

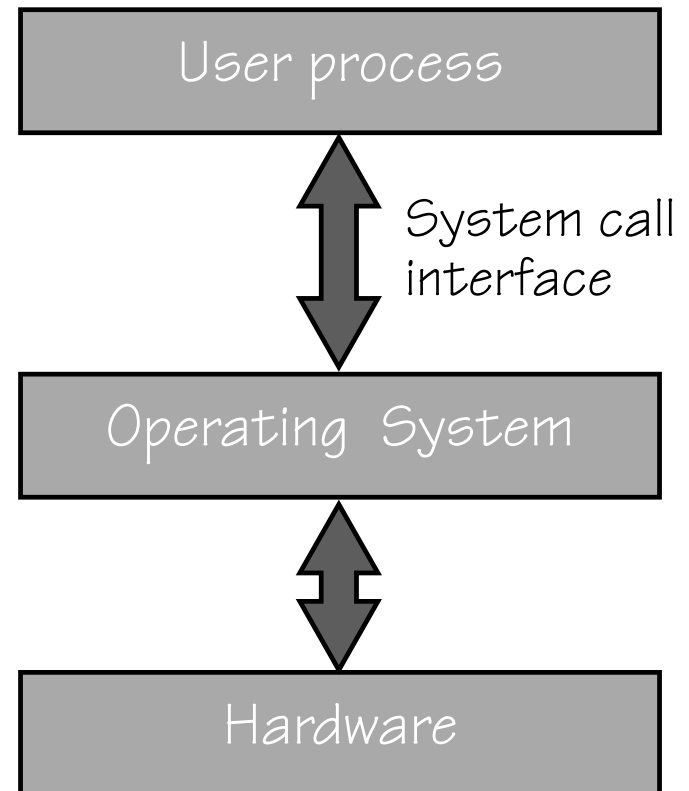
# Operating System Overview

CS450 : Saelee



# Roles of the OS

- Managing hardware
  - CPU, memory, I/O devices
- Managing user processes
  - As a “cop” process
  - As a resource allocator
- Hardware/Software interface
  - Service provider



# OS Services (External)

- Process management
- Error/Exception detection and notification
- Tracing (e.g., for debugging)
- I/O device access (including file I/O)
- Interprocess communication
- Authentication

# Design Objectives

- Abstraction
- Efficiency
- Hardware independence
- Maintainable
- Evolvable
- Secure

# Primary Abstraction: The Process

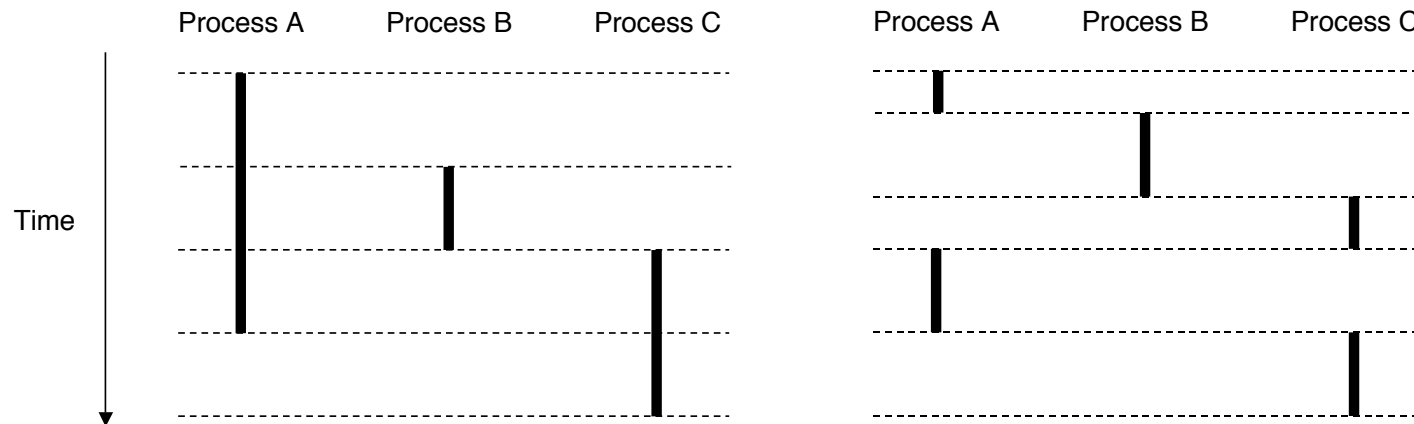
# The Process

- A “program in execution”
  - Program code + data
  - Register and stack context(s)
  - System resources in use
  - Metadata

