

# **Chapter 4**

## **Processor Management**

### **At a Glance**

#### **Instructor's Manual Table of Contents**

- Overview
- Objectives
- Teaching Tips
- Quick Quizzes
- Class Discussion Topics
- Additional Projects
- Additional Resources
- Key Terms

## Lecture Notes

### Overview

This chapter explains how the Processor Manager manages and allocates a single central processing unit (CPU) to execute the incoming jobs. The chapter begins by presenting an overview of single-user systems and providing a foundation of terms and concepts used in processor management. Multi-core technologies are then briefly reviewed, followed by a discussion and comparison of job scheduling and process scheduling. The role of the Process Scheduler is explained in more detail, including a presentation of job and process status concepts, the process control block (PCB) data structure, and queuing. Process scheduling policies are presented next with an explanation of the advantages and disadvantages of preemptive versus nonpreemptive process scheduling algorithms. The goals of process scheduling policies are included. Six common process scheduling algorithms are explained in detail, followed by a description of four cases exemplifying how jobs move through the system. Finally, the chapter discusses the role of internal interrupts and the tasks performed by the interrupt handler.

### Learning Objectives

After completing this chapter, the student should be able to describe:

- The difference between job scheduling and process scheduling, and how they relate
- The advantages and disadvantages of process scheduling algorithms that are preemptive versus those that are nonpreemptive
- The goals of process scheduling policies in single-core CPUs
- Up to six different process scheduling algorithms
- The role of internal interrupts and the tasks performed by the interrupt handler

### Teaching Tips

#### Overview

1. Provide students with an introduction to processor management in single-user and multi-user (multiprogramming) systems. Explain the two states in a single-user system (busy and idle). Discuss the role of the CPU for both types of systems.
2. Make sure students understand the difference between a job and a process. Provide explanations of the terms *program or job*, *process*, *thread*, *processor*, *interrupt*, *higher priority*, and *context switch* using the examples on pages 110-111 in the text. These terms and concepts provide a foundation of understanding for the remaining chapter contents.

## About Multi-Core Technologies

1. Note that the term processor and core are considered equivalent and the processor(s) is located on a single silicon chip. Provide students with an introduction to multi-core technologies. Define multi-core (two or more processors) and explain what dual-core and quad-core mean.
2. Explain how multi-core engineering surfaced due to issues with nano-sized transistors and their ultra-close-placement on a computer chip. Be sure students understand that multi-core engineering helps resolve the current leakage and heat problems, in addition to allowing for the opportunity to permit multiple calculations simultaneously.
3. Note that for the Processor Manager, multi-core processing is more complex to manage than single-core processing. Multi-core processing will be covered in more detail in Chapter 6. This chapter focuses on single-core processing.

<b>Teaching Tip</b>	For a brief article providing an introduction to the multi-core concept and a discussion on factors to consider when determining how to take advantage of multi-core technology, refer your students to the following article: <a href="http://www.dell.com/downloads/global/power/ps2q05-20050103-Fruehe.pdf">www.dell.com/downloads/global/power/ps2q05-20050103-Fruehe.pdf</a>
---------------------	--

## Job Scheduling Versus Process Scheduling

1. Outline the two Processor Manager submanagers: the Job Scheduler and the Process Scheduler. Note the hierarchy between them and provide a brief overview of each.
2. Point out that the Job Scheduler initiates each job based on certain criteria.
3. Outline the most important functions of the Job Scheduler.
4. Be sure to mention the goals of the Job Scheduler in terms of striving for efficient use of system resources and keeping most components of the computer system busy most of the time.

