Introduction to Operating System

Apurba Sarkar

IIEST Shibpur

July 21, 2016

Overview

- What is an operating system?
- Operating systems history
- Computer system and operating system structure

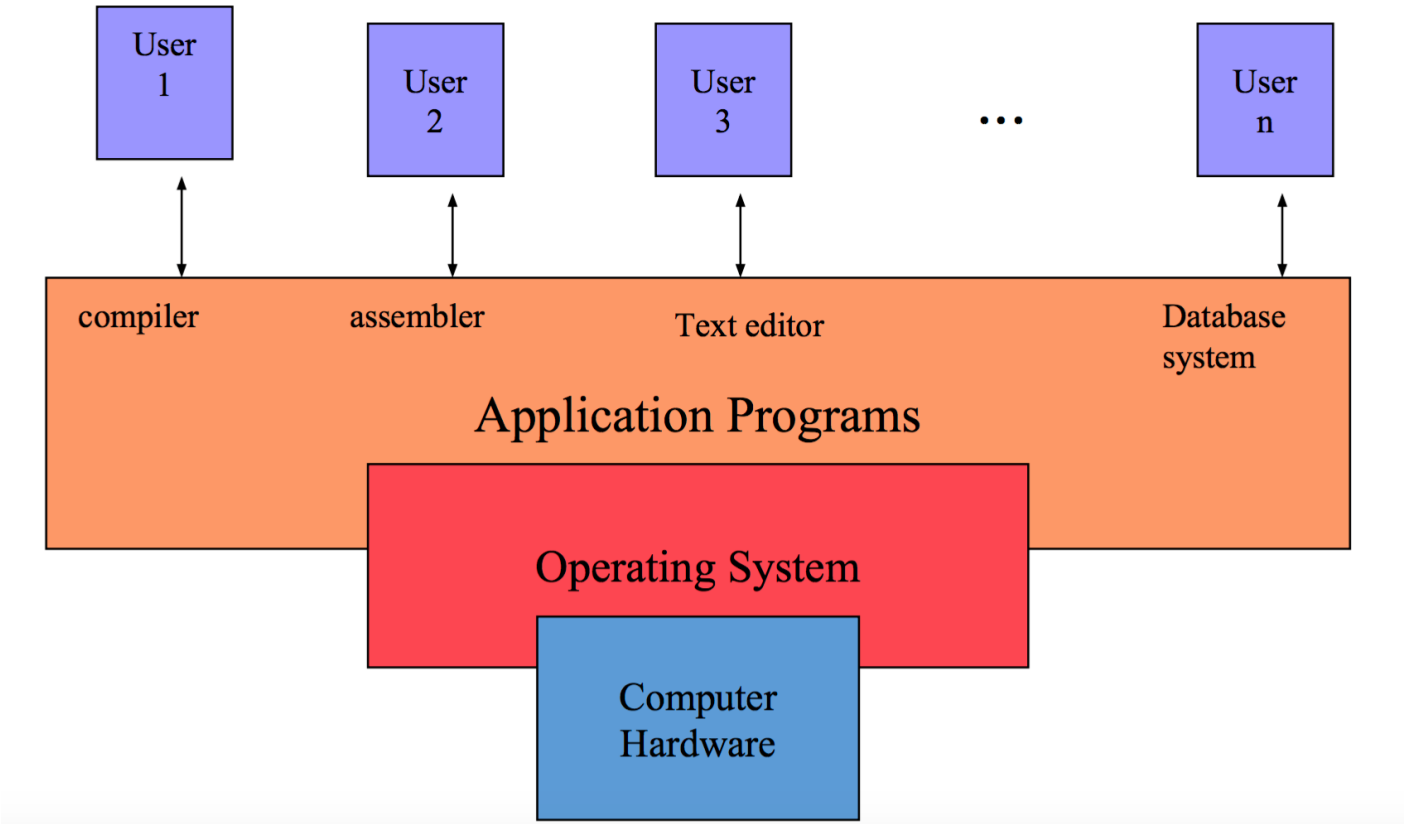
What is an operating system?

- OS is the software that acts an intermediary between the user applications and computer hardware.

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 rolls

- Referee:
  - Resource allocation among users, applications
  - Isolation of different users, applications from each other
  - Communication between users, applications
- Illusionist
  - Files can grow (nearly) arbitrarily large
  - Files persist even when the machine crashes in the middle of a save
- Glue
  - Named directories, printf, . . .

Operating system challenges

- Reliability
  - Does the system do what it was designed to do?

Operating system challenges

- Reliability
  - Does the system do what it was designed to do?
- Availability
  - What portion of the time is the system working?

Operating system challenges

- Reliability
  - Does the system do what it was designed to do?
- Availability
  - What portion of the time is the system working?
- Security
  - Can the system be compromised by an attacker?

Operating system challenges

- Reliability
  - Does the system do what it was designed to do?
- Availability
  - What portion of the time is the system working?
- Security
  - Can the system be compromised by an attacker?
- Privacy
  - Whether data is accessible only to authorized users?

Operating system challenges contd.

- Performance
  - Latency/response time
    - How long does an operation take to complete?

Operating system challenges contd.

- Performance
  - Latency/response time
    - How long does an operation take to complete?
  - Throughput
    - How many operations can be done per unit of time?

Operating system challenges contd.

- Performance
  - Latency/response time
    - How long does an operation take to complete?
  - Throughput
    - How many operations can be done per unit of time?
  - Overhead
    - How much extra work is done by the OS?

Operating system challenges contd.

- Performance
  - Latency/response time
    - How long does an operation take to complete?
  - Throughput
    - How many operations can be done per unit of time?
  - Overhead
    - How much extra work is done by the OS?
  - Fairness
    - How equal is the performance received by different users?

Operating system challenges contd.

- Performance
  - Latency/response time
    - How long does an operation take to complete?
  - Throughput
    - How many operations can be done per unit of time?
  - Overhead
    - How much extra work is done by the OS?
  - Fairness
    - How equal is the performance received by different users?
  - Predictability
    - How consistent is the performance over time?

