Operating System Structures

Course: IN2311 – Operating Systems

Lecturer: Roshani Wijesuriya

Introduction

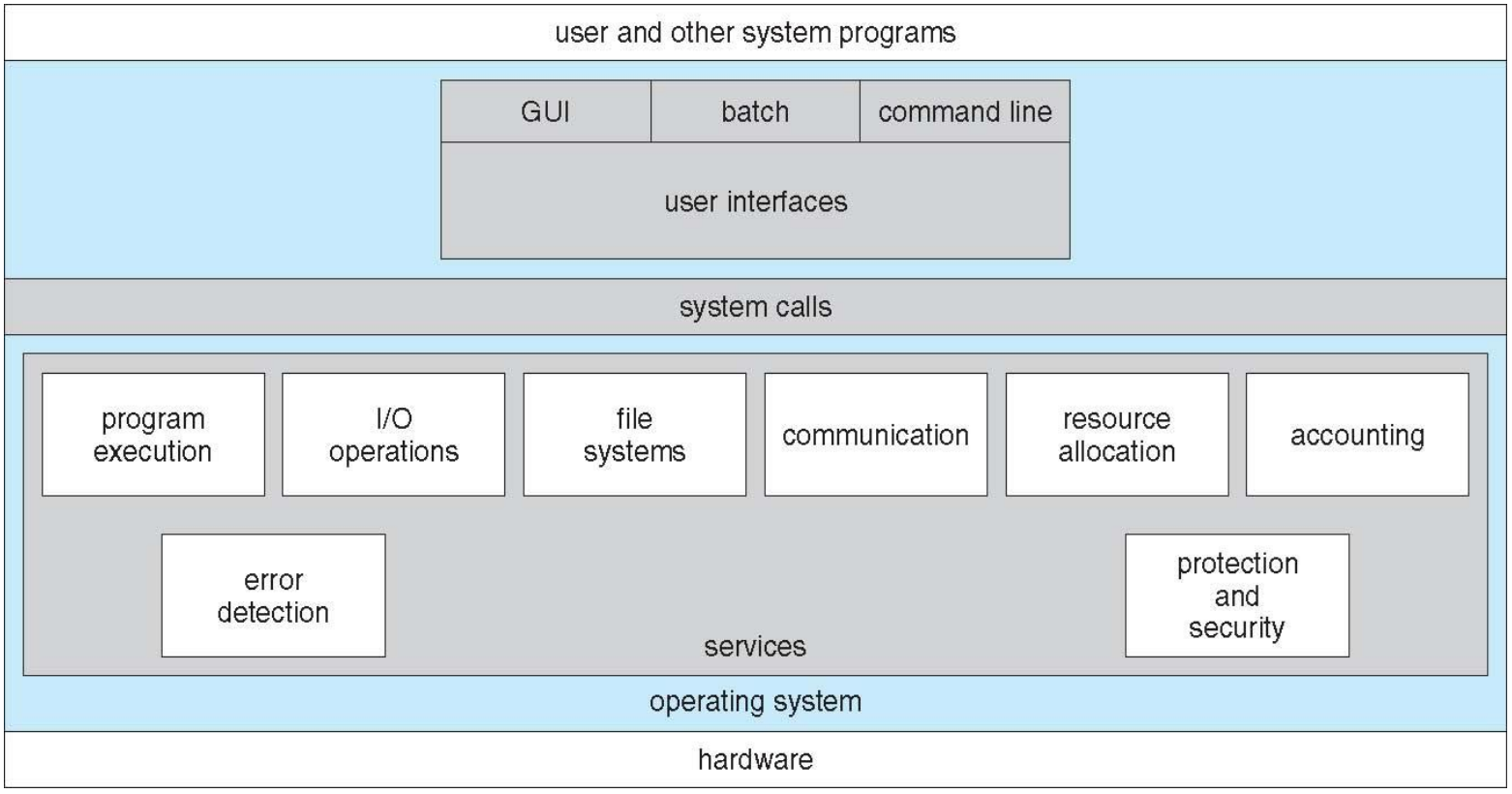
- An operating system (OS) provides the interface between hardware and users.
- OS structure defines how components interact.
- Key topics covered:
  - OS Services
  - System Calls
  - Memory Hierarchy
  - Interrupt Processing
  - OS Design & Implementation

Operating System Services

- User interface - Almost all operating systems have a user interface (UI).
- Program Execution – load, run, terminate.
- I/O Operations – manage input/output devices.
- File System Manipulation – create, delete, read, write.
- Communication – between processes (IPC).
- Error Detection – detect and recover.
- Resource Allocation – CPU, memory, I/O devices.
- Security & Protection – control access.