## Process Scheduler

1. Point out that once the Job Scheduler has placed a job on the READY queue, the Process Scheduler takes over. The Process Scheduler is the lower-level scheduler in the submanager hierarchy. Outline important functions of the Process Scheduler, such as:
  - a. Determining which jobs will get the CPU, when, and for how long
  - b. Deciding when processing should be interrupted
  - c. Determining which queues the job should be moved to during its execution
  - d. Recognizing when a job has concluded and should be terminated

2. Discuss common programming traits. Most programs alternate between CPU and I/O cycles, and the frequency and CPU cycle often vary. Explain that these traits lead to general tendencies of I/O-bound jobs and CPU-bound jobs in terms of CPU cycles and I/O cycles.
3. Explain, using Figure 4.1 on page 114 of the text, that the total effect of all CPU cycles, from both I/O-bound and CPU-bound jobs, approximates a Poisson distribution curve. That is, a greater number of jobs request short CPU cycles (the frequency peaks close to the low end of the CPU cycle axis), and fewer jobs request long CPU cycles.
4. Point out that a middle-level scheduler (third layer) is used in a highly interactive environment. It handles system job overloading by removing active jobs from memory to reduce the degree of multiprogramming and allows jobs to be completed faster. This in turn results in faster job completion overall.
5. Note that in a single-user environment, there is no distinction between job and process scheduling because only one job is active at a time. Therefore, it receives dedicated system resources for the duration of the job.

### **Job and Process Status**

1. Outline the different states that a job takes as it moves through the system (HOLD, READY, WAITING, RUNNING, FINISHED). Note that these states are also referred to as job status or process status.
2. Explain, using Figure 4.2 on page 114 of the text, how the job status changes when a user submits a job to the system via batch or interactive mode.
3. Use the bullet points on page 115 in the text to describe every possible transition from one job or process state to another that is initiated by either the Job Scheduler or the Process Scheduler.

### **Process Control Blocks**

1. Provide students with an overview of Process Control Blocks (PCB), which contain all of the data about the job needed by the operating system to manage the processing of the job. Job data includes the following: what it is; where it is going; how much of its processing has been completed; where it is stored; and how much it has spent in using resources.
2. Describe briefly the contents of the PCB, such as process identification, process status, process state, and accounting.
3. Use Figure 4.3 on page 116 of the text as visual reinforcement of the PCB contents.

## PCBs and Queuing

1. Discuss the queuing process. Mention that queues use PCBs to track jobs. Be sure to emphasize that the PCBs are linked to form queues, not the jobs. Queues must be managed by process scheduling policies and algorithms.
2. Use Figure 4.4 on page 118 in the text to show queuing paths from HOLD to FINISHED. Point out that the Job and Processor Schedulers release resources when the job leaves the RUNNING state.

## Process Scheduling Policies

1. Briefly describe the multiprogramming environment. In this case, there are more jobs to be executed than could possibly run at one time. Outline the three limitations of a system that the operating system must resolve before scheduling all jobs in a multiprogramming environment. As stated in the text on page 118, they are: (1) there are a finite number of resources (such as disk drives, printers, and tape drives); (2) some resources, once they're allocated, can't be shared with another job (such as printers); and (3) some resources require operator intervention; that is, they can't be reassigned automatically from job to job (such as tape drives).
2. Discuss different criteria of a good process scheduling policy. A policy should maximize throughput, minimize response time, minimize turnaround time, minimize waiting time, maximize CPU efficiency, and ensure fairness for all jobs. Notice that some of these criteria contradict each other. Point out that the final decision rests with the system designer, who must determine which criteria are most important for that specific system.
3. Introduce the concepts of a timing mechanism and interrupts. These items are used by a processor in order to remove unacceptable imbalances in the READY and I/O queues.
4. Explain the preemptive scheduling policy and nonpreemptive scheduling policy. Point out that a preemptive scheduling policy is a strategy that interrupts the processing of a job and transfers the CPU to another job. It is widely used in time-sharing environments. The nonpreemptive scheduling policy functions without external interrupts.