Why should I study Operating Systems?

- Need to understand interaction between the hardware and software
- Need to understand basic principles in the design of computer systems
  - efficient resource management, security, flexibility
- Above all of these, it enables you to do things that are difficult/impossible otherwise.

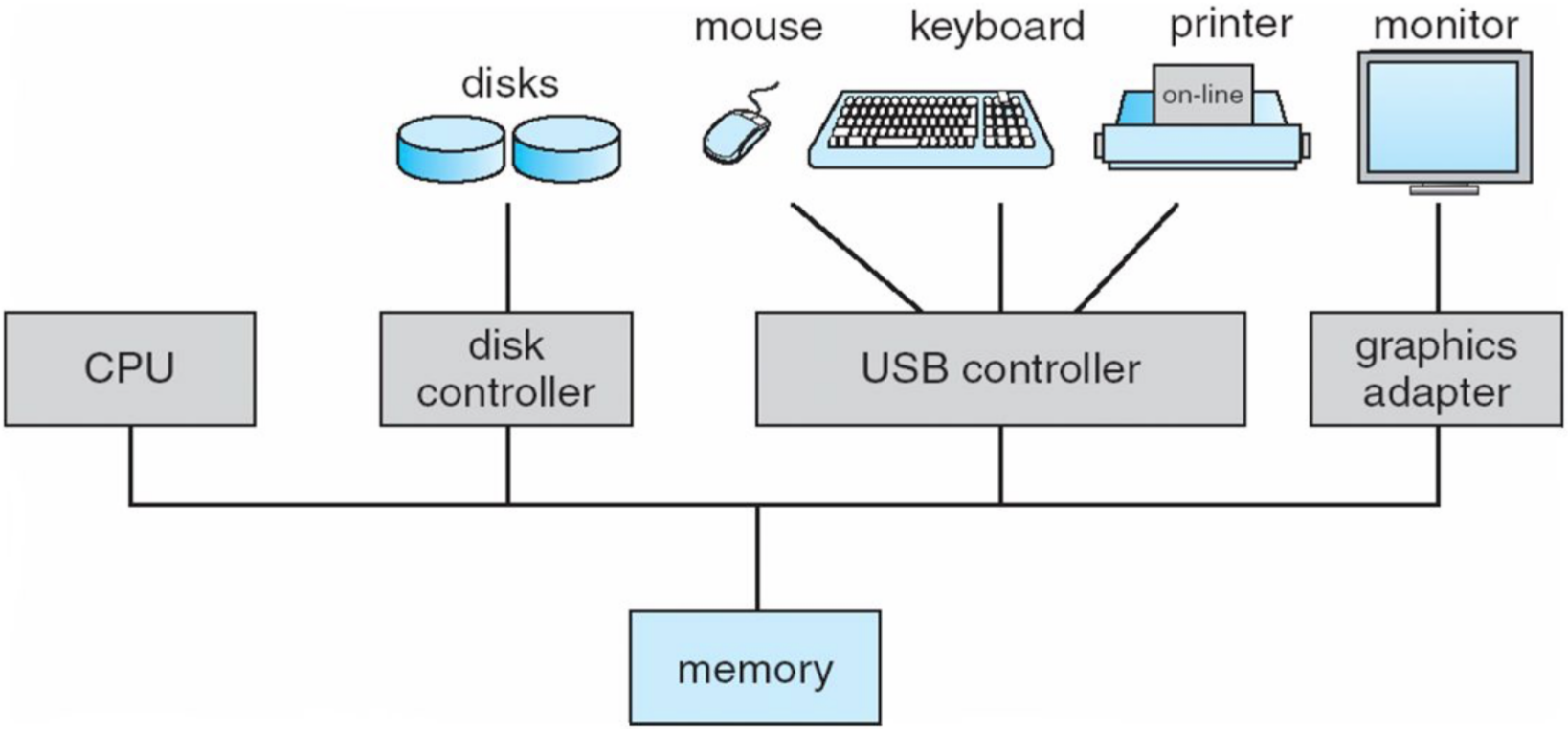
Operating systems history

- Early Operating Systems
- Simple Batch Systems
- Multiprogrammed Batch Systems
- Time-sharing Systems
- Personal and mobile Computer Systems

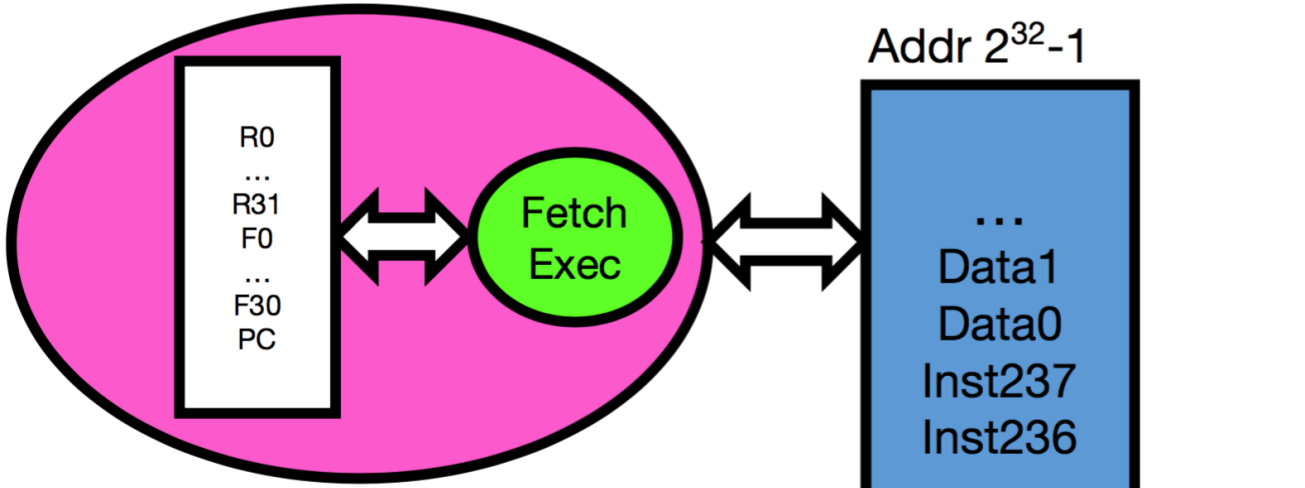
Computer System & OS Structures

- Computer System Organization
- Process abstraction and hardware protection
- System call and OS services
- Storage architecture
- OS organization
- OS Tasks
- Virtual Machines

