# Operating System

## Assignment 1

Q1. Compare monolithic and microkernal architectures, highlight their advantages and disadvantages.

⇒ Monolithic Architecture :-

Definition :- A single large kernal where all OS services (process management, file systems, device drivers, memory management, etc.) run in kernal space.

Advantages :-

(i) High performance (fast system calls, direct communicati

(ii) Less overhead due to fewer context switches.

Disadvantages :-

(i) One fault can crash the entire system.

(ii) Complex to debug and maintain.

Example :- Linux, traditional UNIX.

Microkernal Architecture :-

Definition :- Minimul kernal containing only essential services (IPC, scheduling, basic memory mgmt). other services (drivers, file systems) run in user space.

Advantages :-

(i) Fault isolation (a crashed driver doesn't crash the system)

(ii) Moduler, secure, easier to extend.

Disadvantage.

(i) IPC overhead reduces performance.

(ii) More complex message passing.

Example :- QNX, Minix, macOS (tends towards microkernal)

Analogy :- Monolithic = "all teachers in same room = fast but risky";

Microkernal = "teachers in different rooms = safer but slower."

Q.2. Explain the differences between batch, interactive, time-sharing, and real time operating systems with examples.

⇒ Batch OS :- Groups similar jobs; jobs run sequentially with no user interaction.
Example :- payroll, data processing in mainframes.

Interactive OS :- provides immediate user feedback and interaction.
Example :- Windows, Linux desktops.

Time sharing OS :- CPU time divided into small slices (round-robin). Multiple users share system simultaneously.
Example :- UNIX in university labs.

Real-Time OS (RTOS) :- Must respond within strict deadline. Used in mission-critical systems.
Example :- Air traffic control, robotics, medical devices.

Mnemonic :- "B-I-T-R" → Batch, Interactive, Time-sharing, Real-time.

Q.3. Discuss the role of system calls in operating system functionality and provide examples of common system calls. Discuss in detail the structure and component of an operating system, including process, memory file, and security management.

⇒ Role of system calls :
(i) Interface between user programs and OS kernal.
(ii) Provide controlled access to hardware and services.
Examples :
File : open(), read(), write(), close().
Process : fork(), exec(), exit().
Memory : mmap(), brk()
Communication : socket(), pipe().

OS structure and components :
1. Process Management :
(i) Creation, scheduling, termination, IPC.
(ii) Ensure CPU utilization and responsiveness.
2. Memory Management :
(i) Allocation, deallocation, virtual memory, paging.
(ii) Prevents memory leaks and improves utilization.
3. File Management :
(i) Hierarchical file system, permissions, naming, storage alocation.
4. Security Management :
(i) Authentication, authorization, encryption, auditing.
(ii) protects data and system integrity.

Diagram (text)

User Apps → System Calls → OS kernal (process + Memory + File + Security Mgmt) → Hardware

4. Compare multiuser systems and multithreaded systems, explaining their advantages, challengs, and use cases.

⇒ Multiuser Systems :-

(i) Multiple user access system simultaneously (via terminal or network).
(ii) Advantages :- Resource sharing, cost efficiency.
(iii) challenges :- Security, fair scheduling.
(iv) Use cases :- servers, university labs.

Multithreaded systems :-

(i) Multiple threads within a single processes share memory space.
(ii) Advantages :- Faster execution, responsiveness, better CPU utilization.
(iii) challenges :- synchronization issues, race conditions, debugging complexity.
(iv) Use cases :- Web servers, games, real-time apps.

Key Difference :- Multiuser = many users; Multithreaded = many threads in one process.

Q.5. Explain the architecture and functionality of virtual machines and hypervisors, and discuss their importance in modern operating systems.

⇒ Virtual Machines (VMs):

(i) Software abstraction of physical hardware.

(ii) Each VM runs its own OS (guest OS) on shared physical hardware.

(iii) Provides isolation, flexibility, encapsulation.

Hypervisors:

(i) Type 1 (Bare-metal): Runs directly on hardware, efficient, enterprise-grade.
Examples: VMware ESXi, Xen, Hyper-V.

(ii) Type 2 (Hosted): Runs on top of host OS, easier to use but higher overhead.
Examples: VirtualBox, VMware Workstation.

Importance:

(i) Cloud computing backbone.

(ii) Servers consolidation → reduced costs.

(iii) Isolation improves security.

(iv) enables portability, testing environments, disaster recovery.

Diagram (text)

(i) Type 1 : Hardware → Hypervisor → Multiple VMs.

(ii) Type 2 : Hardware → Host OS → Hypervisor → VMs.

Q.6. Analyze the key principles of operating system design, such as simplicity, efficiency, scalability, and their role in performance evaluation.

(i) Simplicity : Keep design modular, easier debugging and maintenance.
Ex- Microkernel keeps only minimal functionality inside kernel.

(ii) Scalability : Handle growth in users, processes, hardware.
Ex- Multicore processors, distributed OS.

(iii) Efficiency : Minimize overhead, maximize performance.
Ex- Efficient CPU scheduling, memory allocation, I/O algorithms.

Performance Metrics :
(i) Throughput = Tasks completed per unit time.
(ii) Response Time = Time between request and response.
(iii) Resource Utilization = % of CPU/memory/I/O used.

Mnemonic : "S-E-S" → Simplicity, Efficiency, Scalability.

[illegible]
15/09/25