# Supporting Abstractions

- CPU Virtualization: Logical control flow
- Memory Virtualization: Virtual memory & addressing
- Software interrupts
- Uniform I/O architecture
- Mass/Persistent storage: “file” system

# Logical Control Flow and Concurrency



- Context saving and switching
- Associated problems: race conditions & deadlock!

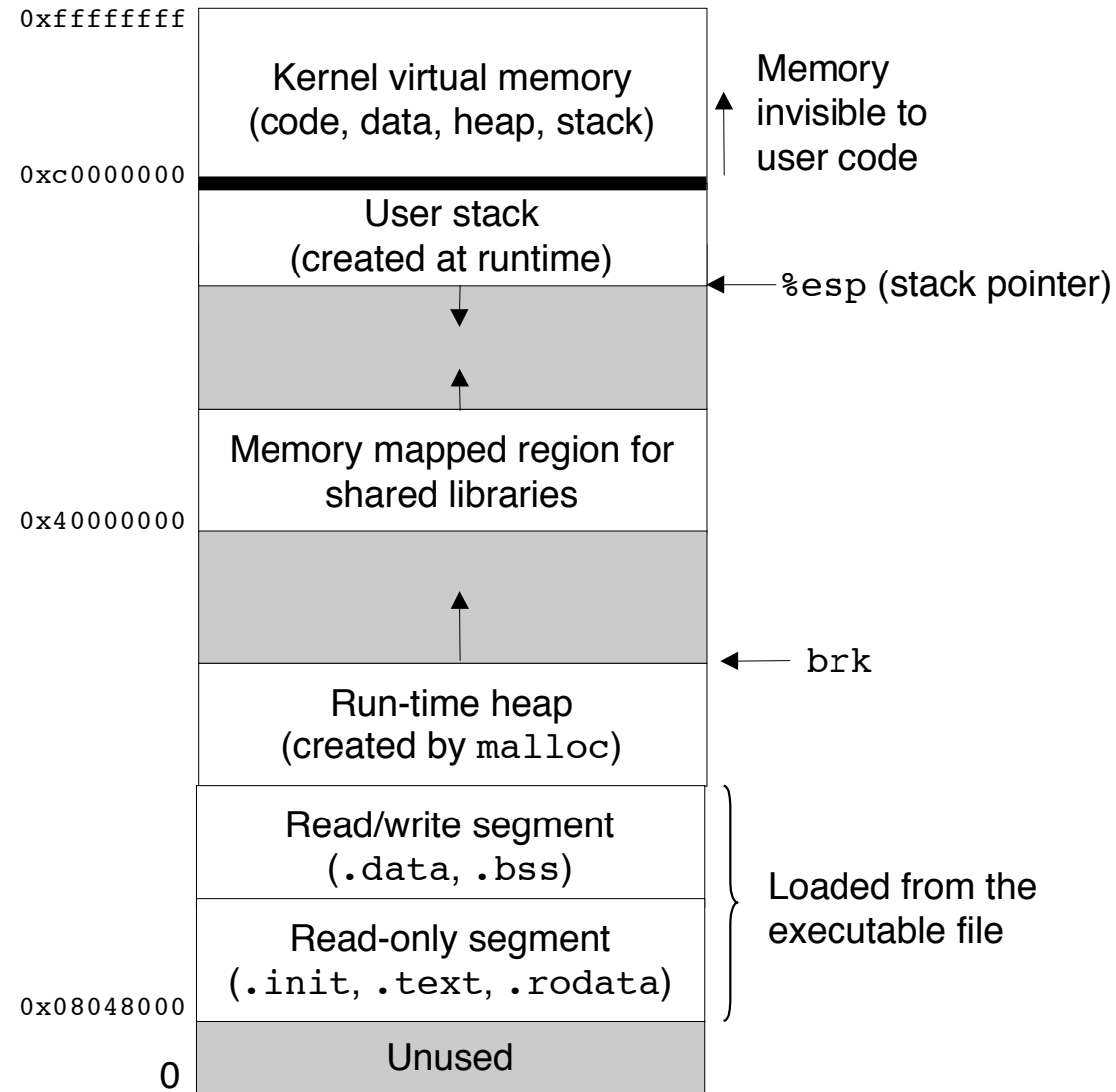
# Software Interrupts

- Notification from kernel that extra-process exception has occurred
  - Purely software driven
  - Error driven; e.g., divide-by-zero
  - Interrupt (hardware) driven; e.g., disk I/O completion
- Implementation: interrupt vector and handler routines
  - Hardware supported