Computer System Organization



CPU Execution

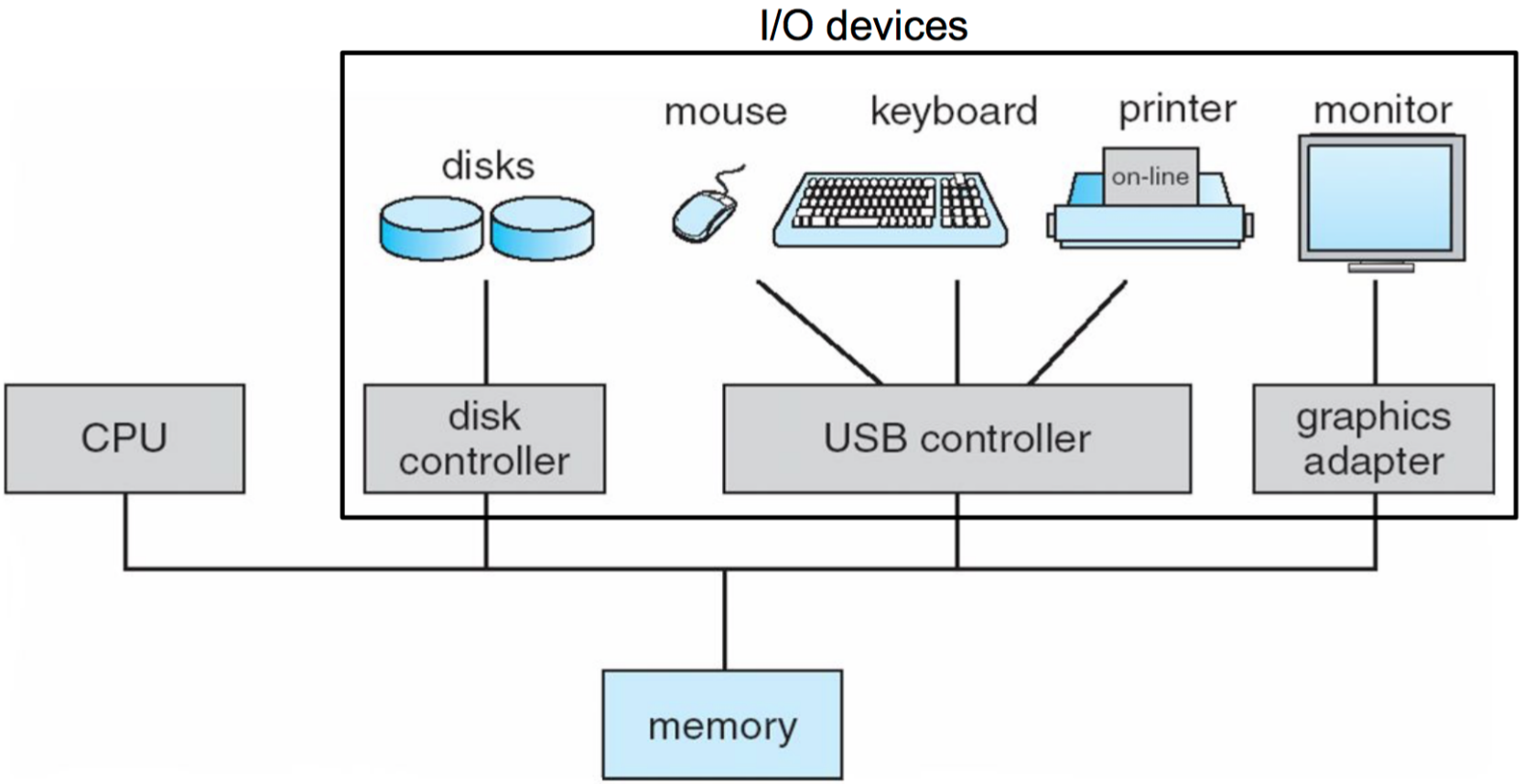


Execution sequence:

- Fetch Instruction at PC
- Decode
- Execute (possibly using registers)
- Write results to registers/mem
- PC = Next Instruction(PC)
- Repeat

From Berkeley OS course

Computer System Organization



I/O devices

- I/O devices and the CPU execute concurrently.
- Each device controller is in charge of a particular device type
- Each device controller has a local buffer. I/O is from the device to local buffer of controller
- CPU moves data from/to main memory to/from the local buffers
- Device controller interrupts CPU on completion of I/O

Interrupts

- Interrupt transfers control to the interrupt service routine
  - Interrupt Service Routine: Segments of code that determine action to be taken for interrupt.
- Determining the type of interrupt
  - Polling: same interrupt handler called for all interrupts, which then polls all devices to figure out the reason for the interrupt
  - Interrupt Vector Table: different interrupt handlers will be executed for different interrupts.

Interrupt handling

- OS preserves the state of the CPU
  - stores registers and the program counter (address of interrupted instruction).
- Incoming interrupts are disabled while another interrupt is being processed to prevent a lost interrupt.

Different types of I/O processing for programs

- Synchronous I/O
  - After I/O is requested, wait until I/O is done. Program will be idle.
- Asynchronous I/O
  - After I/O is requested, control returns to user program without waiting for I/O completion.

