

CPU Scheduling

- Basic Concepts
- Scheduling Criteria
- Scheduling Algorithms
 - Batch systems
 - Interactive systems

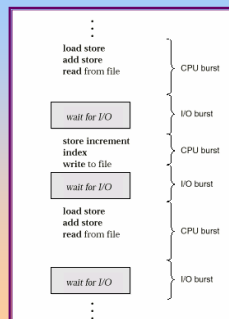
Based on original slides by
Silberschatz, Galvin and Gagne

CPU Scheduling

1

Basic Concepts

- CPU-I/O Burst Cycle
 - Process execution consists of a cycle of CPU execution and I/O wait.
 - CPU-Bound Processes
 - I/O-Bound Processes



CPU Scheduling

2

Basic Concepts (2)

- Maximum CPU utilization obtained with multiprogramming
 - Different part of the systems can be active simultaneously allowing parallel execution of processes
 - The scheduling algorithm should mix appropriately CPU-bound and I/O-Bound Processes

CPU Scheduling

3

CPU Scheduler

- Selects from among the processes in memory that are ready to execute, and allocates the CPU to one of them.
- CPU scheduling decisions may take place in different situations
 - Non-preemptive scheduling
 - The running process terminates
 - The running process performs an I/O operation or waits for an event
 - Preemptive scheduling
 - The running process has exhausted its time slice
 - A process A transits from blocked to ready and is considered more important than process B that is currently running
 - ...

CPU Scheduling

4

Dispatcher

- Dispatcher module gives control of the CPU to the process selected by the short-term scheduler; this involves:
 - Context Switch
 - Switching to user mode
 - Jumping to the proper location in the user program to restart that program
- *Dispatch latency*
 - time it takes for the dispatcher to stop one process and start another running.
- Context-switches should be minimized

CPU Scheduling

5

Type of scheduling

- Batch Systems
 - Maximize the resource utilization
- Interactive Systems
 - Minimize response times
- Real-Time Systems
 - Meet Temporal Constraints

CPU Scheduling

6

Objectives

- General
 - ✦ Fairness
 - ✦ Load Balancing
- Batch Systems
 - ✦ CPU utilization (% of time the CPU is executing processes)
 - ✦ Throughput (# of processes executed per time unit)
 - ✦ Turnaround time (amount of time to execute a particular process)
- Interactive Systems
 - ✦ Response time
 - ☞ amount of time it takes from when a request was submitted until the first response is produced, **not** output
- Real-Time Systems
 - ✦ Temporal Constraints
 - ☞ Avoid data loss
 - ☞ Avoid Quality of Service (QoS) degradation

CPU Scheduling 7

- | | |
|----------------|---|
| CPU Scheduling | 7 |
|----------------|---|

Scheduling for Batch Systems

- FCFS
 - ✎ First-Come First-Served
- SJF
 - ✎ Shortest Job First
- SRJF
 - ✎ Shortest Remaining Job First

CPU Scheduling 8

- | | |
|----------------|---|
| CPU Scheduling | 8 |
|----------------|---|

First-Come, First-Served (FCFS)

<u>Process</u>	<u>Burst Time</u>
P_1	24
P_2	3
P_3	3

- Suppose that the processes arrive in the order: P_1, P_2, P_3

P_1	P_2	P_3
0	24	27 30

- Waiting time for $P_1 = 0$; $P_2 = 24$; $P_3 = 27$
- Average waiting time: $(0 + 24 + 27)/3 = 17$

CPU Scheduling

CPU Scheduling	9
----------------	---

- | | |
|----------------|---|
| CPU Scheduling | 9 |
|----------------|---|

FCFS (Cont.)

Suppose that the processes arrive in the order

P_2, P_3, P_1 .



- Waiting time for $P_1 = 6$; $P_2 = 0$; $P_3 = 3$
- Average waiting time: $(6 + 0 + 3)/3 = 3$
- Much better than previous case.
- *Convoy effect* short process behind long process

CPU Scheduling

10

Shortest-Job-First (SJF)

- Associate with each process the length of its next CPU burst. Use these lengths to schedule the process with the shortest time.
- Two schemes:
 - Non-preemptive
 - once CPU given to the process it cannot be preempted until completes its CPU burst.
 - Preemptive (Shortest-Remaining-Time-First or SRTF).
 - if a new process arrives with CPU burst length less than remaining time of current executing process, preempt. This scheme is known as the
- SJF is optimal
 - gives minimum average waiting time for a given set of processes that are simultaneously available.

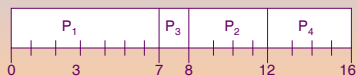
CPU Scheduling

11

Example of Non-Preemptive SJF

Process	Arrival Time	Burst Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

- SJF (non-preemptive)



- Average waiting time = $(0 + 6 + 3 + 7)/4 = 4$

CPU Scheduling

12

Example of Preemptive SJF (SRJF)

Process	Arrival Time	Burst Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

■ SJF (preemptive)



■ Average waiting time = $(9 + 1 + 0 + 2)/4 = 3$

CPU Scheduling

13

Approximate SJF

- Shortest Process Next (when applied to Interactive Systems)
- Requires estimates of the next execution length
- Can be done by using the length of previous execution lengths, using exponential averaging
 1. t_n = actual length of n^{th} CPU burst
 2. τ_{n+1} = predicted value for the next CPU burst
 3. $\alpha, 0 \leq \alpha \leq 1$
 4. Define :

$$\tau_{n+1} = \alpha t_n + (1 - \alpha) \tau_n.$$

CPU Scheduling

14

Examples of Exponential Averaging

■ $\alpha = 0$

- ☞ $\tau_{n+1} = \tau_n$
- ☞ Recent history does not count.

