

CPU Scheduling

Operating System Concepts

CPU Scheduling

- Basic Concepts
- Scheduling Criteria
- Scheduling Algorithms
- Multiple-Processor Scheduling
- Real-Time Scheduling
- Thread Scheduling
- Operating Systems Scheduling Examples
- Scheduling Algorithm Evaluation

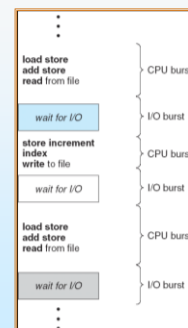
Operating System Concepts

Basic Concepts

- Maximum CPU utilization obtained with multiprogramming
- CPU-I/O Burst Cycle – Process execution consists of a *cycle* of CPU execution and I/O wait
- CPU burst distribution

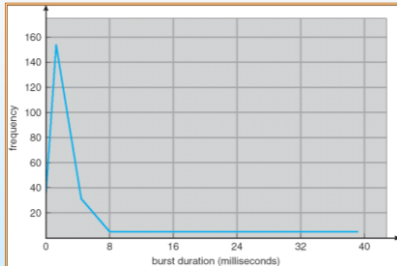
Operating System Concepts

Alternating Sequence of CPU And I/O Bursts: Simple Example



Operating System Concepts

Histogram of a Typical CPU-burst Times



Operating System Concepts

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 when a process:
 1. Switches from **running** to **waiting** state
 2. Switches from **running** to **ready** state
 3. Switches from **waiting** to **ready**
 4. Terminates
- Scheduling under 1 and 4 is *nonpreemptive*
- Others are *preemptive*

Operating System Concepts

Dispatcher: Starting execution of a process

- Dispatcher module gives control of the CPU to the process selected by the short-term scheduler; this involves:
 - switching context
 - switching to user mode
 - jumping to the proper location in the user program (address in PC) to restart that program
- *Dispatch latency* – time it takes for the dispatcher to stop one process and start another running

Operating System Concepts

Scheduling Criteria: Higher performance

- CPU utilization – keep the CPU as busy as possible
- Throughput – # of processes that complete their execution per time unit
- Turnaround time – amount of time to execute a particular process
- Waiting time – amount of time a process has been waiting in the ready queue
- Response time – amount of time it takes from when a request was submitted until the first response is produced, **not** output (for time-sharing environment)

Operating System Concepts

Optimization Criteria

- Max CPU utilization
- Max throughput
- Min turnaround time
- Min waiting time
- Min response time

Operating System Concepts

Scheduling Algorithms

- First Come First Serve-FCFS
- Shortest Job First
- Priority and Priority Classes
- Round Robin
- Multilevel Feedback Queues

Operating System Concepts

First-Come, First-Served (FCFS) Scheduling

Process	Burst Time
P_1	24
P_2	3
P_3	3

- Suppose that the processes arrive in the order: P_1, P_2, P_3
The Gantt Chart for the schedule is:



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

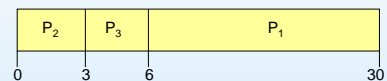
Operating System Concepts

FCFS Scheduling (Cont.)

Suppose that the processes arrive in the order

P_2, P_3, P_1

- Then the Gantt chart for the schedule is:



- 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

Operating System Concepts

Shortest-Job-First (SJR) Scheduling

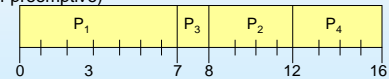
- Associate with each process the length of its next CPU burst. Use these lengths to schedule the process with the shortest time
- Two schemes:
 - nonpreemptive – once CPU given to the process it cannot be preempted until completes its CPU burst
 - preemptive – if a new process arrives with CPU burst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

Operating System Concepts

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)



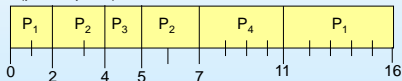
- Average waiting time = $(0 + 6 + 3 + 7)/4 = 4$

Operating System Concepts

Example of 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 (preemptive)



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

Operating System Concepts

Determining Length of Next CPU Burst

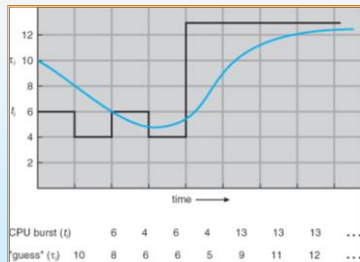
- Estimation of the length of CPU burst time
- Exponential Averaging Method: Can be done by using the length of previous CPU bursts:

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

1. t_n = actual length of n^{th} CPU burst
2. τ_{n+1} = predicted value for the next CPU burst
3. τ_n = Previous predicted value for CPU burst
4. $\alpha, 0 \leq \alpha \leq 1$

Operating System Concepts

Prediction of the Length of the Next CPU Burst



Operating System Concepts

Formulation of Exponential Averaging

- $\alpha = 0$
 - $\tau_{n+1} = \tau_n$
 - Recent history does not count
- $\alpha = 1$
 - $\tau_{n+1} = \alpha t_n$
 - Only the actual last CPU burst counts
- If we expand the formula, we get:

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

$$+ (1 - \alpha)^j \alpha t_{n-j} + \dots$$

$$+ (1 - \alpha)^{n+1} \tau_0$$
- Since both α and $(1 - \alpha)$ are less than or equal to 1, each successive term has less weight than its predecessor