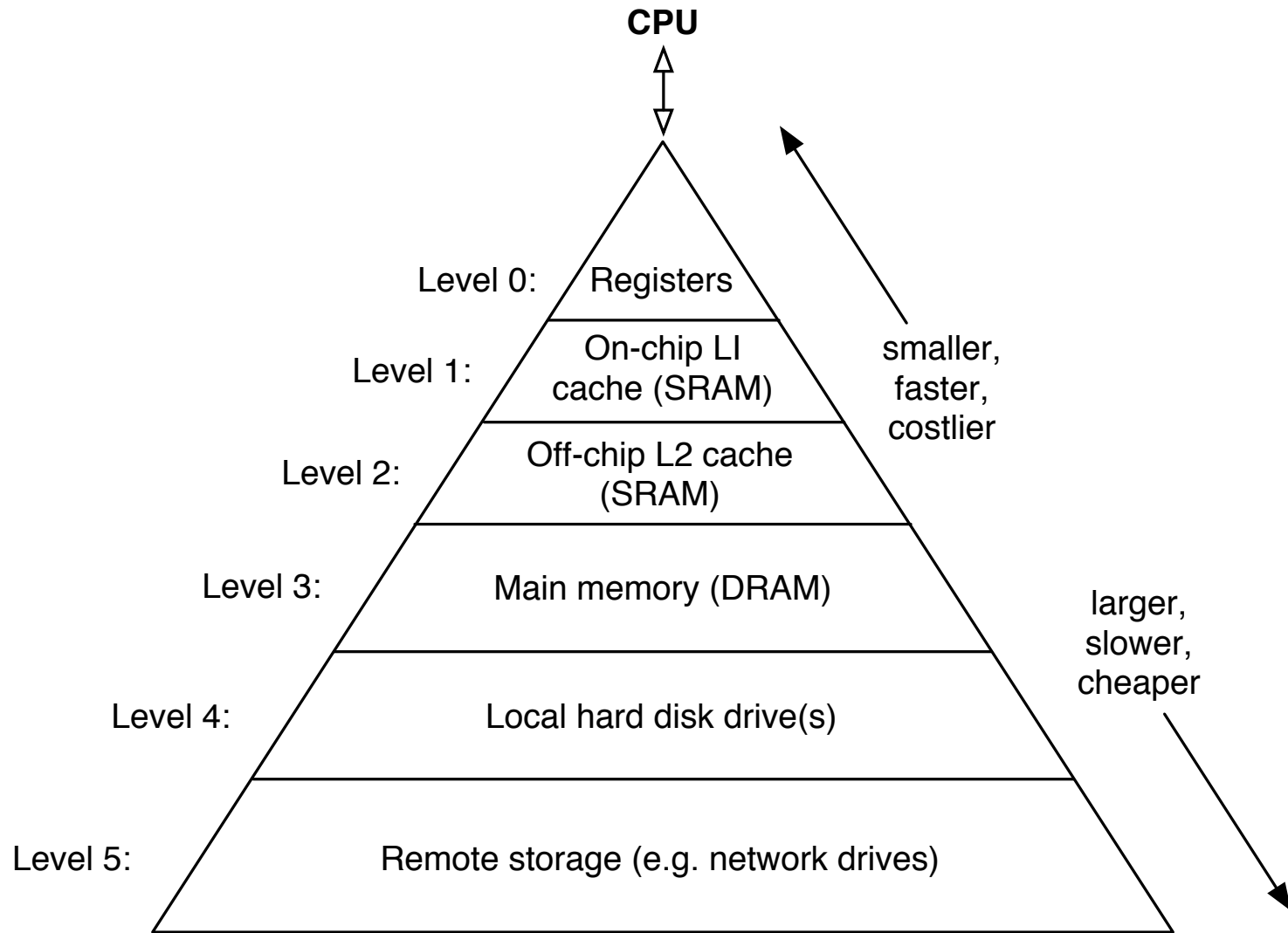
# Virtual Memory

- Uniform view of memory across all processes
- Provide protection and data sharing (e.g., libraries)
- Utilize different memory (hardware) structures
  - Maximize global address space
  - Optimize access times and throughput

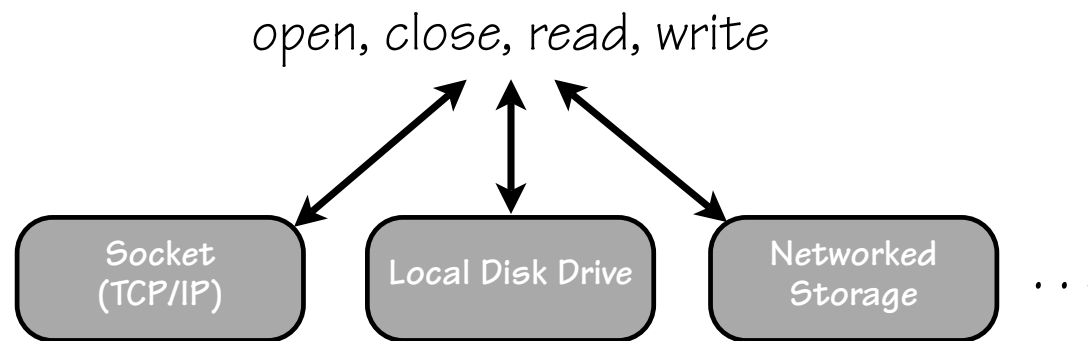
# Process memory image (UNIX)



# The "Memory Hierarchy"



# Uniform I/O Architecture



- I/O requests routed through the OS
- I/O Models: synchronous and asynchronous
  - Polled or interrupt driven
    - Direct memory access improves performance

# File System

- Mass storage
  - Magnetic, optical, solid-state, networked storage
- Slow access times = **system bottleneck**
- Internal and external organization of files
  - Structured/unstructured files; directory organization
- Need to track and allocate free space and schedule file I/O requests
  - Some work delegated to drivers

# Other OS Responsibilities

- Scheduling and resource management
- Security and protection

# Scheduling

- Allocate primary resources (CPU, Memory, I/O devices) to processes
- Accounting
  - Collect statistics
  - Monitor process behavior
  - Assign/ Adjust priorities

# Scheduling Goals

- Maximize throughput
  - Keep total turnaround time small — more processes can complete in a given time frame
  - CPU bursts vs. total process run-time
- Minimize response time
  - More important for interactive processes
- Fairness (?)

