Operating System Cheatsheet by Techie CodeBuddy

(This cheatsheet is free onlt, don't pay even a rupee to anyone)

Key areas of Focus:

- **Introduction to Operating Systems**

  ▪ Types of operating systems, including Batch, Time-Sharing, Distributed, Real-Time, and Embedded systems.

  ▪ Architecture of operating systems, including the kernel, shell, system calls, and user interface.

- **Process management**
  - Process lifecycle (creation, execution, termination)
  - Context switching and process states
  - Thread

- **Process scheduling**
  - Scheduling algorithms (e.g., round-robin, priority-based, shortest job first)
  - Multilevel queue scheduling
  - Real-time scheduling

- **File management system**
  - File organization (directories, files, paths)
  - File permissions (read, write, execute)
  - File system types (e.g., ext4, NTFS, FAT)

- **Memory management**
  - Virtual memory (paging, segmentation)
  - Memory allocation (malloc, free)

- **Process Scheduling**
- **Deadlocks**
- **Thread**
- **Virtual memory management**
- **Disk Scheduling**
- **Multithreaded programming**

Let's Begin

Introduction to Operating Systems

An Operating System (OS) is software that acts as an intermediary between computer hardware and the user. It manages hardware resources, provides common services for computer programs, and facilitates user interaction with the system.

Types of Operating Systems

1. Batch Operating System :

- **Definition:** Executes jobs in batches without user interaction.

- **Example:** Early mainframes where jobs were collected, processed, and output in groups.

- Real-World Use Case: Used in large-scale data processing tasks like payroll systems.

2. Time-Sharing Operating System:

- Definition : Allows multiple users to share system resources simultaneously by dividing the time into small intervals.

- Example : UNIX, where each user gets a time slice to execute commands.

- Real-World Use Case : Multi-user environments in academic institutions or research labs.

3. Distributed Operating System :

- Definition : Manages a group of independent computers and makes them appear as a single coherent system to users.

- Example : Google's Android OS managing multiple servers and devices in a cloud.

- Real-World Use Case : Cloud computing services like AWS, where resources are distributed across multiple servers.

4. Real-Time Operating System (RTOS) :

- Definition : Provides immediate processing and response to input, ensuring time constraints are met.

- Example : VxWorks used in aerospace systems.

- Real-World Use Case : Embedded systems in medical devices or autonomous vehicles where timing is crucial.

5. Embedded Operating System :

- Definition : Designed for specialized hardware and performs dedicated tasks.

- Example : Embedded Linux in smart TVs or IoT devices.

- Real-World Use Case : Operating systems in household appliances like washing machines or microwave ovens.

Architecture of Operating Systems

1. Kernel :

- Definition : The core part of the OS responsible for managing system resources, hardware, and system calls.

- Example : Linux kernel managing system operations and resources.

- Real-World Use Case : Handles tasks like memory management, process scheduling, and device control.

2. Shell :

- Definition : The user interface that allows users to interact with the OS through commands or graphical elements.

- Example : Command Prompt in Windows or Bash in Linux.

- Real-World Use Case : Allows users to execute commands, run programs, and manage files.

3. System Calls :

- Definition : Interfaces through which user programs request services from the OS.

- Example : `open()` for file operations or `fork()` for creating processes.

- Real-World Use Case : Allows programs to perform operations like reading/writing files, creating processes, or managing memory.

4. User Interface :

- Definition : The component of the OS that allows users to interact with the system through graphical or command-based interfaces.

- Example : Windows Desktop environment or macOS Finder.

- Real-World Use Case : Provides a means for users to interact with applications and manage system settings.

Process Management

1. Process Lifecycle

- Creation :

  - Definition : The process lifecycle begins when a process is created. This involves allocating resources and initializing process control blocks.

  - Example : When you open a new application on your computer, the OS creates a new process for that application.

  - Real-World Use Case : Starting a web browser creates a new process for handling the browser's tasks.

- Execution :

  - Definition : Once created, the process moves into the execution phase where it is assigned CPU time and executes instructions.

