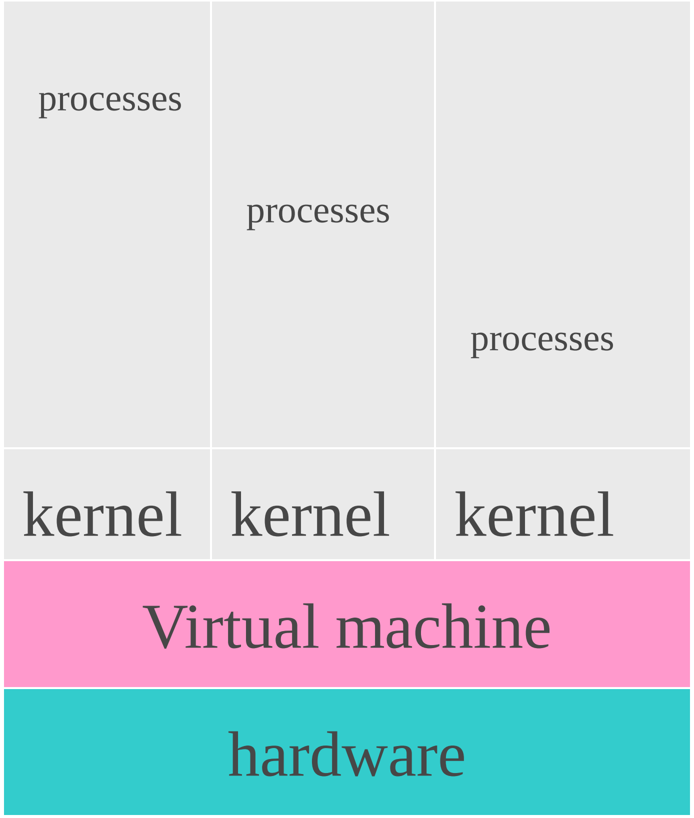
- Moves as much from the kernel into "*user*" space
  - Small core OS running at kernel level
  - OS Services built from many independent user-level processes
- Communication between modules with message passing
- Benefits:
  - Easier to extend a microkernel
  - Easier to port OS to new architectures
  - More reliable (less code is running in kernel mode)
  - Fault Isolation (parts of kernel protected from other parts)
  - More secure
- Detriments:
  - Performance overhead severe for naïve implementation

Modules-based Structure

- Most modern operating systems implement modules
  - Uses object-oriented approach
  - Each core component is separate
  - Each talks to the others over known interfaces
  - Each is loadable as needed within the kernel
- Overall, similar to layers but with more flexible



Virtual Machines



- Logically treats hardware and OS kernel as hardware
- Provides interface identical to underlying bare hardware.
- Creates illusion of multiple processes - each with its own processor and virtual memory

Computer System & OS Structures

- **Computer System Operation**
  - **I/O Structure**
  - **Storage Structure, Storage Hierarchy**
  - **Hardware Protection**
- **Operating System Services, System calls, System Programs**
- **Structuring OS**
  - **Virtual Machine Structure and Organization**
- **OS Design and Implementation**
  - Process Management, Memory Management, Secondary Storage Management, I/O System Management, File Management, Protection System, Networking, Command-Interpreter.

OS Task: Process Management

- **Process - fundamental concept in OS**
  - **Process is 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 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 and disk scheduling.
- I/O system in the OS consists of
  - Buffer caching and management
  - Device driver interface that abstracts device details
  - Drivers for specific hardware devices

OS Task: File System Management

- **File is a collection of related information defined by creator - 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).**
  - **Backup files on archival media (tapes).**

OS Task: Protection and Security

- **Protection mechanisms control access of programs and processes to user and system resources.**
  - **Protect user from himself, user from other users, system from users.**
- **Protection mechanisms must:**
  - **Distinguish between authorized and unauthorized use.**
  - **Specify access controls to be imposed on use.**
  - **Provide mechanisms for enforcement of access control.**
  - **Security mechanisms provide trust in system and privacy**
    - authentication, certification, encryption etc.

OS Task: Networking

- Connecting processors in a distributed system
- Distributed System is a collection of processors that do not share memory or a clock.
- Processors are connected via a communication network.
- Advantages:
  - Allows users and system to exchange information
  - provide computational speedup
  - increased reliability and availability of information

Summary of OS Structures

- **Operating System Concepts**
- **Operating System Services, System Programs and System calls**
- **Operating System Design and Implementation**
- **Structuring Operating Systems**

References

Part of the contents of this lecture has been adapted from the book Abraham Silberschatz, Peter B. Galvin, Greg Gagne: "Operating System Concept ", Publisher : Wiley; 9 edition (December 17, 2012), ISBN-13: 978-1118063330

Slides also contain lecture materials from John Kubiatowicz (Berkeley), John Ousterhout (Stanford), Nalini (UCI), Rainer (UCI), and others

Some slides adapted from http://www-inst.eecs.berkeley.edu/~cs162/ Copyright © 2010 UCB

Thank you for your attention