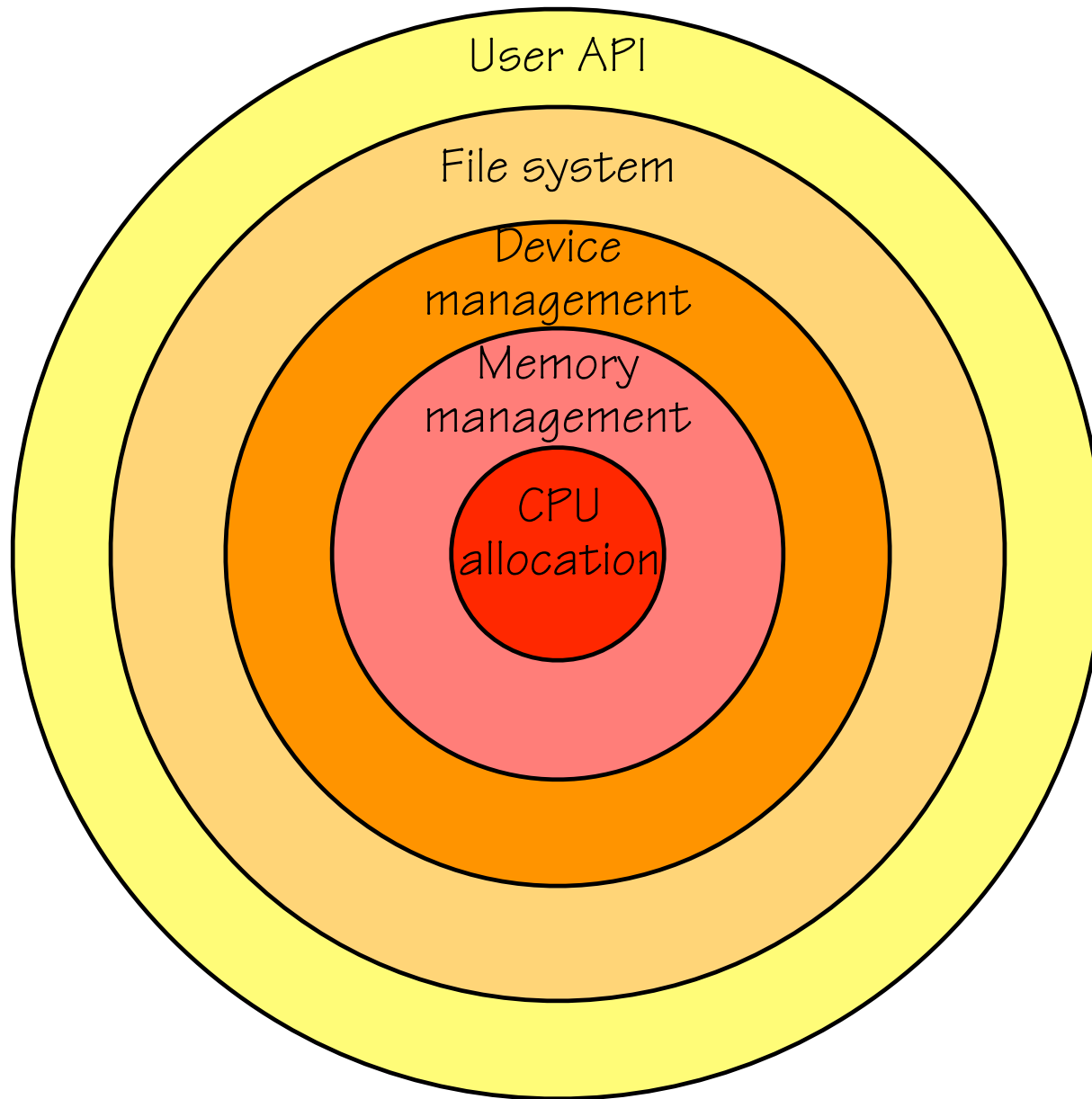
# Security

- Access control
  - Protected memory
  - Restricted file access
- Data encryption (in-core, in-transit, on-disk)

# Layered OS Model

- Divide OS in a few modules and arrange them in layers
- Each outer layer depends on the inner layer
  - In practice, only a guide
  - Multiple cross-dependencies
    - e.g., virtual memory depends on disk access, disk access (to swap data) requires virtual memory data structures

# Operating Systems are like Onions



# The “Kernel”

- “Small” portion of the operating system running in privileged/kernel mode
  - a.k.a. “supervisor”, “nucleus”
- Always core-resident
- Implements core OS functionality
  - Required by all “other” OS services / abstractions
  - First tier of security
- How big should it be? What should it include?

