CPU Scheduling Overview

CPU Scheduling

CPU scheduling is the basis of multiprogrammed operating systems. By switching the CPU among processes, the operating system can make the computer more productive. In this section, we introduce basic CPU -scheduling concepts and present several CPU -scheduling algorithms. We also consider the problem of selecting an algorithm for a particular system.

Basic Concept

In a single-processor system, only one process can run at a time. Others must wait until the CPU is free and can be rescheduled. The objective of multiprogramming is to have some process running at all times, to maximize CPU utilization. The idea is relatively simple. A process is executed until it must wait, typically for the completion of some I/O request. In a simple computer system, the CPU then just sits idle. All this waiting time is wasted; no useful work is accomplished. With multiprogramming, we try to use this time productively. Several processes are kept in memory at one time.

When one process has to wait, the operating system takes the CPU away from that process and gives the CPU to another process. This pattern continues. Every time one process has to wait, another process can take over use of the CPU .
Scheduling of this kind is a fundamental operating-system function. Almost all computer resources are scheduled before use. The CPU is, of course, one of the primary computer resources. Thus, its scheduling is central to operating-system design.
There are two types of Scheduling scheme. We can say **Non-preemptive** and **Preemptive** .

Non-preemptive : Under non-preemptive scheduling, once the CPU has been allocated to a process, the process keeps the CPU until it releases the CPU either by terminating or by switching to the waiting state. This scheduling method is used by the Microsoft Windows 3.1 and by the Apple Macintosh operating systems.
It is the only method that can be used on certain hardware platforms, because It does not require the special hardware(for example: a timer) needed for preemptive scheduling.

Preemptive : In this type of Scheduling, the tasks are usually assigned with priorities. At times it is necessary to run a certain task that has a higher priority before another task although it is running. Therefore, the running task is interrupted for some time and resumed later when the priority task has finished its execution.

Difference Between Preemptive and Non-preemptive Scheduling

Preemptive	Non-preemptive
Processor can be preempted to execute a different process in the middle of execution of any current process.	Once processor starts to execute a process it must finish it before executing other. It can't be paused in middle.
CPU utilization is more compared to non-preemptive scheduling.	CPU utilization is less compared to preemptive scheduling.
Waiting time and response time is less.	Waiting time and response time is more.
Preemptive scheduling is flexible.	Non-preemptive scheduling is rigid.
The Preemptive scheduling is prioritized. The highest priority process should always be the	When a process enters the state of running, the state of that process isn't deleted from the

process that is currently utilized.	scheduler until it finishes its service time.
If a high priority process frequently arrives in the ready queue, low priority process may starve.	If a process with long burst time is running CPU, then another process with less CPU burst time may starve.
Ex : SRTF, Round Robin,etc.	Ex : FCFS, SJF, Priority, etc.

Dispatcher

Another component involved in the CPU -scheduling function is the dispatcher. The dispatcher is the module that gives control of the CPU to the process selected by the short-term scheduler. This function involves the following:

- Switching context
- Switching to user mode
- Jumping to the proper location in the user program to restart that program

The dispatcher should be as fast as possible, since it is invoked during every process switch. The time it takes for the dispatcher to stop one process and start another running is known as the dispatch latency.

Scheduling Criteria

CPU utilization : We want to keep the CPU as busy as possible. Conceptually, CPU utilization can range from 0 to 100 percent. In a real system, it should range from 40 percent (for a lightly loaded system) to 90 percent (for a heavily loaded system).

Throughput : If the CPU is busy executing processes, then work is being done. One measure of work is the number of processes that are completed per time unit, called throughput. For long

processes, this rate may be one process per hour; for short transactions, it may be ten processes per second.

Turnaround time : From the point of view of a particular process, the important criterion is how long it takes to execute that process. The interval from the time of submission of a process to the time of completion is the turnaround time. Turnaround time is the sum of the periods spent waiting to get into memory, waiting in the ready queue, executing on the CPU , and doing I/O .

Waiting time : The CPU -scheduling algorithm does not affect the amount of time during which a process executes or does I/O . It affects only the amount of time that a process spends waiting in the ready queue. Waiting time is the sum of the periods spent waiting in the ready queue.

Response time : In an interactive system, turnaround time may not be the best criterion. Often, a process can produce some output fairly early and can continue computing new results while previous results are being output to the user. Thus, another measure is the time from the submission of a request until the first response is produced. This measure, called response time, is the time it takes to start responding, not the time it takes to output the response. The turnaround time is generally limited by the speed of the output device.

Various CPU Scheduling Algorithms

- FCFS (First Come First Serve)
- SJF (Shortest Job First)
- Priority Scheduling
- Round Robin

We will discuss these algorithms in the next each section.