<b>Teaching Tip</b>	Refer to the following Web site to learn more about process scheduling policies: <a href="http://docs.hp.com/en/5965-4642/ch01s13.html">http://docs.hp.com/en/5965-4642/ch01s13.html</a>
---------------------	---

<b>Teaching Tip</b>	To learn more about the optimality of a scheduling policy for processing a job stream, download the article from the following link: <a href="http://portal.acm.org/citation.cfm?id=2080.357395">http://portal.acm.org/citation.cfm?id=2080.357395</a>
---------------------	---

## **Quick Quiz 1**

1. Which of the following are the functions of a Process Scheduler? (Choose all that apply.)
  - a. Initiates the job based on certain criteria
  - b. Decides when processing should be interrupted
  - c. Puts jobs in a sequence such that they use all system resources as fully as possible
  - d. Determines which queues the job should be moved to during its execution

Answer: b and d

2. The total effect of all CPU cycles, from both I/O-bound and CPU-bound jobs, approximates which of the following distribution curves?
  - a. Gaussian distribution
  - b. Poisson distribution
  - c. Lorentzian Distribution
  - d. Random Distribution

Answer: b

3. The \_\_\_\_\_ contains all of the data about the job needed by the operating system to manage the processing of the job.

Answer: Process Control Block

## **Process Scheduling Algorithms**

1. Remind students that a Process Scheduler allocates the CPU and moves jobs through the system. List the six different process scheduling algorithms. Point out that the early systems used nonpreemptive policies to move batch jobs. Most current systems, with their emphasis on interactive use and response time, use preemptive policies that take care of the immediate requests of interactive users.

### **First-Come, First-Served**

1. Explain the first-come, first-served (FCFS) algorithm. Point out that it is nonpreemptive and handles jobs according to their arrival time. Outline FCFS advantages and disadvantages. For example, it is simple to implement and good for batch systems; however, turnaround time is unpredictable.
2. Use the examples shown in Figure 4.5 and Figure 4.6 on page 121 of the text to demonstrate how this algorithm works.

### **Shortest Job Next**

1. Explain the shortest job next (SJN) algorithm. Point out that it is nonpreemptive and handles jobs based on the length of their CPU cycle time.

2. Outline SJN advantages and disadvantages. For example, it is simple to implement in batch systems; however, it does not work in interactive systems. Also, it is optimal only when all jobs are available at the same time and the CPU estimates are available and accurate.
3. Use the example shown in Figure 4.7 on page 122 of the text to demonstrate how this algorithm works.

### **Priority Scheduling**

1. Explain the priority scheduling algorithm. Point out that it is nonpreemptive and gives preferential treatment to important jobs. That is, programs with the highest priority are processed first and they are not interrupted until CPU cycles are completed or a natural wait occurs.
2. Point out that a FCFS policy is used if two or more jobs have equal priority in the READY queue.
3. Note that the system administrator or the process manager may use different methods for assigning priorities.
4. Outline different criteria the Processor Manager uses to determine priorities, such as memory requirements, number and type of peripheral devices, total CPU time, and amount of time already spent in the system.

### **Shortest Remaining Time**

1. Explain the shortest remaining time (SRT) algorithm. Point out that it is a preemptive version of the SJN algorithm, and the processor is allocated to jobs that are closest to completion.
2. Outline SRT advantages and disadvantages. For example, it ensures fast completion of short jobs; however, it involves more overhead than SJN due to context switching. Also, this algorithm cannot be implemented in an interactive system because it requires advance knowledge of the CPU time required to finish each job.
3. Use the examples shown in Figure 4.8 and Figure 4.9 on page 125 to demonstrate how this algorithm works. Also discuss the situation in which the turnaround time is greater in SJN than in SRT. Note that a precise comparison of SRT and SJN would have to include the time required to do the context switching in the SRT algorithm.

### **Round Robin**

1. Explain the round robin algorithm. Point out that it is preemptive, used extensively in interactive systems, and based on a predetermined slice of time called a time quantum. Mention that this algorithm ensures that the CPU is equally shared among all active processes so that the CPU is not monopolized by any one job.

