

Operating
Systems:
Internals
and
Design
Principles

Chapter 9 Uniprocessor Scheduling

Ninth Edition By William Stallings



Table 9.1

Types of Scheduling

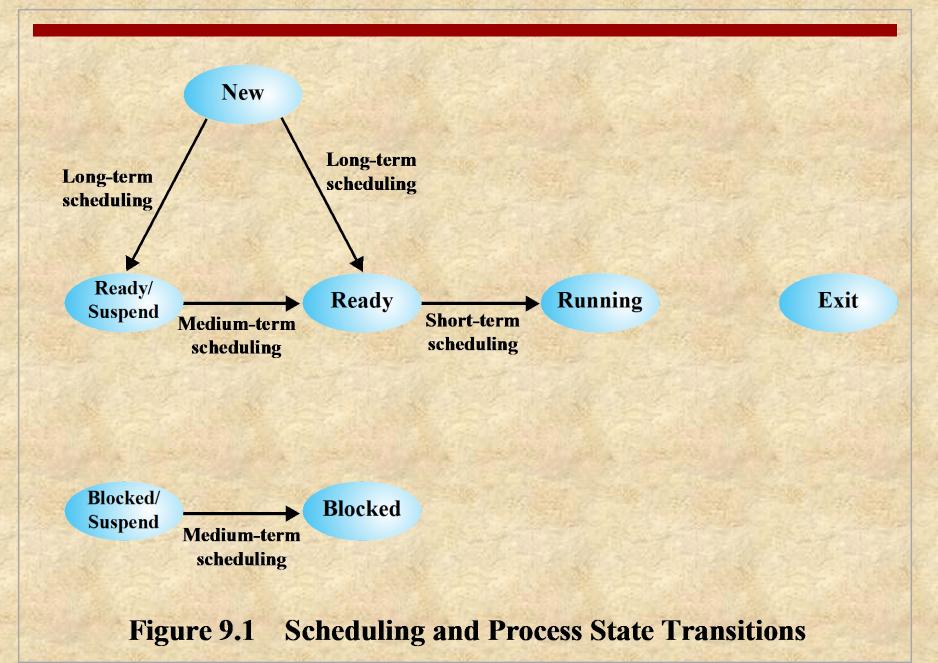
Lo	ong-term scheduling	The decision to add to the pool of processes to be executed
M∈	edium-term scheduling	The decision to add to the number of processes that are partially or fully in main memory
Sì	hort-term scheduling	The decision as to which available process will be executed by the processor
I	O scheduling	The decision as to which process's pending I/O request shall be handled by an available I/O device



- Aim is to assign processes to be executed by the processor in a way that meets system objectives, such as response time, throughput, and processor efficiency
- Broken down into three separate functions:









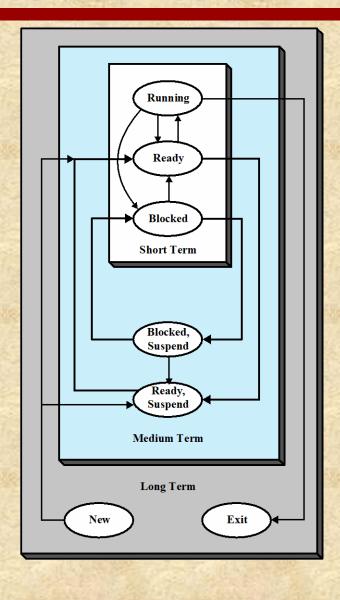
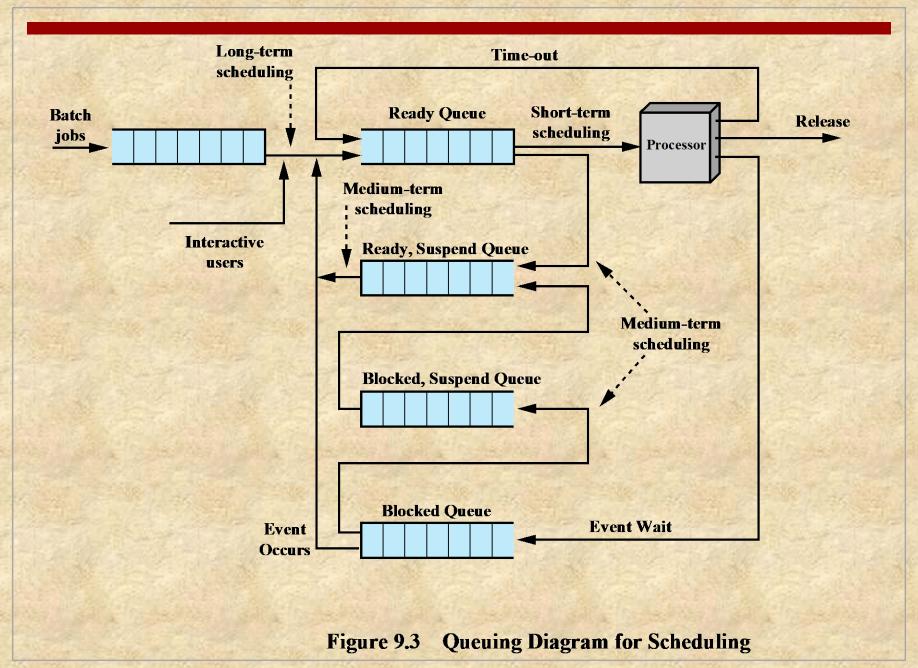


