# UNIT-I: Operating Systems and Memory Management

## Contents

# 1 Introduction to Operating Systems

An Operating System (OS) is system software that manages computer hardware and software resources, acting as an intermediary between users and hardware to ensure efficient resource utilization and user-friendly interaction.

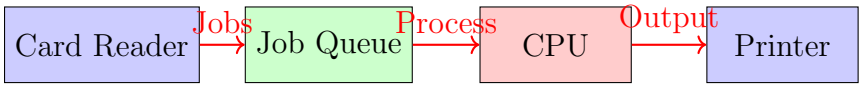
# 2 Types of Operating Systems

## 2.1 Simple Batch Systems

Early systems where jobs were collected and processed sequentially in batches without user interaction. Jobs were submitted via punch cards or tapes and processed one at a time.

- **Features**: Job scheduling, minimal resource sharing, sequential execution.
- **Example**: IBM 1401.

**Diagram:**



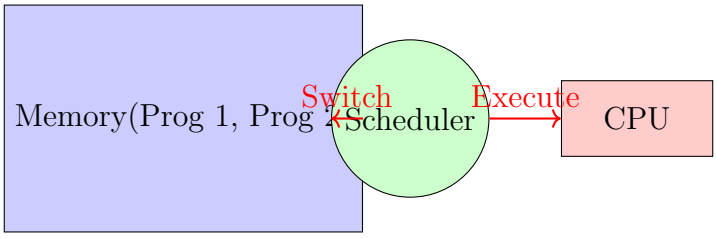

figureSimple Batch System: Sequential job processing from card reader to printer.

## 2.2 Multi-programmed Batch Systems

Multiple programs are loaded into memory and executed concurrently, improving CPU utilization by switching between programs when one waits for I/O.

- **Features**: Job scheduling, memory management, CPU multiplexing.

**Diagram:**



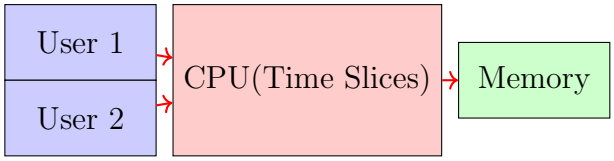

figureMulti-programmed Batch System: CPU switches between programs in memory.

## 2.3 Time-Sharing Systems

An extension of multiprogramming allowing multiple users to interact with the system simultaneously via time-slicing.

- **Features**: Interactive computing, quick response times, resource sharing.
- **Example**: UNIX.

**Diagram:**



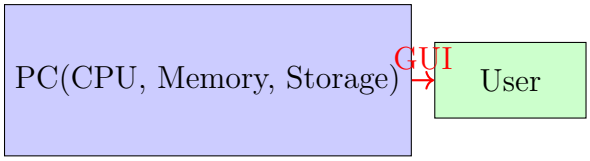

figureTime-Sharing System: Multiple users share CPU via time-slicing.

## 2.4 Personal Computer Systems

Designed for single-user environments, focusing on ease of use and responsiveness.

- **Features**: GUI, single-user multitasking, resource management.
- **Example**: Windows, macOS.

**Diagram:**



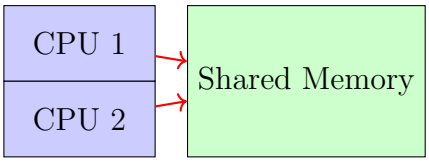

figurePersonal Computer System: Single-user system with GUI.

## 2.5 Parallel Systems

Use multiple processors to execute tasks concurrently, improving performance.

- **Features**: Symmetric/Asymmetric Multiprocessing.
- **Example**: Supercomputers, multi-core systems.

**Diagram:**



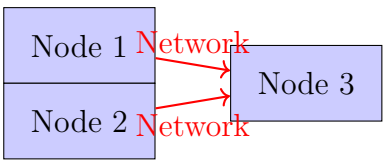

figureParallel System: Multiple CPUs process tasks concurrently.