2. Discuss job processing within the round robin algorithm and explain the different situations possible, such as when the job's CPU cycle is shorter or longer than the time quantum.
3. Use the example shown in Figure 4.10 on page 127 of the text to demonstrate how this algorithm works.
4. Explain the effect of the size of the time quantum on efficiency, in relation to the average CPU cycle. Use the example shown in Figure 4.11 on page 128 of the text to show how this works.
5. Outline the two general rules of thumb for selecting the proper time quantum. First, it should be long enough to allow 80 percent of CPU cycles to run to completion. Second, it should be at least 100 times longer than the time required to perform one context switch. Point out that these rules are flexible and depend on the particular system.

### **Multiple-Level Queues**

1. Provide a brief overview of multiple-level queues. Point out that this concept works in conjunction with several other schemes. It is found in systems where jobs can be grouped according to a common characteristic.
2. Provide examples of systems that employ multiple-level queues. One example is that of a priority-based system with different queues for each priority level. A second example is when another kind of system might gather all of the CPU-bound jobs in one queue and all I/O-bound jobs in another. The Process Scheduler then alternately selects jobs from each queue to keep the system balanced. A third example is a hybrid environment that supports both batch and interactive jobs.
3. Briefly mention the four primary methods to handle the movement of a job before moving to a detailed discussion of each case.

### **Case 1: No Movement Between Queues**

1. Discuss the pros and cons of this job movement policy. For example, it is quite simple and rewards high-priority jobs. The processor is allocated to the jobs in the high-priority queue using the first-come, first-serve algorithm. Low-priority jobs get the processor only when the high-priority queues are empty.
2. Note that this policy can be justified if there are relatively few users with high-priority jobs, so the top queues quickly empty out, allowing the processor to spend a fair amount of time running the low-priority jobs.

### Case 2: Movement Between Queues

1. Discuss the pros and cons of this job movement policy between queues. For example, the processor adjusts priorities assigned to each job. High-priority jobs are initially given a favorable assignment; however, they are later adjusted to be like all other jobs. When a time quantum interrupt occurs, the job is preempted and moved to the end of the next lower queue. A job may also have its priority increased; for example, when it issues an I/O request before its time quantum has expired.
2. Note that this policy is considered quite fair in a system in which the jobs are handled according to their computing cycle characteristics: CPU-bound or I/O-bound.

### Case 3: Variable Time Quantum Per Queue

1. Note that this case represents a variation of Case 2 above (the movement between queues policy), and it allows for faster turnaround of CPU-bound jobs. Use the example in the text on page 130, in which each of the queues is given a time quantum twice as long as the previous queue. As discussed in the text, the highest queue might have a time quantum of 100 milliseconds. Therefore, the second-highest queue would have a time quantum of 200 milliseconds; the third would have 400 milliseconds, and so on. If there are enough queues, the lowest one might have a relatively long time quantum of three seconds or more.
2. The advantage of this policy is that a CPU-bound job executes longer and is provided longer time periods in which to run. Thus, this policy improves a CPU-bound job's chance of finishing faster.

### Case 4: Aging

1. Aging is used to ensure that jobs in the lower-level queues will eventually complete their execution. The system keeps track of a job's wait time, and if a job becomes too "old" waiting for the CPU, the system moves the job to the next highest queue. This movement continues until the "old" job reaches the top available queue. In some severe cases, the job may move drastically to the highest queue for processing.
2. The advantage of this job movement policy is that it guards against the problem of indefinite postponement of unwieldy jobs. As discussed in the text on page 131, indefinite postponement means that a job's execution is delayed for an undefined amount of time because it is repeatedly preempted so other jobs can be processed. Eventually the situation could lead to the old job's starvation. Indefinite postponement is a major problem. This problem of indefinite postponement is discussed further in Chapter 5.

