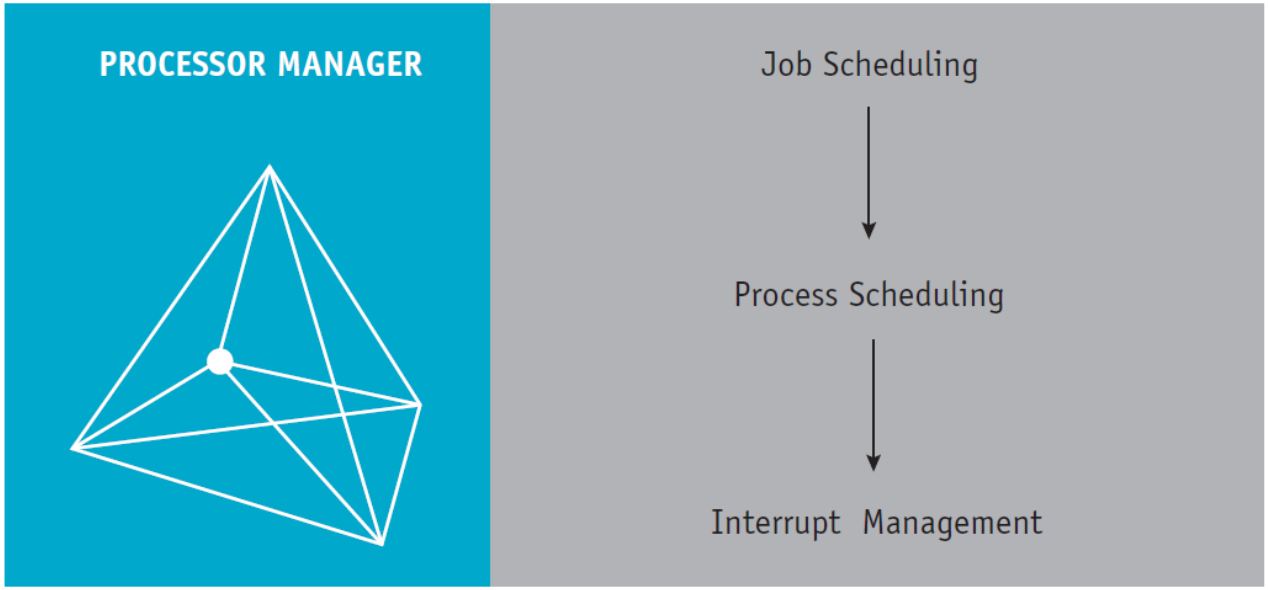
Operating Systems I

LECTURE VII: PROCESSOR MANAGEMENT (2/2)

Siv Ratha | siv.ratha89@gmail.com

Department of Information and Communication Engineering
Institute of Technology of Cambodia

Processor Management



"Nature acts by progress … It goes and returns, then advances further, then twice as much backward, then more forward than ever."

Learning Objectives

- Students will understand/know:
  - The relationship between job and process scheduling
  - The advantages/disadvantages of process scheduling algorithms
  - The goals of process scheduling policies using a single-core CPU
  - The similarities and differences between processes and threads
  - The role of internal interrupts and of the interrupt handler

Previously…

- Program/job, process/task, and thread
- Multithreading
- Multiprogramming
- Multi-Core technologies
- Scheduling submanagers (thread and process schedulers)
- Job, process, and thread states
- Control block (TCB and PCB)
- Queuing and queuing paths
- Scheduling policy

What's Next?

- Scheduling Algorithms
  - First-Come, First-Served (FCFS)
  - Shortest Job Next
  - Priority Scheduling
  - Shortest Remaining Time
  - Round Robin
  - Multiple-Level Queues
  - Earliest Deadline First
- Managing Interrupts

Scheduling Algorithms

- The Process Scheduler relies on a scheduling algorithm, based on a specific scheduling policy, to allocate the CPU in the best way to move jobs through the system efficiently.
- Most systems place an emphasis on fast user response time.
- To keep this discussion simple, all the algorithms we will see are referred as process scheduling algorithms, though they are also used to schedule threads.