Operating System Concepts

Priority Scheduling

- A priority number (integer) is associated with each process
- The CPU is allocated to the process with the highest priority (smallest integer \equiv highest priority)
 - Preemptive
 - nonpreemptive
- SJF is a priority scheduling where priority is the predicted next CPU burst time
- Problem \equiv Starvation – low priority processes may never execute
- Solution \equiv Aging – as time progresses increase the priority of the process

Operating System Concepts

Round Robin (RR)

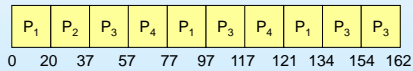
- Each process gets a small unit of CPU time (*time quantum*), usually 10-100 milliseconds. After this time has elapsed, the process is preempted and added to the end of the ready queue.
- If there are n processes in the ready queue and the time quantum is q , then each process gets $1/n$ of the CPU 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

Operating System Concepts

Example of RR with Time Quantum = 20

Process	Burst Time
P_1	53
P_2	17
P_3	68
P_4	24

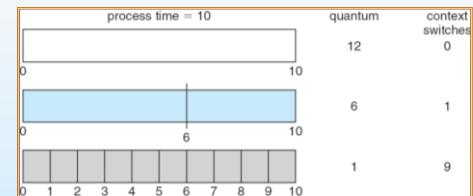
- The Gantt chart is:



- Typically, higher average turnaround than SJF, but better *response*

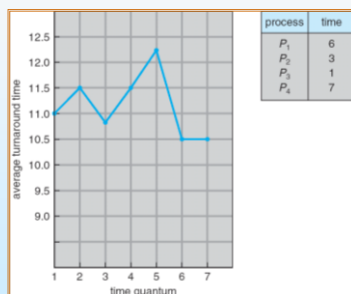
Operating System Concepts

Time Quantum and Context Switch Time



Operating System Concepts

Turnaround Time Varies With The Time Quantum



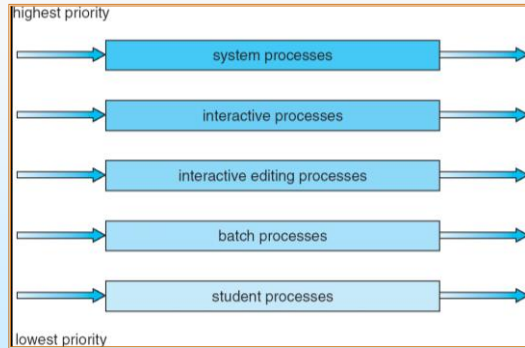
Operating System Concepts

Multilevel Queue: Used in real OS cases

- 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 as well:
 - Fixed priority scheduling; (i.e., serve all from foreground then from background). This may cause starvation.
 - Time slice: each queue gets a certain amount of CPU time which it can schedule amongst its processes; i.e., 80% to foreground in RR
 - 20% to background in FCFS

Operating System Concepts

Multilevel Queue Scheduling



Operating System Concepts

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
 - method used to determine which queue a process will enter when that process needs service

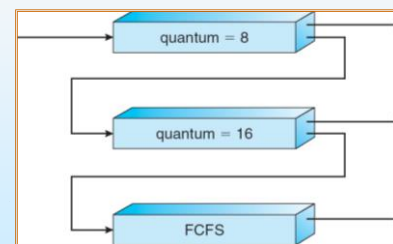
Operating System Concepts

Example of Multilevel Feedback Queue

- Three queues:
 - Q_0 – RR with time quantum 8 milliseconds
 - Q_1 – RR time quantum 16 milliseconds
 - Q_2 – pure FCFS
- Scheduling
 - A new job enters queue Q_0 which is served RR. 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 RR and receives 16 additional milliseconds. If it still does not complete, it is preempted and moved to queue Q_2 .

Operating System Concepts

Multilevel Feedback Queues



Operating System Concepts

Multiple-Processor Scheduling

- CPU scheduling more complex when multiple CPUs are available
- *Homogeneous processors* within a multiprocessor
 - *Load sharing with proper distribution is required*
- *Asymmetric multiprocessing*; for example, only one processor accesses the system data structures, alleviating the need for data sharing

Operating System Concepts

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

Operating System Concepts

The Relationship Between Priorities and Time-slice length

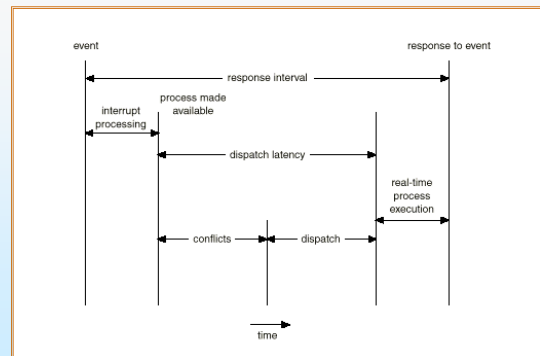
numeric priority	relative priority	time quantum
0	highest	200 ms
•		
•		
99		
100		
•	lowest	10 ms
•		
•		
140		
•		

real-time tasks

other tasks

Operating System Concepts

Realtime Events Require Dispatch Latency



Operating System Concepts