Process Abstraction

- **Process**: an instance of a program, running with limited rights
  - **Thread**: a sequence of instructions within a process
    - Potentially many threads per process.
  - Address space: set of rights of a process
    - Memory that the process can access
    - Other permissions the process has (e.g., which system calls it can make, what files it can access)

How to limit process rights

Hardware Protection

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

Should a process be able to execute any instructions

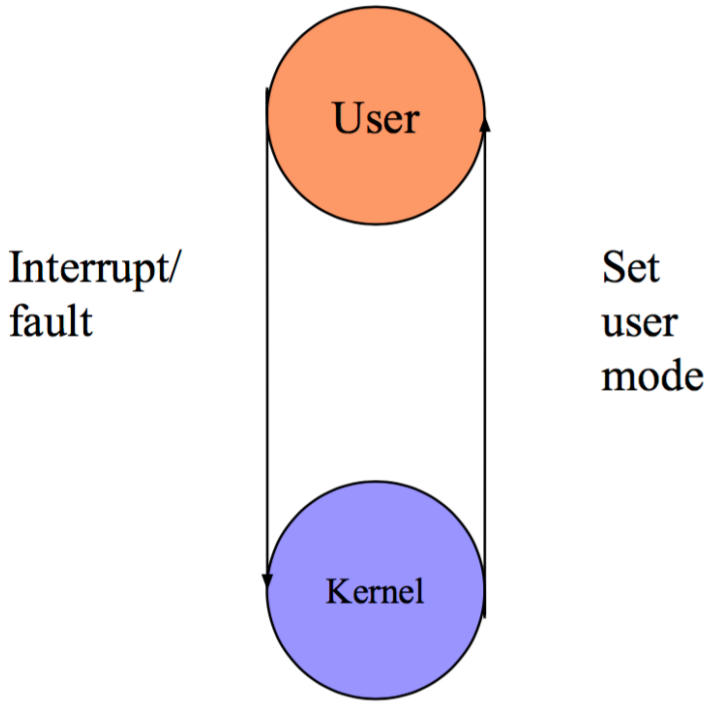
- No
  - Can alter system configuration
  - Can access unauthorized memory
  - Can access unauthorized I/O
  - etc.
- How to prevent?

Dual-mode operation

- Provide hardware support to differentiate between at least two modes of operation:
  - User mode – execution done on behalf of a user.
  - Kernel mode (monitor/supervisor/system mode) – execution done on behalf of operating system.
- “Privileged” instructions are only executable in the kernel mode
- Executing privileged instructions in the user mode “traps” into the kernel mode
  - Trap is a software generated interrupt caused either by an error or a user request.

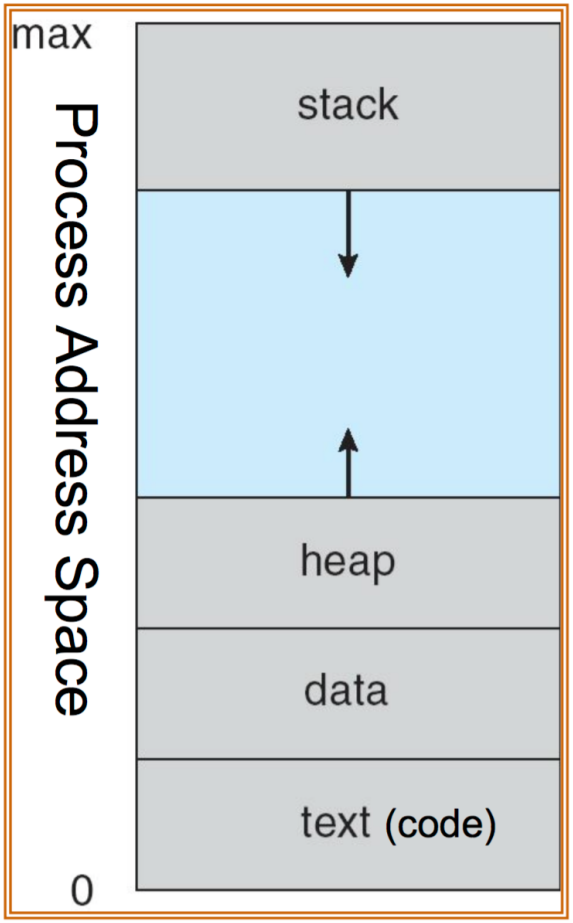
Dual-mode operation contd.

- Mode bit added to computer hardware to indicate the current mode: kernel(0) or user(1).
- When an interrupt or trap occurs, hardware switches to kernel mode.