■ $\alpha = 1$

- ☞ $\tau_{n+1} = t_n$
- ☞ Only the actual last execution counts.

■ If we expand the formula, we get:

$$\tau_{n+1} = \alpha t_n + (1 - \alpha) \alpha t_{n-1} + \dots + (1 - \alpha)^j \alpha t_{n-j} + \dots + (1 - \alpha)^n t_0$$

- Since both α and $(1 - \alpha)$ are less than or equal to 1, each successive term has less weight than its predecessor.

CPU Scheduling

15

Scheduling for Interactive Systems

- Round-Robin (RR)
- Priority-based
- Shortest Process Next
 - Approximated SJF
- Multi-level

CPU Scheduling16

Round Robin (RR)

- Each process gets a small unit of CPU time (*time quantum*).
 - After this time has elapsed, the process is preempted and added to the end of the ready queue.
- n processes in the ready queue; time quantum = q
 - Each process gets $1/n$ of the CPU time in chunks of at most q time units at once.
 - No process waits more than $(n-1)q$ time units.
- Performance
 - q large \Rightarrow FIFO
 - q small $\Rightarrow q$ must be large with respect to context switch, otherwise overhead is too high.

CPU Scheduling17

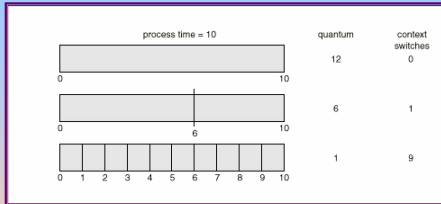
Example of RR with Time Quantum = 20

Process	Burst Time
P_1	53
P_2	17
P_3	68
P_4	24

P_1	P_2	P_3	P_4	P_1	P_3	P_4	P_1	P_3	P_3	
0	20	37	57	77	97	117	121	134	154	162

CPU Scheduling18

Time Quantum and Performance



- Time quantum impacts on
 - Number of Context Switches
 - Average Response Time
- RR is **fair** by definition
 - All processes are given the same chances

CPU Scheduling

19

Priority Scheduling

- A priority number (integer) is associated with each process
 - Static vs. Dynamic Priority
- The CPU is allocated to the process with the highest priority (smallest integer \equiv highest priority).
 - Preemptive vs. Non-preemptive
- SJF (SRJF) is a priority scheduling where priority is the predicted (remaining) CPU burst time.
- **Problem**
 - Starvation – low priority processes may never execute.
- **Solution**
 - Aging – as time progresses increase the priority of the process.
- Classes of priority

CPU Scheduling

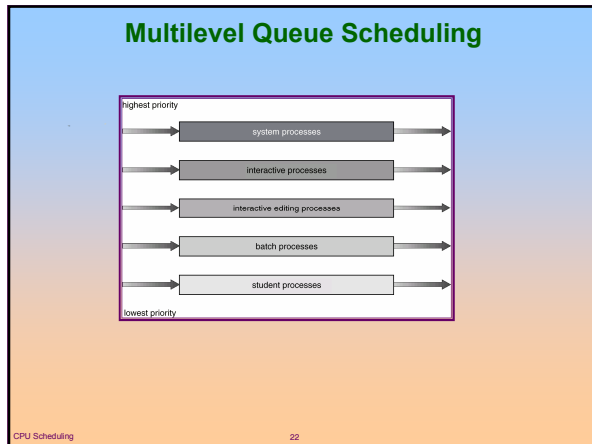
20

Multilevel Queue

- Ready queue is partitioned into separate queues:
 - foreground (interactive)
 - background (batch)
- Each queue has its own scheduling algorithm
 - foreground – RR
 - background – FCFS
- Scheduling must be done between the queues.
 - Fixed priority scheduling
 - i.e., serve all from foreground then from background
 - Possibility of starvation.
 - Time slice
 - each queue gets a certain amount of CPU time which it can schedule amongst its processes
 - Example: 80% to foreground in RR; 20% to background in FCFS

CPU Scheduling

21



Multilevel Feedback Queue

- A process can move between the various queues; aging can be implemented this way.
- Multilevel-feedback-queue scheduler defined by the following parameters:
 - number of queues
 - scheduling algorithms for each queue
 - method used to determine when to upgrade a process
 - method used to determine when to demote a process

CPU Scheduling 23

Example of Multilevel Feedback Queue

- Three queues:
 - Q_0 – time quantum 8 ms
 - Q_1 – time quantum 16 ms
 - Q_2 – FCFS
- Scheduling
 - A new job enters queue Q_0 which is served FCFS.
 - When it gains CPU, job receives 8 milliseconds.
 - If it does not finish in 8 milliseconds, job is moved to queue Q_1 .
 - At Q_1 job is again served FCFS and receives 16 additional milliseconds.
 - If it still does not complete, it is preempted and moved to queue Q_2 .

CPU Scheduling 24

Real-Time Scheduling

- *Hard real-time* systems – required to complete a critical task within a guaranteed amount of time.
- *Soft real-time* computing – requires that critical processes receive priority over less fortunate ones.