Example: Windows provides services like file explorer, Linux provides shell-based services.

A View of Operating System Services



System Calls

Interface between user programs and OS kernel.

Categories:

- Process control (fork, exec)
- File manipulation (open, read, write, close)
- Device management
- Information maintenance
- Communication (sockets, pipes)

Example: printf() in C → internally invokes system call write().

Examples of Windows and Unix System Calls

	Windows	Unix
Process Control	CreateProcess() ExitProcess() WaitForSingleObject()	fork() exit() wait()
File Manipulation	CreateFile() ReadFile() WriteFile() CloseHandle()	open() read() write() close()
Device Manipulation	SetConsoleMode() ReadConsole() WriteConsole()	ioctl() read() write()
Information Maintenance	GetCurrentProcessID() SetTimer() Sleep()	getpid() alarm() sleep()
Communication	CreatePipe() CreateFileMapping() MapViewOfFile()	pipe() shmget() mmap()
Protection	SetFileSecurity() InitlializeSecurityDescriptor() SetSecurityDescriptorGroup()	chmod() umask() chown()

Memory Hierarchy

Structure: Registers → Cache → Main Memory → Secondary Storage.
Characteristics:
Speed vs Capacity tradeoff.
Registers (fastest, smallest) → Hard Disk (slowest, largest).
OS responsibility: memory management across these layers.

Example: OS caches frequently used data in RAM to speed up execution.

Interrupt Processing

Interrupts: signals from hardware/software requiring attention.

Steps:

1. Interrupt occurs.
2. CPU saves current state.
3. Control transfers to Interrupt Service Routine (ISR).
4. After ISR, CPU resumes previous task.

Example: Keyboard press, I/O completion, timer interrupt.

OS Design & Implementation

1

Monolithic Kernel: All services in kernel space (Linux)

2

Layered: Built in layers, each built on lower (THE OS)

3

Microkernel: Minimal kernel; services in user space (Minix, QNX)

Monolithic Systems

Definition

- A monolithic operating system is one where all OS functionalities are bundled together into a single large kernel.
- The kernel runs entirely in kernel mode and has complete access to hardware.

Advantages

- High Performance – Since everything is in one layer, there's less communication overhead.
- Simplicity – Easy to design initially because it's just one big program.
- Direct Access to Hardware – Good for speed, as drivers run in kernel mode.

Disadvantages

- Difficult to Maintain – A bug in one part can crash the whole OS.
- Poor Modularity – Hard to add or remove features.
- Security Risks – Since all services share the same space, a compromised driver can affect the entire OS.

Example: MS-DOS

Layered Approach in Operating Systems

Definition

- In the Layered Approach, the operating system is divided into a hierarchy of layers, each built on top of the lower one.
- Each layer only interacts with the layer directly below it, and provides services to the layer above it.

Advantages

- Modularity – Easier to understand and modify.
- Ease of Debugging – Problems can be traced layer by layer.
- Security & Reliability – Layers can enforce restricted access.

Disadvantages

- Performance Overhead – Every request may pass through multiple layers → slower.
- Rigid Design – Strict layering can make it hard to optimize performance if two layers need direct communication.

Examples: THE Operating System, windows NT

Microkernel Architecture

Definition

- A microkernel is a minimalistic kernel design where only the most essential functions run in kernel mode.
- Non-essential services (like file systems, device drivers, networking) run as user-space processes outside the kernel.
- The kernel handles only basic mechanisms:
  - Low-level address space management
  - Thread/process management
  - Inter-Process Communication (IPC)

Advantages

- Reliability & Stability
- Security
- Modularity

Disadvantages

- Performance Overhead
- Complexity of IPC

Examples: Minix, QNX, Mach

Thank you