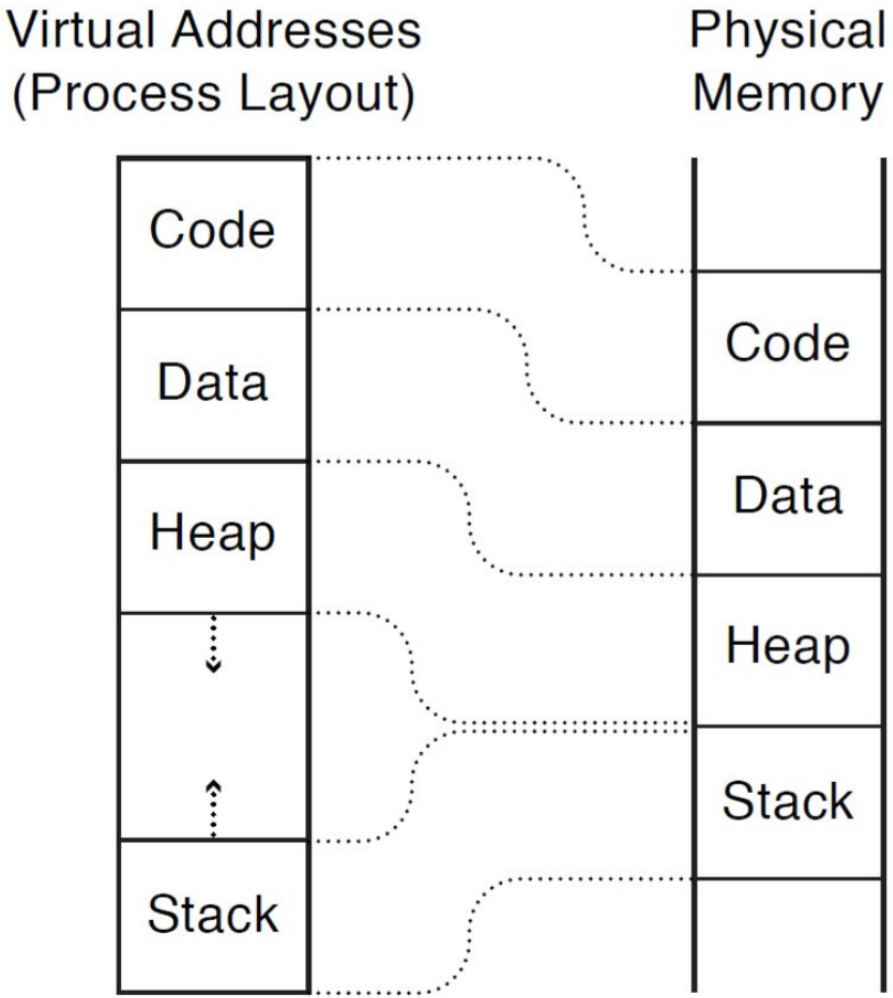
Process Address-Space

- Address space $\Rightarrow$ the set of accessible addresses $+$ state associated with them:
- For a 32-bit processor there are $232 = 4$ billion addresses