  - Example : While browsing a website, the browser process is actively executing code to render the page.

  - Real-World Use Case : Running a video game where the game's process is actively being executed to display graphics and respond to user inputs.

- Termination :

  - Definition : The process lifecycle ends when the process has completed its execution or is terminated by the OS.

  - Example : Closing a document in a word processor ends the process associated with that document.

  - Real-World Use Case : Exiting a software application or when an application crashes, its process is terminated.

2. **Context Switching and Process States**

- Context Switching :

- Definition : The process of saving the state of a currently running process and loading the state of another process to allow multitasking.

- Example : Switching between multiple open applications on your computer.

- Real-World Use Case : In a multi-tasking environment, context switching enables smooth operation of multiple applications by periodically switching CPU control among them.

- Process States :

- Definition : Different stages a process can be in during its lifecycle:

  - New : Process is being created.

  - Ready : Process is waiting to be assigned CPU time.

  - Running : Process is currently executing.

  - Waiting : Process is waiting for an event or resource (e.g., I/O operation).

  - Terminated : Process has finished execution or is terminated.

- Example : When you open an application, it moves from the new state to the ready state and then to running as it starts executing.

- Real-World Use Case : An application waiting for user input is in the waiting state, and once input is provided, it moves back to the running state.

3. Thread

- Definition : A thread is the smallest unit of execution within a process. Threads share the same resources and memory space of the process but can execute independently.

- Example : A web browser might use multiple threads to handle different tabs or tasks, like loading content and processing user input simultaneously.

- Real-World Use Case : In a word processor, one thread might handle typing input while another handles spell-checking, allowing both operations to happen concurrently.

Process Scheduling

1. Scheduling Algorithms

- Round-Robin Scheduling :

- Definition : A preemptive scheduling algorithm where each process is assigned a fixed time slice or quantum. After the quantum expires, the process is moved to the back of the queue.

- Example : If you have three tasks, A, B, and C, and each gets a 10-millisecond time slice, the scheduler will allocate CPU time as A (10ms), B (10ms), C (10ms), and repeat.

- Real-World Use Case : Suitable for time-sharing systems where all processes need fair CPU access, such as in a multi-user system like UNIX.

- Priority-Based Scheduling :

- Definition : Processes are assigned priorities, and the scheduler selects the process with the highest priority for execution. In some variants, a process's priority may change over time.

- Example : In a print queue, a high-priority document might be printed before a lower-priority one.

- Real-World Use Case : Used in systems where certain tasks need to be completed sooner, like handling critical system processes or urgent user requests.

- Shortest Job First (SJF) :

- Definition : A non-preemptive scheduling algorithm where the process with the shortest estimated runtime is executed first.

- Example : If you have tasks with estimated times of 5ms, 10ms, and 15ms, the scheduler will execute the 5ms task first.

- Real-World Use Case : Often used in batch processing where tasks with shorter execution times are prioritized to reduce overall completion time.

2. Multilevel Queue Scheduling

- Definition : A scheduling method where processes are divided into multiple queues based on their characteristics (e.g., foreground vs. background). Each queue can have its own scheduling algorithm.

- Example : An operating system might have separate queues for interactive processes (using round-robin) and batch processes (using SJF).

- Real-World Use Case : In a desktop OS, foreground applications (like a web browser) might be given higher priority over background tasks (like virus scans) to ensure responsiveness.

3. Real-Time Scheduling

- Definition : Scheduling algorithms designed to meet deadlines for real-time systems, where tasks must be completed within strict timing constraints.

- Types :

- Hard Real-Time : Tasks must be completed by their deadlines with no exception. Example: medical equipment where failure to meet deadlines could be critical.

- Soft Real-Time : Tasks should ideally meet deadlines but can tolerate some delays. Example: video streaming where occasional lag is acceptable.

- Example : Rate Monotonic Scheduling (RMS) and Earliest Deadline First (EDF) are common real-time scheduling algorithms.

- Real-World Use Case : Used in embedded systems, like automotive control systems, where timely execution is crucial for safety and performance.