## 2.6 Distributed Systems

Multiple independent computers work together over a network to achieve a common goal.

- **Features**: Resource sharing, scalability, fault tolerance.
- **Example**: Cloud computing, clusters.

**Diagram:**



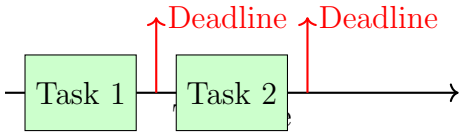

figureDistributed System: Nodes collaborate via network.

### 2.7 Real-Time Systems

Designed for time-critical applications with strict deadlines.

- **Types**: Hard real-time (e.g., avionics), Soft real-time (e.g., streaming).
- **Features**: Deterministic response, high reliability, minimal latency.

**Diagram:**




figureReal-Time System: Tasks with strict deadlines.

## 3 Memory Management

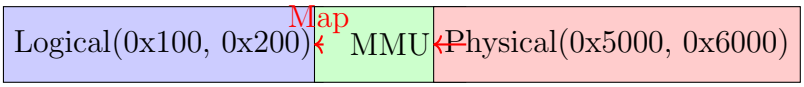
### 3.1 Background

Memory management controls and coordinates computer memory, ensuring efficient allocation to processes and preventing conflicts.

### 3.2 Logical vs. Physical Address Space

- **Logical Address Space**: Addresses generated by a program (virtual).
- **Physical Address Space**: Actual addresses in physical memory.
- Mapping done via Memory Management Unit (MMU).

**Diagram:**



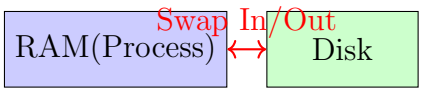

figureLogical to Physical Address Mapping via MMU.

### 3.3 Swapping

Temporarily moves processes from main memory to secondary storage to free memory.

- **Use**: Manages memory in multiprogramming.
- **Drawback**: Slow due to disk I/O.

**Diagram:**



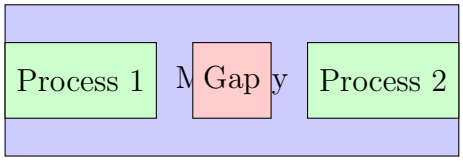

figureSwapping: Moving processes between RAM and disk.

### 3.4 Contiguous Allocation

Each process is allocated a single, contiguous block of memory.

- **Types**: Fixed-size and variable-size partitions.
- **Issues**: External fragmentation (gaps), internal fragmentation (unused space within partitions).

**Diagram:**



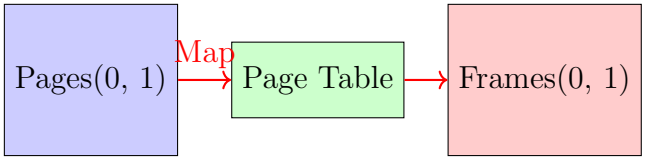

figureContiguous Allocation: Processes with external fragmentation.

## 3.5 Paging

Divides memory into fixed-size pages (logical) and frames (physical), eliminating external fragmentation.

- **Mechanism**: Page table maps pages to frames.
- **Drawback**: Internal fragmentation within pages.

**Diagram:**



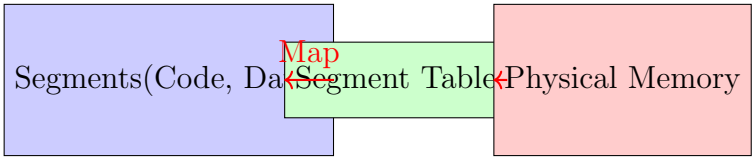

figurePaging: Mapping pages to frames via page table.

## 3.6 Segmentation

Divides memory into variable-sized segments (e.g., code, data, stack).

- **Benefits**: Aligns with program structure.
- **Drawback**: External fragmentation.

**Diagram:**




figureSegmentation: Mapping segments to physical memory.