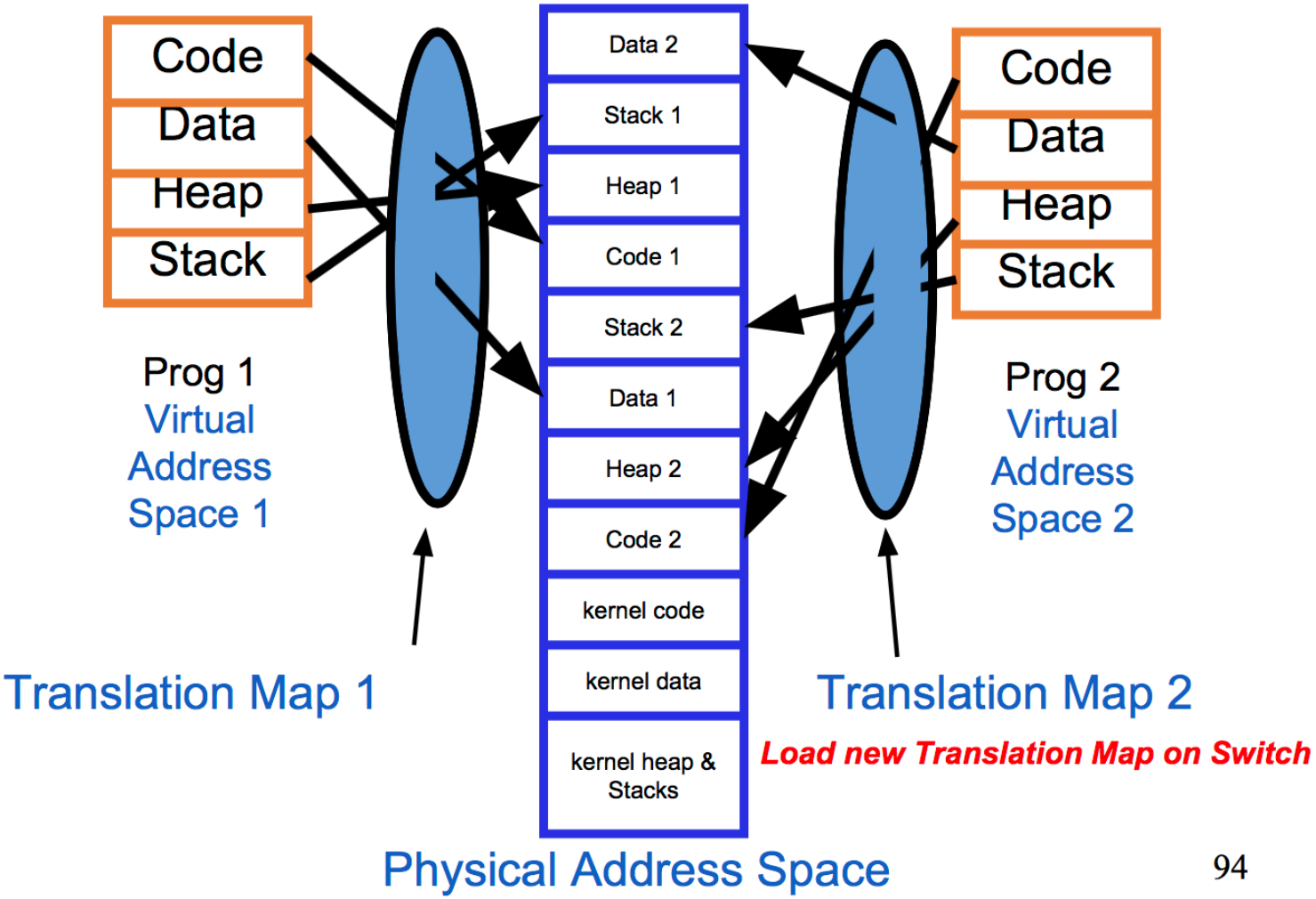


Virtual Address

- Provides the illusion of Separate Address Spaces

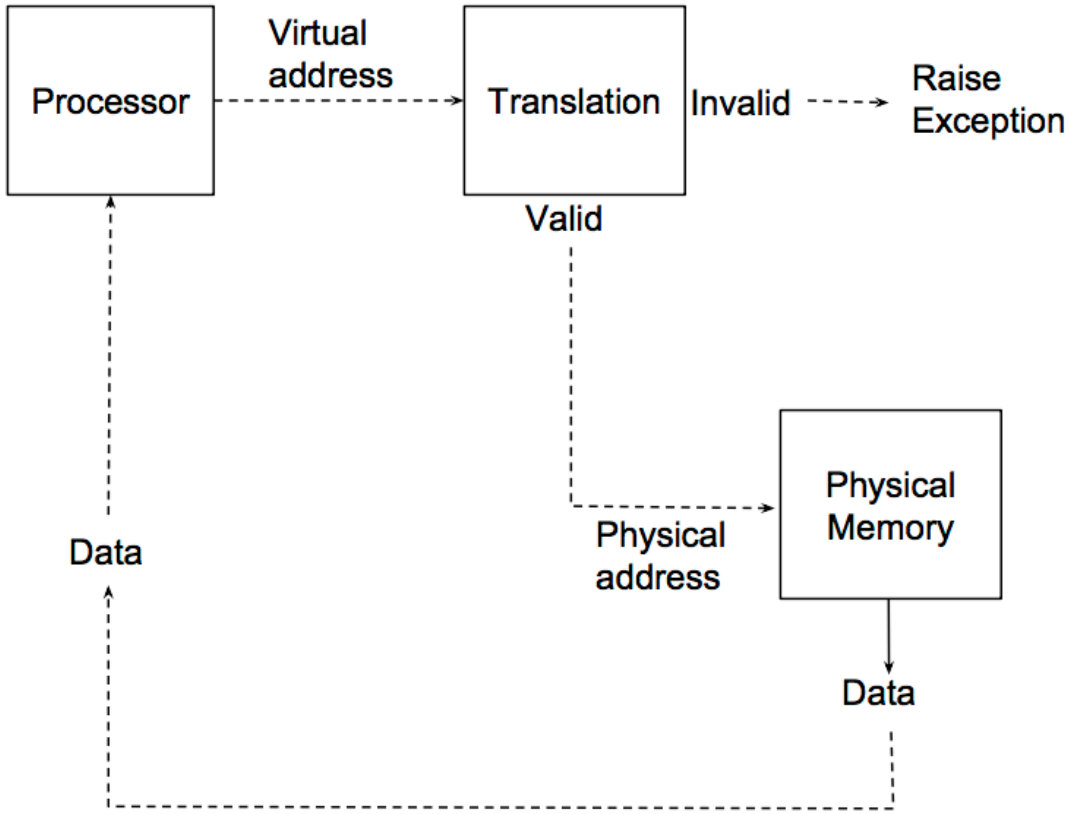


Virtual Address



94

Address translation and memory protection

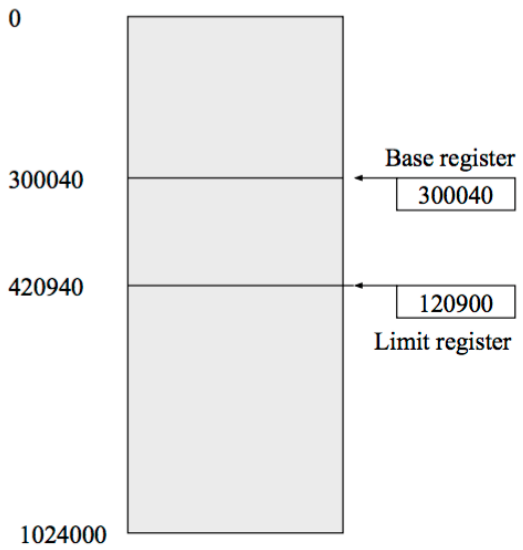


Memory Protection

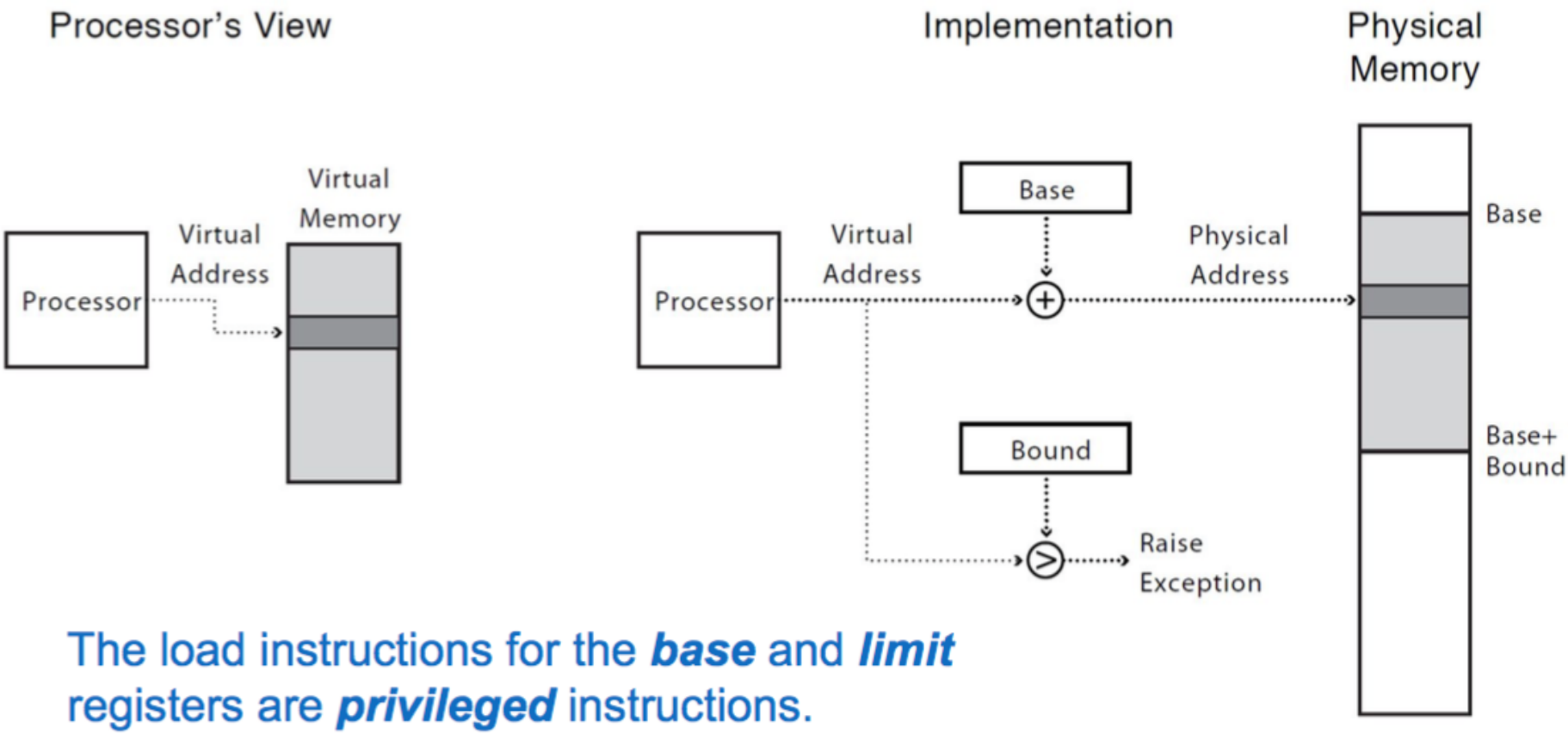
- Must provide memory protection at least for the interrupt vector and the interrupt service routines.
- When a process is running, only memory in that process address space must be accessible.
- When executing in kernel mode, the kernel has unrestricted access to all memory.