File Management System

1. File Organization

- Directories :

- Definition : Containers that hold files and other directories, helping to organize data into a hierarchical structure.

- Example : A directory named `Documents` might contain subdirectories like `Work` and `Personal`.

- Real-World Use Case : Organizing files on your computer's desktop into folders like `Projects`, `Photos`, and `Downloads` for easier access.

- Files :

- Definition : Collections of data stored on disk, each identified by a name and extension (e.g., `report.docx`).

- Example : A file named `resume.pdf` contains your resume data.

- Real-World Use Case : A text document, image, or video file stored in a specific directory for various uses.

- Paths :

- Definition : Addresses that specify the location of a file or directory within the file system. Paths can be absolute (starting from the root) or relative (starting from the current directory).

- Example : An absolute path might be `/home/user/Documents/report.docx`, while a relative path might be `Documents/report.docx` if you are already in the `user` directory.

- **Real-World Use Case :** Navigating to a file in a specific directory using file explorers or command-line interfaces.

2. File Permissions

- **Read :**

- Definition : Allows a user to view the contents of a file.

- Example : A user with read permission can open and view the contents of `document.txt`.

- Real-World Use Case : Ensuring that users can view files without making changes.

- **Write :**

- Definition : Allows a user to modify or delete the file.

- Example : A user with write permission can edit and save changes to `data.csv`.

- Real-World Use Case : Allowing users to update and manage content in documents or configuration files.

- **Execute :**

- Definition : Allows a user to run a file as a program or script.

- Example : A user with execute permission can run a script file like `install.sh`.

- Real-World Use Case : Executing a program or script that performs specific tasks on the system.

3. File System Types

- **ext4 (Extended File System 4) :**

  - Definition : A journaling file system commonly used in Linux, providing features like large file support, improved performance, and reliability.

  - Example : Linux distributions like Ubuntu use ext4 for system and data partitions.

  - Real-World Use Case : Used in Linux-based systems for managing files with robust data integrity.

- **NTFS (New Technology File System) :**

  - Definition : A file system used by Windows operating systems that supports large files, advanced permissions, and file system journaling.

  - Example : The default file system for Windows 10 and Windows 11 installations.

  - Real-World Use Case : Provides high performance and security features for Windows users.

- **FAT (File Allocation Table) :**

  - Definition : An older file system with various versions (FAT16, FAT32) used for compatibility across different operating systems. It's simpler but has limitations on file size and volume size.

  - Example : FAT32 is often used in USB drives and SD cards for compatibility across different devices.

  - Real-World Use Case : Commonly used in removable storage devices like flash drives and memory cards for easy file sharing between different systems.

Memory Management

1. Virtual Memory

Paging :

- Definition : A memory management scheme that eliminates the need for contiguous allocation of physical memory. The process is divided into small blocks called "pages," which are mapped to physical memory frames.

- Example : If a process requires 10 pages of memory, these pages can be scattered across various physical memory locations.

- Real-World Use Case : Enables efficient memory use and multitasking in modern operating systems, such as Windows or Linux, allowing large applications to run even with limited physical memory.

- Segmentation :

- Definition : A memory management technique where memory is divided into segments based on logical divisions, like code, data, and stack. Each segment can grow and shrink independently.

- Example : A program might have separate segments for code, heap, and stack, each with different sizes and permissions.

- Real-World Use Case : Useful in systems where programs have logically distinct sections that need different memory management strategies, such as in certain academic or enterprise applications.

2. Memory Allocation

- malloc (Memory Allocation):

- Definition : A function in C/C++ used to allocate a block of memory on the heap. It returns a pointer to the beginning of the block.

- Syntax : `void* malloc(size_t size);`

- Example : `int* arr = (int*)malloc(10 * sizeof(int));` allocates memory for an array of 10 integers.

- Real-World Use Case : Used in programs where dynamic memory allocation is needed, such as creating arrays or data structures whose size can change at runtime.