<b>Teaching Tip</b>	Various articles related to process scheduling algorithms can be found by entering <i>process scheduling algorithm</i> in the search toolbar at the following Web site: <a href="http://portal.acm.org">http://portal.acm.org</a>
---------------------	--

## A Word About Interrupts

1. Briefly discuss various types of interrupts, such as page interrupts to accommodate job requests in memory management, time quantum expiration interrupts, I/O interrupts, internal interrupts (synchronous interrupts), illegal arithmetic operations (attempts to divide by zero), and illegal job instructions.
2. Briefly explain the interrupt handler, which is used to control the program that handles the interruption sequence of events. Outline its functions and sequence of steps when addressing a nonrecoverable error.

**Teaching  
Tip**

More on the topic of context switching may be found at the following Web site:  
[www.linfo.org/context\\_switch.html](http://www.linfo.org/context_switch.html)

## Quick Quiz 2

1. Which of the following are preemptive scheduling algorithms? (Choose all that apply.)
  - a. FCFS
  - b. SRT
  - c. SJN
  - d. Round robin

Answer: b and d

2. Which of the following algorithms employ the concept of a time quantum that is given to each job to ensure that the CPU is equally shared among all active processes?
  - a. SRT
  - b. SJN
  - c. Round robin
  - d. Priority scheduling

Answer: c

3. The control program that handles the interruption sequence of events is called the \_\_\_\_\_.

Answer: interrupt handler

## Class Discussion Topics

1. Have students discuss the pros and cons of the various process management algorithms presented in this chapter. Given a choice, which algorithm would they prefer to implement and why?

2. Have students discuss the different criteria for designing a process scheduling policy. Do they think that it is possible to optimize the design of a process scheduling policy to suit any system? If so, how? If not, why? Ask them to provide some examples to corroborate their ideas.

## Additional Projects

1. Have students compile a list of the important characteristics of processors commonly employed in today's personal computers.
2. Have students research how to write the algorithms for shortest job next (SJN) and shortest remaining time (SRT) policies.

## Additional Resources

1. Microsoft.com:  
[www.microsoft.com](http://www.microsoft.com)
2. Osddata.com:  
[www.osdata.com/kind/history.htm](http://www.osdata.com/kind/history.htm)
3. Intel® Multi-Core:  
[www.intel.com/multi-core/](http://www.intel.com/multi-core/)
4. AMD® Quad-Core Resource Center:  
<http://multicore.amd.com/us-en/AMD-Multi-Core/Resources.aspx>
5. Multi-core Enterprise Technology at Dell®:  
[www1.euro.dell.com/content/topics/topic.aspx/emea/topics/products/pedge/en/multi\\_core?c=uk&cs=ukbsdt1&l=en&s=bsd](http://www1.euro.dell.com/content/topics/topic.aspx/emea/topics/products/pedge/en/multi_core?c=uk&cs=ukbsdt1&l=en&s=bsd)
6. Freescale's Multi-core Platform:  
[www.freescale.com/webapp/sps/site/overview.jsp?nodeId=0162468rH3bTdG25E4&tid=NEV20070625MULTIC](http://www.freescale.com/webapp/sps/site/overview.jsp?nodeId=0162468rH3bTdG25E4&tid=NEV20070625MULTIC)

## Key Terms

- **Aging:** a policy used to ensure that jobs that have been in the system for a long time in the lower-level queues will eventually complete their execution.
- **Context switching:** the acts of saving a job's processing information in its PCB so the job can be swapped out of memory, and of loading the processing information from the PCB of another job into the appropriate registers so the CPU can process it. Context switching occurs in all preemptive policies.
- **CPU-bound:** a job that will perform a great deal of nonstop processing before issuing an interrupt.