Memory Protection: base and bounds

- 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.



Virtually Addressed Base and Bounds



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

CPU Protection

- How to prevent a process from executing indefinitely?

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.

I/O Protection

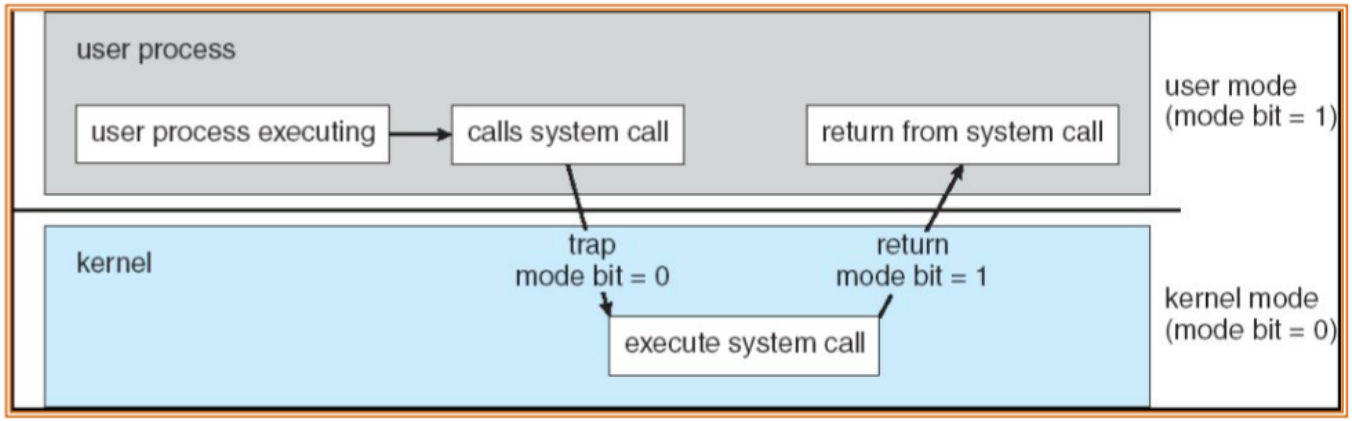
- All I/O instructions are privileged instructions.

A Question

- 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.

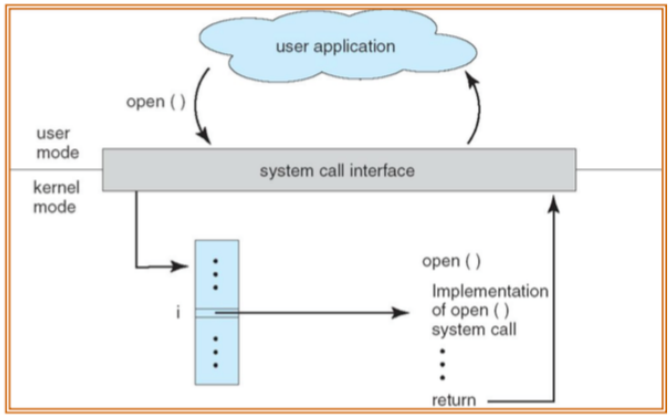
System Calls

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



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.
- Linux has about 300 system calls
- `read()`, `write()`, `open()`, `close()`, `fork()`, `exec()`, `ioctl()`,.....

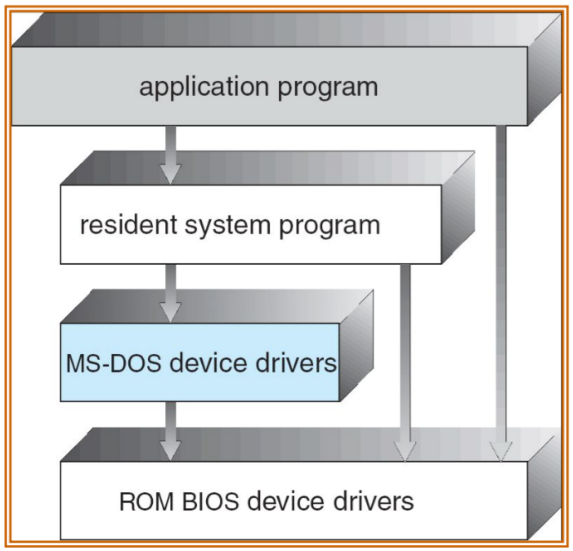


How are the Operating System organized?