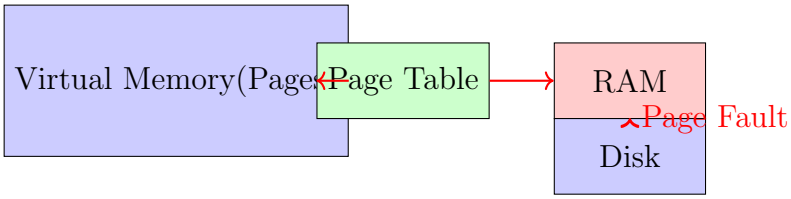
# 4 Virtual Memory

Virtual memory allows programs to use more memory than physically available by using disk space as an extension of RAM.

## 4.1 Demand Paging

Pages are loaded into memory only when needed, using a valid-invalid bit in the page table.

**Diagram:**

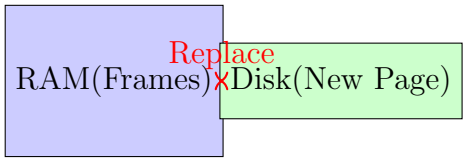



figureDemand Paging: Loading pages from disk on demand.

## 4.2 Page Replacement

Replaces an existing page with a new one when memory is full.

**Diagram:**

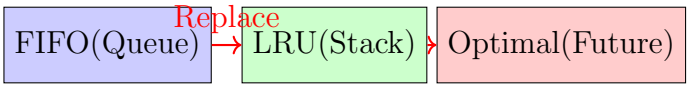



figurePage Replacement: Swapping pages between RAM and disk.

## 4.3 Page Replacement Algorithms

- **FIFO**: Replaces oldest page.
- **LRU**: Replaces least recently used page.
- **Optimal**: Replaces page not used for the longest time (ideal).

**Diagram:**

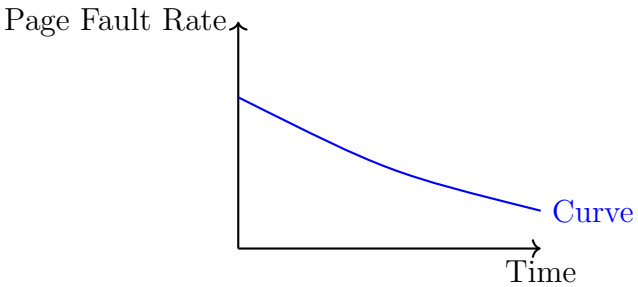



figurePage Replacement Algorithms: FIFO, LRU, and Optimal.

## 4.4 Performance of Demand Paging

Measured by page fault rate, affected by page size, algorithm efficiency, and memory availability.

**Diagram:**

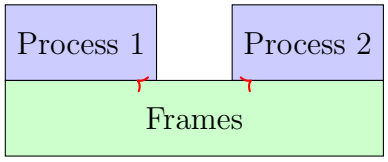



figurePerformance: Page fault rate over time.

## 4.5 Allocation of Frames

Determines how many frames are allocated to each process (equal or proportional).

**Diagram:**

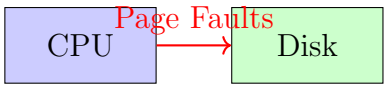



figureAllocation of Frames: Equal or proportional.

## 4.6 Thrashing

Excessive page faults due to insufficient memory, reducing CPU efficiency.

**Diagram:**

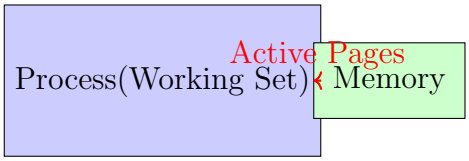



figureThrashing: High page fault rate to disk.

## 4.7 Other Considerations

- **Working Set Model**: Tracks active pages to prevent thrashing.
- **Page Size**: Balances fragmentation and page table size.
- **I/O Interlock**: Prevents swapping pages during I/O.
- **Prepaging**: Loads pages before needed to reduce faults.

**Diagram (Working Set Model):**




figureWorking Set Model: Tracking active pages.