presents

OS^2

OPERATING SYSTEMS FOR OPEN SOURCE

Electronics Club IITK

WHAT IS A PROCESS?

A process in an operating system (OS) is a program in execution. It’s one of the most fundamental concepts in an OS and represents the dynamic state of a program as it runs.

01

PROGRAM CODE

The instructions that the CPU executes.

02

CURRENT ACTIVITY

Includes the program counter (which tracks the next instruction to execute), registers, and variables.

03

MEMORY

- **Code Section**: Contains the program’s executable code.
- **Data Section**: Contains global variables.

WHAT IS A THREAD?

A thread is the smallest unit of execution within a process. Multiple threads can exist within the same process, sharing the process's resources like memory and file handles.

Example: A web browser's process might have multiple threads—one for rendering the page, one for handling user input, and another for downloading data.

KEY DIFFERENCES B/W PROCESSES AND THREADS

ASPECT	PROCESS	THREAD
Definition	Independent program in execution	Smallest unit of execution in a process
Memory	Separate memory space	Shared memory within the process
Overhead	High (resource-intensive)	Low (lightweight)
Dependency	Independent of other processes	Dependent on the parent process
Example	Running a music player	Threads in a music player for playback and UI

VIRTUALIZATON

Virtualization abstracts physical hardware (CPU, memory, storage) to create virtual environments.

Examples:

- Virtual Machines (VMware, VirtualBox).
- Containers (Docker, Kubernetes).

01

MEMORY VIRTUALIZATION

Maps virtual memory addresses to physical addresses using page tables.

02

COMPUTE VIRTUALIZATION

Creates virtual CPUs that share physical CPU cycles (e.g., VMs).

CONCURRENCY

RUNNING PROCESSES IN PARALLEL

01

MULTI-THREADING

Threads within a process share resources and run in parallel.

02

MULTI-PROCESSING

Multiple processes run independently on different CPUs or cores.

SCHEDULING MECHANISM

- **Scheduling is the process of selecting which process will execute on the CPU.**
- **Ensures efficient resource utilization and improves system performance.**
- **Objectives:**
  - Maximize CPU utilization.
  - Minimize waiting and response times.
  - Ensure fairness among processes.

01

READY QUEUE

- A queue of processes waiting to execute is maintained.
- The scheduler selects the next process to run based on a scheduling policy and performs a context switch if required.
- Processes can transition between Running, Ready, and Waiting states.

02

IDLE PROCESS

If no processes are in the ready queue, an idle process executes an instruction like HLT (halt), assembly language instruction which halts the central processing unit (CPU) until the next external interrupt is fired.

03

INVOCATION TYPES

- **Non-Preemptive Scheduling:** The process triggers the scheduler upon completion.
- **Preemptive Scheduling:** The OS interrupts a process to schedule another.

SCHEDULING POLICIES

01

First Come First Serve (FCFS)

- Processes are executed in the order of arrival (FIFO).
- **Advantages**: Easy to implement.
- **Disadvantages**: Convoy effect; not suitable for interactive systems.

02

Shortest Job First (SJF)

- Selects the process with the shortest CPU burst time.
- **Advantages:** Optimal for waiting and turnaround time.
- **Disadvantages**: Unrealistic since burst time prediction is required.

03

Shortest Time to Completion First (STCF)

- Preemptive version of SJF, selecting the process with the shortest remaining time.
- **Advantages**: Improves SJF's efficiency at the cost of increased context switches.

04

Round Robin (RR)

- Time-sliced preemptive scheduling using a circular queue.
- **Design Choice**: Time quantum size affects performance.
- **Advantages**: Good response time for interactive tasks.
- **Disadvantages**: High scheduling overhead if the quantum is too small.

05

Priority Scheduling

- Processes are selected based on priority, either preemptively or non-preemptively.
- **Advantages**: Practical for prioritizing tasks.
- **Disadvantages**: Starvation for low-priority processes.

SCHEDULING IS MORE COMPLEX IN REAL OS

01

Different Requirements of Different Processes

- **Real-time processes**: Should meet strict deadlines
- **Interactive processes**: Responsive scheduling
- **Batch processes**: Starvation free scheduling

02

Well intentioned users should be able to influence the scheduling policy in a positive manner

03

Greed of greedy users should be controlled by the OS

THANK YOU!