

## CPU Scheduling

07/10/2006 00:00 by the6dmin

### Processor Scheduling Policies

Scheduling determines which process is chosen to make the transition from the ready state at the processor waiting queue to actually running on the processor. In a multiprocessor system, the scheduling will involve more than one processor, but at any given time, there will be only one process running per processor.

Scheduling algorithms fall into two categories: preemptive and non-preemptive algorithms. Preemptive is when an algorithm, such as Round-Robin, Shortest Remaining Time (SRT), removes a process from the processor when its associated time-slice (quantum) expires, to enable this the next process on the queue is given a higher priority or the waiting-time for other processes is increased. Preemptive reduces overall waiting time and makes such algorithms generally suitable for interactive systems. On the other hand, non-preemptive schedulers, such as First-in-First-Out (FIFO), Shortest Process First (SPF) and Highest-Response-Ratio-Next (HRRN), allow a process to run to its completion or until it voluntarily gives its CPU time. Non-preemptive can result in depriving waiting processes from the processor time in favor of the running process.

[-pagebreak-]

This is clear when a process has a high priority and a longer time-slice. FIFO and Round Robin represent the basic algorithms, their functionality aims at moving the processes in a simply path; placing processes as they come at the end of the CPU queue. While FIFO simply adds them up, Round Robin preempts processes to the end of the queue if they exceeded their time-slice. Some mechanism tend to favor processes that are set with a short turn around, this is evident in SPF and HRRN, or have little processing time left as in SPT. This generally makes the CPU more efficient as it services more, and speeds up the system performance. On the down side, longer processes can suffer from an unjust long waiting time. To resolve this, HRRN has a priority boost for those processes that waited a longer time, resulting in an increased throughput. To simulate a healthy system, the waiting-times for all processes must be tuned, along with an adequate time-slice and a priority boost, IO and CPU processes can run utilizing both the processor and IO devices. In the mean time, they all get their fair share in using the processor. In real time scheduling, time limits are set to force processes to finish on time. In critical systems usually a high response rate is more important than efficient resource utilization.

In conclusion, system developers need to find a way to manage the algorithms overhead and complexity in favor of gaining efficiency in resource allocation and management.

### References

Abraham Silberschatz, Peter Galvin, Greg Gagne. 2000. Applied Operating System Concepts. New York: John Wiley & Sons, Inc. Operating Systems (CS/CPE 408): CPU Scheduling  
[http://www.bridgeport.edu/sed/projects/cs503/Spring\\_2001/kode/os/scheduling.htm](http://www.bridgeport.edu/sed/projects/cs503/Spring_2001/kode/os/scheduling.htm) (1 May 2001). Pieter Dumon, The standard cpu scheduling algorithm, Tunes Organization, 14 Dec 1997, <http://tunes.org/~unios/std-sched.html> (1998)