# Monolithic Design

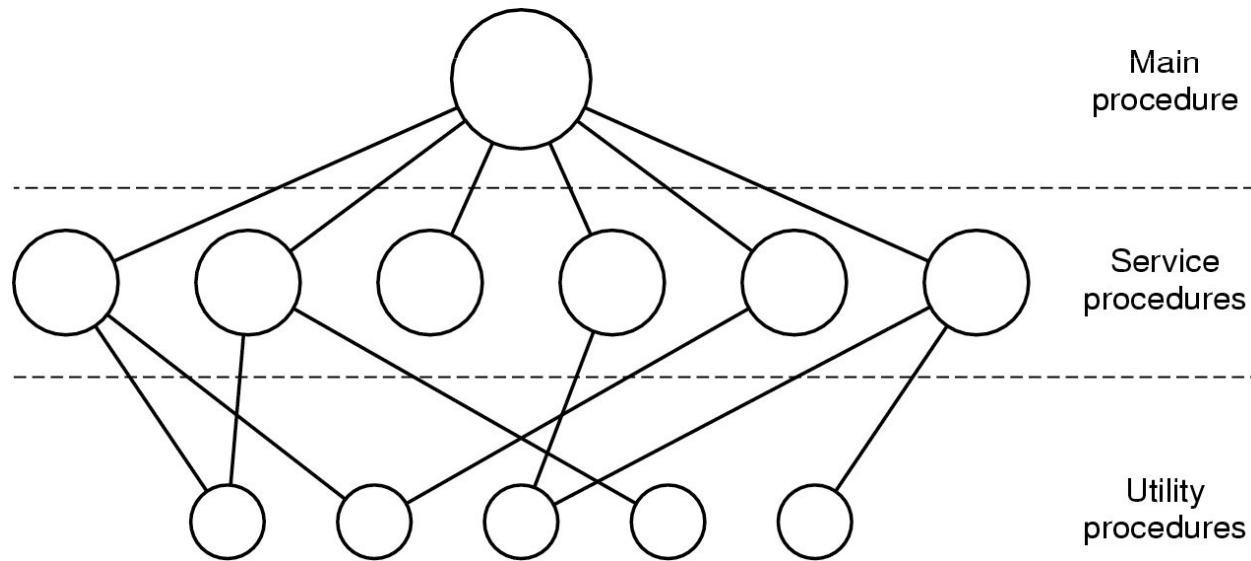
- No real separation of modules or concerns
- Start with main (e.g., bootloader), and arrive at a big jumble of functions
  - Isolation of concerns is at programmer's discretion

# The Monolithic Kernel



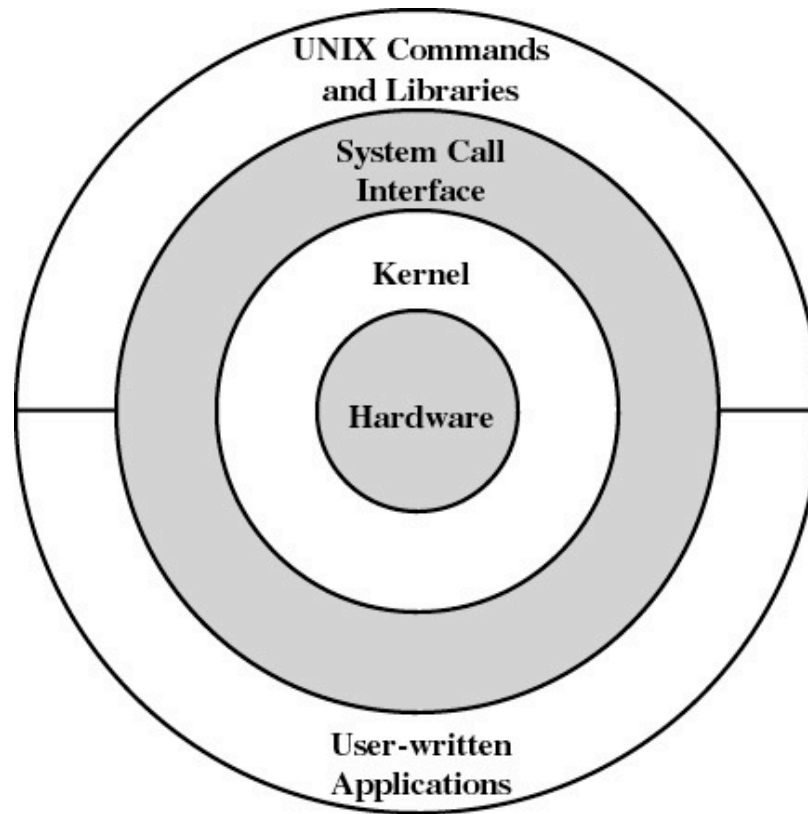
*Image courtesy of Wikipedia*

Some basic organization can be present...

