Process Scheduling

Operating Systems 1: Process Scheduling

1. Definition and Objectives of Scheduling

Scheduling governs the transitions between different process states. Its objectives are:

1. To maximize CPU utilization;
2. To ensure fairness among processes;
3. To provide acceptable response times to users;
4. To maintain good system throughput;
5. To enforce certain priorities.

If multiple processes are in the "ready" state, the scheduler selects which one to execute next (moving it to the "running" state) using an algorithm that implements a scheduling policy. Scheduling can be preemptive or non-preemptive. It is *non-preemptive* if the CPU is retained by a process until it finishes. It is *preemptive* otherwise.

---

2. FCFS Policy (First-Come, First-Served)

This is the simplest algorithm: the process that requests the CPU first is the first to be served. Once the running process finishes, the CPU is allocated to the next. It is simple to implement (e.g., using a queue), but its performance is limited.

---

3. SJF Policy (Shortest Job First)

This algorithm associates an estimated execution time to each process. The process with the shortest estimated time is selected. This favors short jobs and is commonly used in batch systems. The main difficulty is knowing the execution times in advance.

SJF can be preemptive or not. The preemptive version is known as **SRTF (Shortest Remaining Time First)**. It addresses the following:

If a new process arrives with an estimated time shorter than the remaining time of the running process, the current one is interrupted and the CPU is allocated to the new process.

---

4. Round Robin (RR) Policy

This popular and reliable algorithm is designed for time-sharing systems. Each ready process is given a fixed time slice (quantum). When a process uses up its quantum, finishes earlier, or is blocked (e.g., for I/O), the next process in the queue is scheduled. The interrupted process goes to the end of the queue.

The key parameter is the **quantum duration**. It should minimize system overhead while remaining user-friendly. A typical compromise is a quantum of 100–200 ms.

---

5. Highest Priority Policy

In Round Robin, all processes are treated equally. Sometimes, certain processes must be prioritized. The priority algorithm selects the ready process with the highest priority (usually the smallest numeric value).

It can be **preemptive** or **non-preemptive**:

- In preemptive mode: if a new higher-priority process arrives, it interrupts the current one.
- In non-preemptive mode: it is inserted at the appropriate position in the ready queue.

Priorities can be **static** (e.g., system processes) or **dynamic** (e.g., user processes whose priority may decrease after using the CPU).

---

6. Round Robin with Priorities

This is a hybrid: each priority level has its own Round Robin queue. The scheduler selects the highest-priority non-empty queue and serves it.

To ensure all processes eventually run, priorities must be periodically adjusted.

---

7. Thread Scheduling

Threads are also scheduled:

- **Windows NT**: 32 priority levels (fixed or dynamic). The highest-priority thread runs. Dynamic priorities vary within bounds and may increase after I/O wait.
  Use `SetThreadPriority()` to change thread priority.
- **Java**: Priorities range from `Thread.MIN_PRIORITY` to `Thread.MAX_PRIORITY`, with normal being `Thread.NORM_PRIORITY`. Threads with equal priority may be scheduled via round robin (with or without quantum limits), depending on the interpreter or machine (e.g., IBM 580 vs. Sun E3000).

Exercises – OS1: Process Scheduling

Exercise 1

Given the following processes:

Process	Estimated Execution Time	Priority
P1	10	3
P2	1	1
P3	2	3
P4	1	4
P5	5	2

Assume arrival order is roughly at T=0 and that a lower value means higher priority.
The **response time** is the duration between arrival and completion.

Determine response times and the average for each method:

1. FCFS
2. SJF
3. Round Robin (quantum = 1, then 2)
4. Non-preemptive priority

Exercise 2

Given:

Process	Arrival Time	Estimated Exec Time	Priority
P1	0	8	3
P2	1	4	1
P3	2	1	2

Calculate response time and average for:

1. FCFS
2. Non-preemptive priority
3. Preemptive priority
4. SJF
5. SRTF

Exercise 3

Five jobs (A–E) arrive at the same time. Execution times: 10, 6, 2, 4, and 8 mins. Priorities: 3, 5, 2, 1, and 4.

Calculate average waiting and response times for:

- Round Robin (quantum = 1)
- Priority-based scheduling
- FCFS
- SJF

Notes:

- For RR: assume multiprogramming and fair CPU sharing.
- For others: each job runs until completion.
- No I/O involved.

Exercise 4

Simulate two processes (P1 and P2) with:

- **Dynamic priorities** (1–5): 5 is highest. If a process doesn't use its full quantum, its priority decreases.
- **Dynamic quantums** (1–5): If unused fully, the quantum increases.
- Priorities and quantum's vary inversely.
- **Context switch time**: 1 time unit.
- **Scheduler overhead**: negligible.

Processes:

- Submitted simultaneously
- Initial quantums: P1 = 2, P2 = 5
- Initial priorities: P1 = 2, P2 = 3
- Execution time: 16 units each (no I/O)
- P1 triggers I/O at time 4 and 10 (each 1 unit)
- P2 triggers I/O at time 3 (7 units) and 11 (3 units)

Determine response times with execution scenario.