FCFS – First-Come, First-Served

- FCFS is a nonpreemptive scheduling algorithm that handles all incoming objects according to their arrival time: the earlier they arrive, the sooner they're served.
- This algorithm is similar with FIFO, and it is fine for most batch systems, but it is unacceptable for interactive systems because interactive users expect quick response times.
- With FCFS, as a new job enters the system, its PCB is linked to the end of the READY queue and it is removed from the front of the READY queue after the jobs before it runs to completion and the processor becomes available.
- Turnaround time (the time required to execute a job and return the output to the user) is unpredictable.

FCFS – Example

- Consider the following three jobs (time in *ms*):

Case 1: A, B, C

$$\text{Average turnaround time} = \frac{(15 - 0) + (17 - 0) + (18 - 0)}{3} = 16.67$$

Case 2: C, B, A

$$\text{Average turnaround time} = \frac{(1 - 0) + (3 - 0) + (18 - 0)}{3} = 7.3$$

SJN – Shortest Job Next

- SJN is a nonpreemptive scheduling algorithm (also known as shortest job first, or SJF) that handles jobs based on the length of their CPU cycle time.
- It is possible to implement in batch environments where the estimated CPU time required to run the job is given in advance by each user at the start of each job.
- It does not work in most interactive systems because users do not estimate in advance the CPU time required to run their jobs.
- However, the SJN algorithm is optimal only when all of the jobs are available at the same time, and the CPU estimates must be available and accurate.

SJN – Example

- Job : A B C D
- CPU cycle : 5 2 6 4

Average turnaround time = $\frac{(2-0)+(6-0)+(11-0)+(17-0)}{4} = 9.0$

Priority Scheduling

- It is one of the most common scheduling algorithms for batch systems and is a nonpreemptive algorithm (in a batch environment).
- It gives preferential treatment to important jobs.
- It allows the programs with the highest priority to be processed first, and they aren't interrupted until their CPU cycles (run times) are completed or a natural wait occurs.
- If two or more jobs with equal priority are present in the READY queue, the processor is allocated to the one that arrived first (first-come, first-served within priority).

Priority Determination

- Priorities can be assigned by a system administrator using characteristics extrinsic to the jobs.
- Priorities can also be determined by the Processor Manager based on characteristics intrinsic to the jobs such as:
  - **Memory requirements**: Jobs requiring large amounts of memory could be allocated lower priorities or vice versa.
  - **Number and type of peripheral devices**: Jobs requiring many peripheral devices would be allocated lower priorities.
  - **Total CPU time**: Jobs having a long CPU cycle, or estimated run time, would be given lower priorities.
  - **Amount of time already spent in the system**: This is the total amount of elapsed time since the job was accepted for processing.

SRT – Shortest Remaining Time

- SRT is a preemptive version of the SJN algorithm.
- SRT involves more overhead than SJN.
- The processor is allocated to the job closest to completion, but even this job can be interrupted if a newer job in the READY queue has a time to completion that is shorter.
- This algorithm can't be implemented in an interactive system because it requires advance knowledge of the CPU time required to finish each job, but It can work well in batch environments, because it can give preference to short jobs.

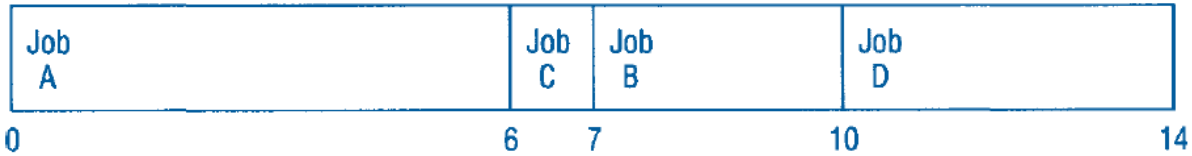
SRT – Example

- Arrival : 0 1 2 3
- Job : A B C D
- CPU cycle : 6 3 1 4

SRT

$$\text{Average turnaround time} = \frac{(14 - 0) + (5 - 1) + (3 - 2) + (9 - 3)}{4} = 6.25$$