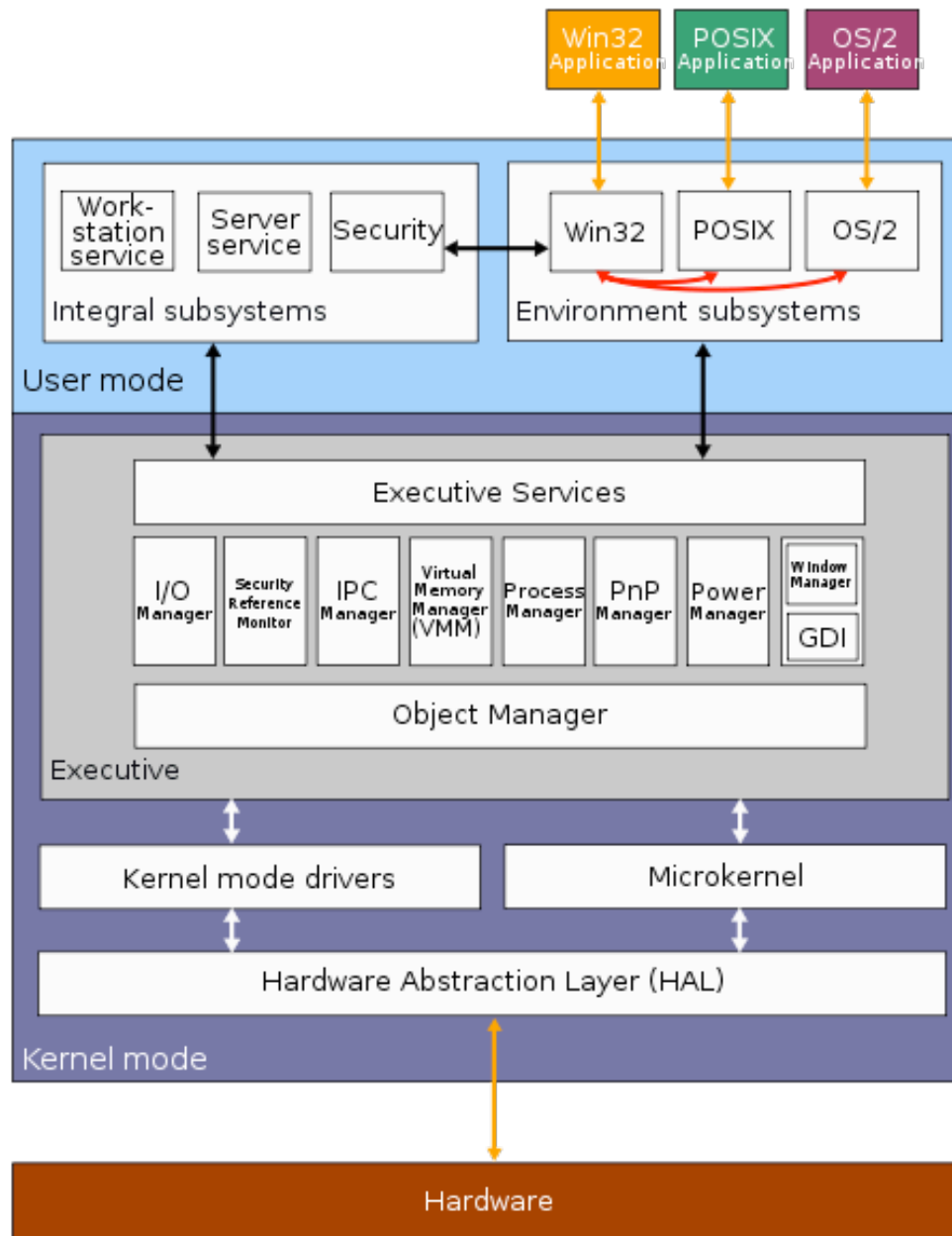


Most popular OS organization;  
e.g., Linux, Windows

# Traditional UNIX OS Organization



# Windows NT OS Organization



# Microkernel Design

- Kernel only implements core functions
  - Addressing
  - IPC
  - Basic scheduling (CPU)
  - Security
- All other services provided by user level modules
  - Most “system calls” become IPC requests to servers