- **First-come, first-served (FCFS):** a nonpreemptive process scheduling policy (or algorithm) that handles jobs according to their arrival time.
- **High-level scheduler:** a synonym for the Job Scheduler.
- **I/O-bound:** a job that requires a large number of input/output operations, resulting in much free time for the CPU.
- **Indefinite postponement:** signifies that a job's execution is delayed indefinitely because it is repeatedly preempted so other jobs can be processed.
- **Interrupt:** a hardware signal that suspends execution of a program and activates the execution of a special program known as the interrupt handler.
- **Interrupt handler:** the program that controls what action should be taken by the operating system when a sequence of events is interrupted.
- **Job Scheduler:** the high-level scheduler of the Processor Manager that selects jobs from a queue of incoming jobs based on each job's characteristics.
- **Job status:** the condition of a job as it moves through the system from the beginning to the end of its execution.
- **Low-level scheduler:** a synonym for the Process Scheduler.
- **Middle-level scheduler:** a scheduler used by the Processor Manager to manage processes that have been interrupted because they have exceeded their allocated CPU time slice.
- **Multiple-level queues:** a process scheduling scheme (used with other scheduling algorithms) that groups jobs according to a common characteristic.
- **Multiprogramming:** a technique that allows a single processor to process several programs residing simultaneously in main memory and interleaving their execution by overlapping I/O requests with CPU requests.
- **Natural wait:** a common term used to identify an I/O request from a program in a multiprogramming environment that would cause a process to wait "naturally" before resuming execution.
- **Nonpreemptive scheduling policy:** a job scheduling strategy that functions without external interrupts so that once a job captures the processor and begins execution, it remains in the running state uninterrupted until it issues an I/O request or it's finished.
- **Preemptive scheduling policy:** any process scheduling strategy that, based on predetermined policies, interrupts the processing of a job and transfers the CPU to another job. It is widely used in time-sharing environments.
- **Priority scheduling:** a nonpreemptive process scheduling policy (or algorithm) that allows for the execution of high-priority jobs before low-priority jobs.
- **Process:** an instance of execution of a program that is identifiable and controllable by the operating system.
- **Process Control Block (PCB):** a data structure that contains information about the current status and characteristics of a process.
- **Process Scheduler:** the low-level scheduler of the Processor Manager that establishes the order in which processes in the READY queue will be served by the CPU.
- **Process scheduling algorithm:** an algorithm used by the Job Scheduler to allocate the CPU and move jobs through the system.
- **Process scheduling policy:** any policy used by the Processor Manager to select the order in which incoming jobs will be executed.
- **Process status:** information stored in the job's PCB that indicates the current position of the job and the resources responsible for that status.
- **Processor:** (1) a synonym for the CPU, or (2) any component in a computing system capable of performing a sequence of activities.

- **Program:** an interactive unit, such as a file stored on a disk.
- **Queue:** a linked list of PCBs that indicates the order in which jobs or processes will be serviced.
- **Response time:** a measure of the efficiency of an interactive system that tracks the speed with which the system will respond to a user's command.
- **Round robin:** a preemptive process scheduling policy (or algorithm) that allocates to each job one unit of processing time per turn to ensure that the CPU is equally shared among all active processes and isn't monopolized by any one job.
- **Shortest job next (SJN):** a nonpreemptive process scheduling policy (or algorithm) that selects the waiting job with the shortest CPU cycle time.
- **Shortest remaining time (SRT):** a preemptive process scheduling policy (or algorithm) similar to the SJN algorithm that allocates the processor to the job closest to completion.
- **Task:** (1) the term used to describe a process, or (2) the basic unit of concurrent programming languages that defines a sequence of instructions that may be executed in parallel with other similar units.
- **Thread:** a portion of a program that can run independently of other portions. Multithreaded applications programs can have several threads running at one time with the same or different priorities.
- **Time quantum:** a period of time assigned to a process for execution before it is preempted.
- **Turnaround time:** a measure of a system's efficiency that tracks the time required to execute a job and return output to the user.