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

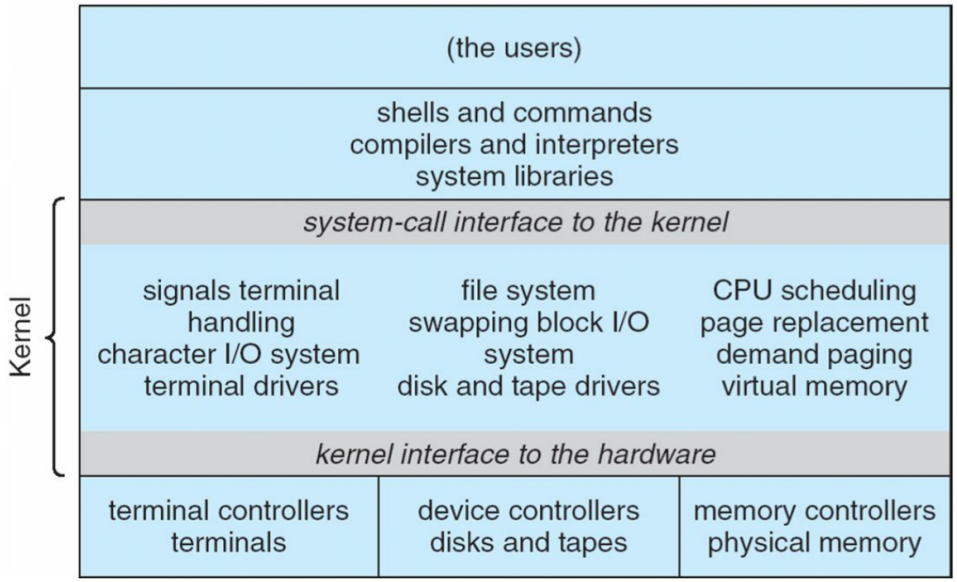
Os structure: Simple Approach

- 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

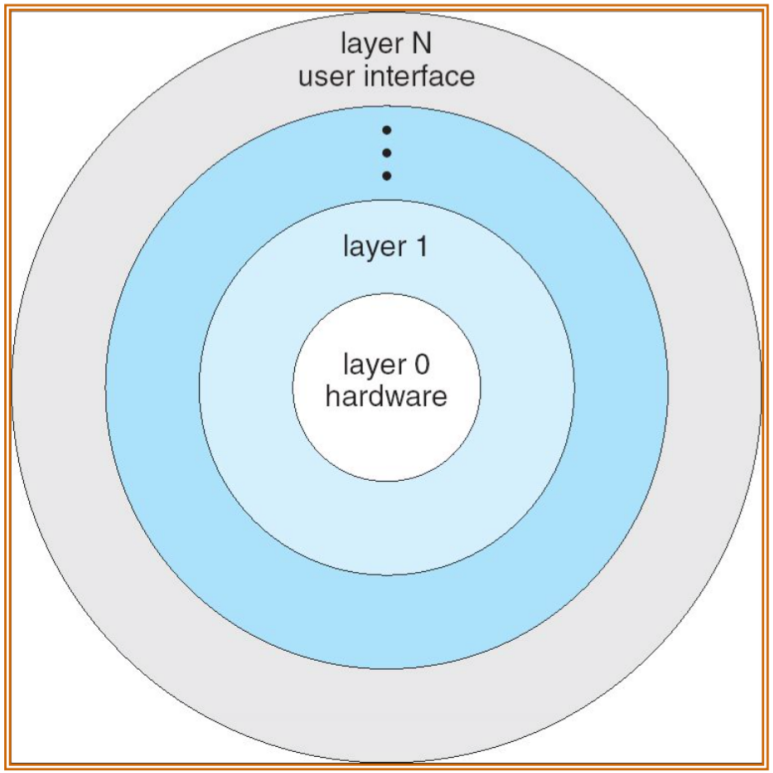
- Limited structuring, has 2 separable parts
  - Systems programs
  - Kernel
    - everything below system call interface and above physical hardware.
    - Filesystem, CPU scheduling, memory management.



Layered OS Structure

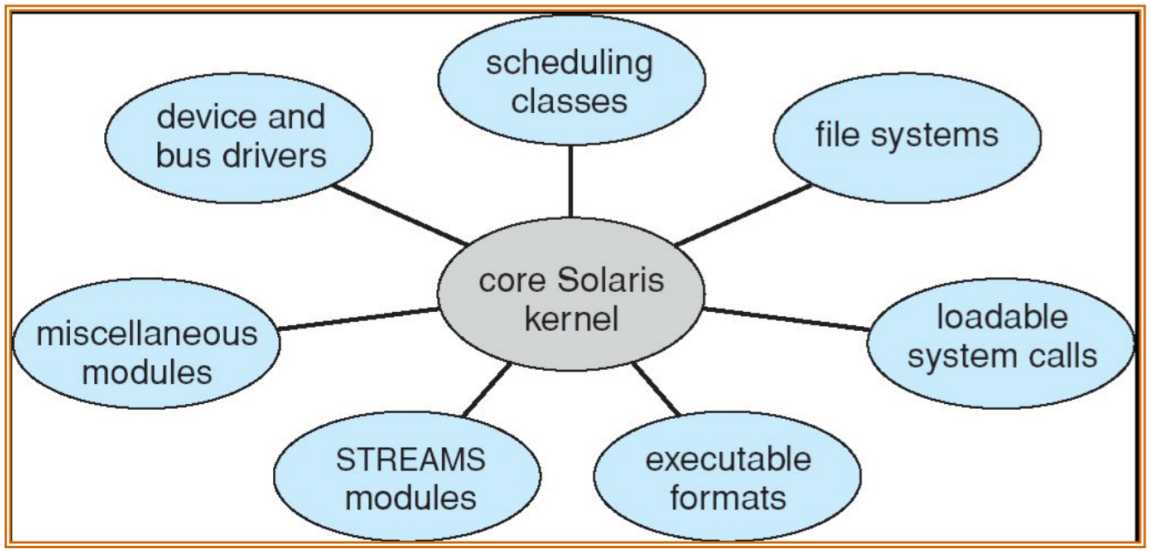
- OS divided into number of layers - bottom layer is hardware, highest layer is the user interface.
- Each layer uses functions and services of only lower- level layers.
- THE Operating System and Linux Kernel has successive layers of abstraction

Layered OS Structure



Modules-based Structure

- Most modern operating systems implement modules
  - 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 flexibility
  - Linux makes extensive use of modules



Monolithic vs. Microkernel OS

- Monolithic OSes have large kernels with a lot of components
  - Linux, Windows, Mac
- Microkernels 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

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 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.