# Comparison of Monolithic and Microkernel OS Organization

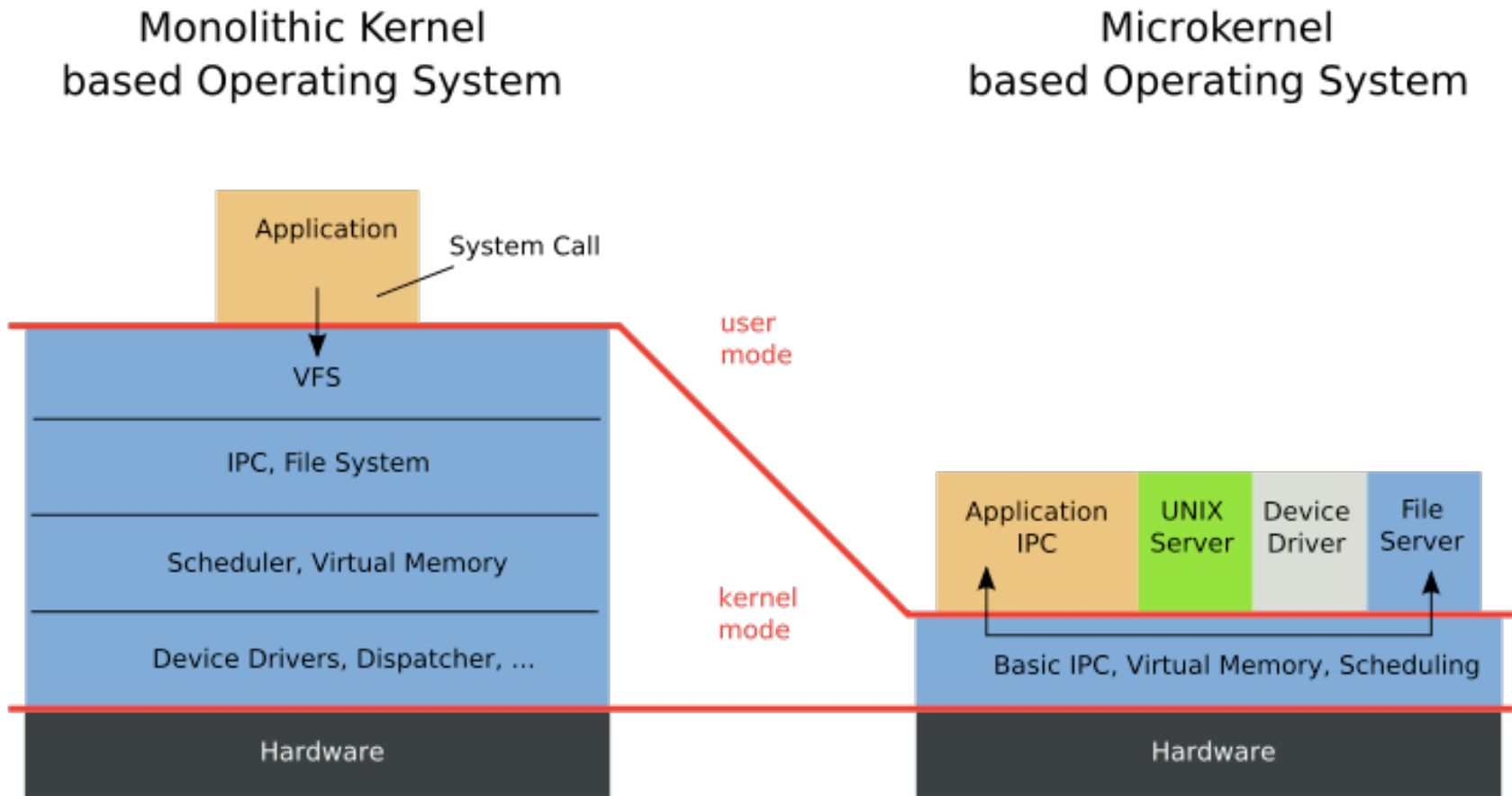


Figure courtesy of Wikipedia