SJN



$$\text{Average turnaround time} = \frac{(6 - 0) + (7 - 2) + (10 - 1) + (14 - 3)}{3} = 7.75$$

Round Robin

- Round Robin is a preemptive process scheduling algorithm that is used extensively in interactive systems.
- It is easy to implement, which is not based on job characteristics but on a predetermined slice of time that's given to each job to ensure that the CPU is equally shared among all active processes and is not monopolized by any one job.
- This time slice is called a time quantum; its size is crucial to the performance of the system, which can vary from 100 milliseconds to 1 or 2 seconds.

Round Robin – How It Works

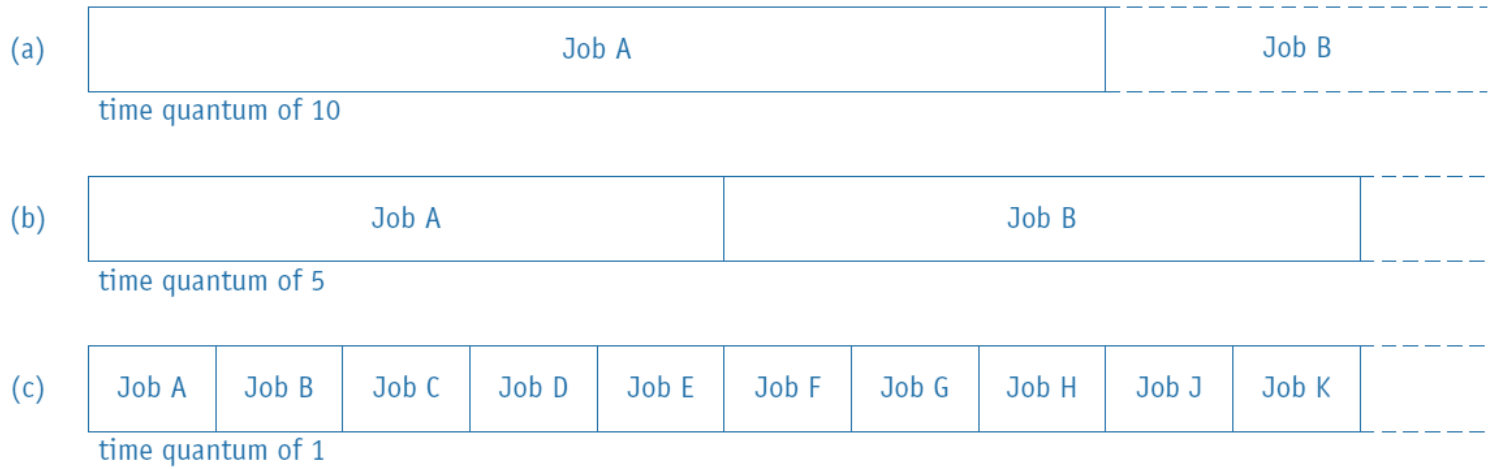
- Jobs are placed in the READY queue using a first-come, first-served scheme.
- The Process Scheduler selects the first job from the front of the queue, sets the timer to the time quantum, and allocates the CPU to this job.
- If processing isn't finished when time expires, the job is preempted and put at the end of the READY queue, and its information is saved in its PCB.

Round Robin – Example

- Arrival : 0 1 2 3
- Job : A B C D
- CPU cycle : 8 4 9 5
- Time quantum : 4

Average turnaround time = $\frac{(20-0)+(8-1)+(26-2)+(25-3)}{3} = 18.25$

Round Robin – Study Cases



- The efficiency of Round Robin depends on the size of the time quantum in relation to the average CPU cycle.
- Two general rules of thumb for selecting the time quantum:
  - It should be long enough to allow 80 percent of the CPU cycles to run to completion.
  - And, it should be at least 100 times longer than the time required to perform one context switch.