- free :

- Definition : A function in C/C++ used to deallocate memory that was previously allocated with `malloc`. It releases the allocated memory back to the system.

- Syntax : `void free(void* ptr);`

- Example : `free(arr);` releases the memory allocated for the `arr` array.

- Real-World Use Case : Prevents memory leaks by ensuring that dynamically allocated memory is properly deallocated when it is no longer needed.

Process Scheduling

1. Process Scheduling Algorithms :

- Round-Robin : Allocates a fixed time slice to each process in a cyclic order.

- Priority-Based : Processes are assigned priorities; the highest priority process is executed first.

- Shortest Job First (SJF) : Executes the process with the shortest estimated run time first.

- Real-World Use Case : Ensures efficient CPU usage and response times, balancing fairness and performance in multi-tasking environments.

2. Multilevel Queue Scheduling :

- Definition : Processes are divided into different queues based on their characteristics (e.g., foreground, background), and each queue uses a different scheduling algorithm.

- Real-World Use Case : Improves system performance by prioritizing interactive processes over batch processes.

3. Real-Time Scheduling :

- Definition : Designed to meet strict timing constraints for real-time systems.

- Types :

  - Hard Real-Time : Must meet deadlines with no exceptions (e.g., medical devices).

  - Soft Real-Time : Deadlines are preferable but not critical (e.g., video streaming).

Deadlocks

1. Definition :

- A situation where a set of processes are blocked because each process is holding a resource and waiting for another resource held by another process.

2. Conditions for Deadlock :

- Mutual Exclusion : Only one process can use a resource at a time.

- Hold and Wait : Processes holding resources can request additional resources.

- No Preemption : Resources cannot be forcibly taken from a process.

- Circular Wait : A circular chain of processes exists where each process holds a resource needed by the next.

3. Avoidance and Prevention :

- Avoidance : Use algorithms like Banker's Algorithm to avoid deadlock by ensuring resource allocation does not lead to unsafe states.

- Prevention : Ensure at least one of the necessary conditions for deadlock is not met (e.g., by requiring processes to request all resources at once).

Thread

1. Definition :

- The smallest unit of execution within a process, sharing the process's resources but able to execute independently.

2. Advantages :

- Concurrency : Multiple threads can run concurrently, improving performance on multi-core processors.

- Resource Sharing : Threads share resources within the same process, which reduces overhead compared to separate processes.

3. Real-World Use Case :

- Web Browsers : Use threads to handle multiple tabs and background tasks simultaneously.

Virtual Memory Management

1. Paging :

- Definition : Divides process memory into fixed-size pages that are mapped to physical memory frames, allowing non-contiguous allocation.

- Real-World Use Case : Enables efficient use of memory and multitasking by handling large applications and processes.

2. Segmentation :

- Definition : Divides memory into segments based on logical divisions (e.g., code, data, stack), allowing each segment to grow independently.

- Real-World Use Case : Useful in complex applications where different memory areas have different requirements.

Disk Scheduling

1. First-Come, First-Served (FCFS) :

- Definition : Services disk I/O requests in the order they arrive.

- Real-World Use Case : Simple but can be inefficient if requests are not sequential.

2. Shortest Seek Time First (SSTF) :

- Definition : Selects the disk I/O request with the shortest seek time.

- Real-World Use Case : Reduces the average seek time by handling the closest requests first.

3. Elevator (SCAN) :

- Definition : Moves the disk arm in one direction, servicing requests until it reaches the end, then reverses direction.

- Real-World Use Case : Provides a more balanced approach by reducing overall seek time compared to FCFS.

Multithreaded Programming

1. Definition :

- Programming approach where multiple threads are used within a single process to perform concurrent tasks.

2. Advantages :

- Responsiveness : Improves responsiveness in applications by handling multiple tasks simultaneously.

- Resource Sharing : Threads share memory and resources, which is more efficient than using separate processes.

3. Real-World Use Case :

- Server Applications : Web servers and database servers use multithreading to handle multiple client requests simultaneously.

Operating System One Shot Video (HINDI): Click Here