Operating Systems Cheatsheet

A comprehensive cheat sheet covering fundamental concepts in Operating Systems, including process management, memory management, file systems, and I/O systems. This guide provides concise explanations and examples for quick reference.

Process Management

Process States

New	The process is being created.
Ready	The process is waiting to be assigned to a processor.
Running	Instructions are being executed.
Waiting	The process is waiting for some event to occur (e.g., I/O completion or reception of a signal).
Terminated	The process has finished execution.

Process Scheduling Algorithms

First-Come, First-Served (FCFS)	Processes are executed in the order they arrive. Simple but can lead to long waiting times.
Shortest Job First (SJF)	Processes with the shortest execution time are executed first. Minimizes average waiting time.
Priority Scheduling	Processes are executed based on priority. Higher priority processes are executed first.
Round Robin	Each process gets a fixed time slice (quantum). If a process doesn't complete in its quantum, it's moved to the end of the queue.

Inter-Process Communication (IPC)

IPC mechanisms facilitate communication between processes.

Common methods include:

- **Shared Memory:** Processes access a common memory region.
- **Message Passing:** Processes exchange messages.
- **Pipes:** Data flows unidirectionally between processes.
- **Sockets:** Enables communication over a network.

Memory Management

Memory Allocation Techniques

Contiguous Allocation	Each process is allocated a contiguous block of memory. Simple but can lead to fragmentation.
Non-Contiguous Allocation	Processes are allocated memory in non-contiguous blocks. Reduces fragmentation but more complex.
Paging	Memory is divided into fixed-size blocks called pages. Processes are also divided into pages, allowing non-contiguous allocation.
Segmentation	Memory is divided into logical segments. Each segment represents a logical unit (e.g., code, data, stack).

Virtual Memory

Virtual memory allows processes to use more memory than physically available.

- **Demand Paging:** Pages are loaded into memory only when needed.
- **Page Replacement Algorithms:** Algorithms to decide which page to replace when a new page needs to be loaded (e.g., FIFO, LRU, Optimal).

Page Replacement Algorithms

FIFO (First-In, First-Out)	The oldest page in memory is replaced.
LRU (Least Recently Used)	The page that has not been used for the longest time is replaced.
Optimal	The page that will not be used for the longest time in the future is replaced. (Theoretical, not practical).

File Systems

File System Structures

A file system organizes files and directories on a storage device.

Key components include:

- **Directory Structure:** Hierarchical organization of directories and files.
- **File Metadata:** Information about files (e.g., name, size, timestamps, permissions).
- **Allocation Methods:** Methods for allocating disk space to files (e.g., contiguous, linked, indexed).

Common File Systems

FAT32	File Allocation Table 32. Simple file system, widely compatible but with limitations on file size.
NTFS	New Technology File System. Advanced file system with features like security, compression, and journaling.
ext4	Fourth extended file system. Default file system for many Linux distributions, offering good performance and scalability.

File Access Methods

Sequential Access	Files are accessed in order, one record after another.
Direct Access	Files can be accessed in any order. Requires a way to determine the location of a specific record.
Indexed Sequential Access	An index is used to locate specific records, allowing both sequential and direct access.

I/O Systems

I/O Hardware

I/O hardware components facilitate communication between the computer and external devices.

Key components include:

- **Controllers:** Electronic components that manage I/O devices.
- **Device Drivers:** Software that interfaces with controllers.
- **I/O Ports:** Physical interfaces for connecting devices.

I/O Techniques

Polling	The CPU repeatedly checks the status of an I/O device. Simple but inefficient.
Interrupts	The I/O device signals the CPU when it's ready for data transfer. More efficient than polling.
Direct Memory Access (DMA)	The I/O device transfers data directly to or from memory without CPU intervention. Very efficient for large data transfers.

Buffering and Caching

Buffering and caching are used to improve I/O performance.

- **Buffering:** Storing data temporarily in a buffer to handle speed mismatches between devices.
- **Caching:** Storing frequently accessed data in a cache for faster retrieval.