Multiple-Level Queues

- Multiple-level queues isn't really a separate scheduling algorithm, but works in conjunction with several of the schemes already discussed and is found in systems with jobs that can be grouped according to a common characteristic.
- The scheduling policy is based on some predetermined scheme that allocates special treatment to the jobs in each queue.
- The system designers can choose to use different algorithms for different queues, allowing them to combine the advantages of several algorithms.

Multi-Level Que. – Examples

- The Priority-based system using different queues for each priority level
- The system with all CPU-bound jobs in one queue and all I/O-bound jobs in another
- The hybrid system with batch jobs in background queue and interactive jobs in a foreground queue

Multi-Level Que. – Cases

- Case 1 – No Movement Between Queues
- Case 2 – Movement Between Queues
- Case 3 – Variable Time Quantum Per Queue
- Case 4 – Aging

Multi-Level Que. – Case 1

- Case 1 – No Movement Between Queues:
  - The processor is allocated to the jobs in the high-priority queue in FCFS fashion.
  - It is allocated to jobs in low-priority queues only when the high priority queues are empty.

Multi-Level Que. – Case 2

- Case 2 – Movement Between Queues:
  - High-priority jobs are treated like all the others once they are in the system.
  - 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, such as when it issues an I/O request before its time quantum has expired.

Multi-Level Que. – Case 3

- Case 3 – Variable Time Quantum Per Queue:
  - Variable time quantum per queue is a variation of the movement between queues policy.
  - It allows for faster turnaround of CPU-bound jobs.
  - Each of the queues is given a time quantum twice as long as the previous queue.
  - CPU-bound job can execute for longer and longer periods of time, thus improving its chances of finishing faster.

Multi-Level Que. – Case 4

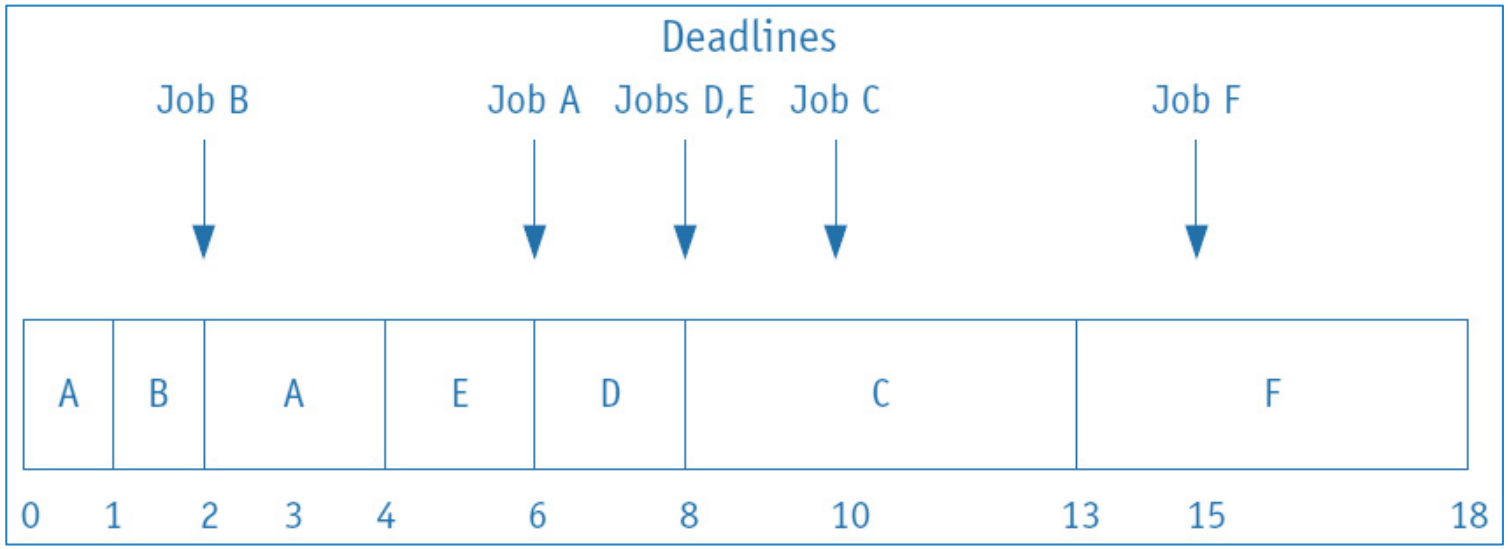
- Case 4 – Aging:
  - It is used to ensure that jobs in the lower-level queues will eventually complete their execution.
  - When a job gets too old—that is, when it reaches a certain time limit—the system moves the job to the next highest queue, and so on, until it reaches the top queue.
  - It guards against the indefinite postponement of unwieldy jobs.

EDF – Earliest Deadline First

- It is known as a preemptive dynamic priority algorithm.
- It is built to address the critical processing requirements of real-time systems and their pressing deadlines.
- The priority can be adjusted as the job moves through execution from START to FINISHED.
- The primary goal of EDF is to process all jobs in the order that is most likely to allow each to run to completion before reaching their respective deadlines.
- The closer the deadline, the higher the priority of a job will be.
- In case of the same deadline, FCFS is used.

EDF – Example

Job:	A	B	C	D	E	F
Arrival time:	0	1	2	3	3	5
Execution Time:	3	1	5	2	2	5
Deadline:	6	2	10	8	8	15
Time-before-deadline (at arrival time)	6	1	8	5	5	10



Comparison (Page 131)

Algorithm	Policy Type	Disadvantages	Advantages
First Come, First Served	Nonpreemptive	Unpredictable turnaround times; has an element of chance	Easy to implement
Shortest Job Next	Nonpreemptive	Indefinite postponement of some jobs; requires execution times in advance	Minimizes average waiting time
Priority Scheduling	Nonpreemptive	Indefinite postponement of some jobs	Ensures fast completion of important jobs
Shortest Remaining Time	Preemptive	Overhead incurred by context switching	Ensures fast completion of short jobs
Round Robin	Preemptive	Requires selection of good time quantum	Provides reasonable response times to interactive users; provides fair CPU allocation
Multiple-Level Queues	Preemptive/ Nonpreemptive	Overhead incurred by monitoring queues	Flexible scheme; allows aging or other queue movement to counteract indefinite postponement; is fair to CPU-bound jobs
Earliest Deadline First	Preemptive	Overhead required to monitor dynamic deadlines	Attempts timely completion of jobs

Managing Interrupts

- There are many types of interrupt:
  - Page interrupts to accommodate job requests
  - I/O interrupts when a READ or WRITE command is issued
  - Internal interrupts, or synchronous interrupts, occurring as a direct result of the arithmetic operation or other instruction currently being processed.
  - Interrupts by illegal arithmetic operations such as exception, overflow and underflow.
  - Illegal job instructions such as accessing protected or nonexistent storage locations, using undefined operation code, operating invalid data, and other unauthorized attempting, etc.
- The control program that handles the interruption sequence of events is called the **interrupt handler**.

Sequence of Interrupt Handler

- When the operating system detects an error that is not recoverable, the interrupt handler typically follows this sequence:
  - The type of interrupt is described and stored—to be passed on to the user as an error message.
  - The state of the interrupted process is saved, including the value of the program counter, the mode specification, and the contents of all registers.
  - The interrupt is processed: The error message and the state of the interrupted process are sent to the user; program execution is halted; any resources allocated to the job are released; and the job exits the system.
  - The processor resumes normal operation.

Homework

1. Exercise 11 on page 136 of ref. book.