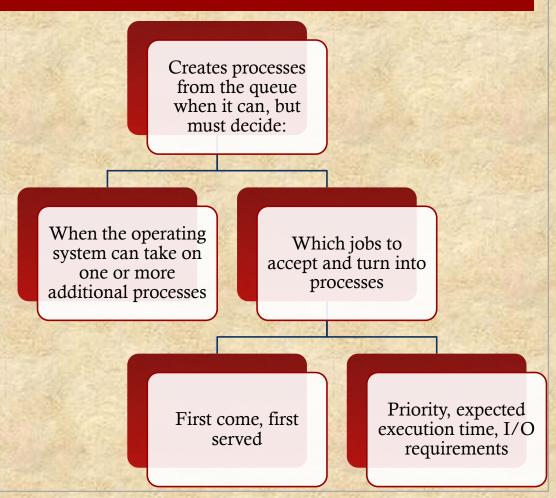
Figure 9.2 Levels of Scheduling







- Determines which programs are admitted to the system for processing
- Controls the degree of multiprogramming
 - The more processes that are created, the smaller the percentage of time that each process can be executed
 - May limit to provide satisfactory service to the current set of processes





Medium-Term Scheduling

- Part of the swapping function
- Swapping-in decisions are based on the need to manage the degree of multiprogramming
 - Considers the memory requirements of the swapped-out processes



- Known as the dispatcher
- Executes most frequently
- Makes the fine-grained decision of which process to execute next
- Invoked when an event occurs that may lead to the blocking of the current process or that may provide an opportunity to preempt a currently running process in favor of another

Examples:

- Clock interrupts
- I/O interrupts
- Operating system calls
- Signals (e.g., semaphores)



- Main objective is to allocate processor time to optimize certain aspects of system behavior
- A set of criteria is needed to evaluate the scheduling policy

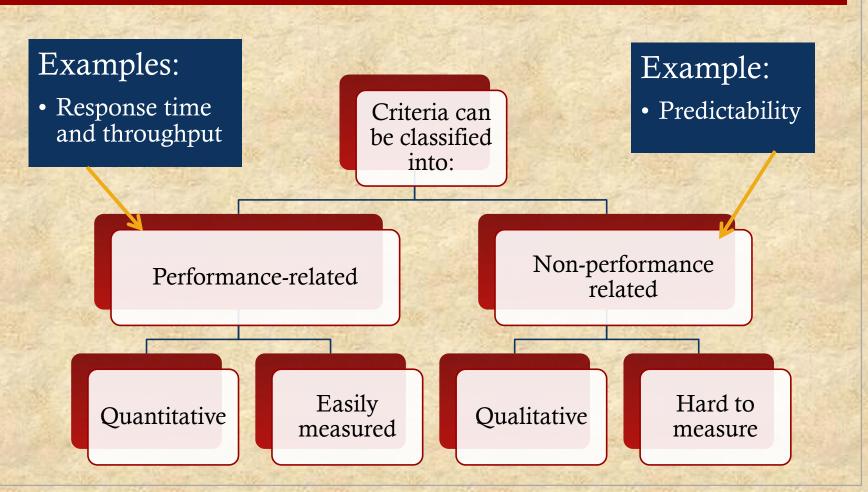
User-oriented criteria

- Relate to the behavior of the system as perceived by the individual user or process (such as response time in an interactive system)
- Important on virtually all systems

System-oriented criteria

- Focus is on effective and efficient utilization of the processor (rate at which processes are completed)
- Generally of minor importance on singleuser systems







User Oriented, Performance Related

Turnaround time This is the interval of time between the submission of a process and its completion. Includes actual execution time plus time spent waiting for resources, including the processor. This is an appropriate measure for a batch job.

Response time For an interactive process, this is the time from the submission of a request until the response begins to be received. Often a process can begin producing some output to the user while continuing to process the request. Thus, this is a better measure than turnaround time from the user's point of view. The scheduling discipline should attempt to achieve low response time and to maximize the number of interactive users receiving acceptable response time.

Deadlines When process completion deadlines can be specified, the scheduling discipline should subordinate other goals to that of maximizing the percentage of deadlines met.

User Oriented, Other

Predictability A given job should run in about the same amount of time and at about the same cost regardless of the load on the system. A wide variation in response time or turnaround time is distracting to users. It may signal a wide swing in system workloads or the need for system tuning to cure instabilities.

System Oriented, Performance Related

Throughput The scheduling policy should attempt to maximize the number of processes completed per unit of time. This is a measure of how much work is being performed. This clearly depends on the average length of a process but is also influenced by the scheduling policy, which may affect utilization.

Processor utilization This is the percentage of time that the processor is busy. For an expensive shared system, this is a significant criterion. In single-user systems and in some other systems, such as real-time systems, this criterion is less important than some of the others.

System Oriented, Other

Fairness In the absence of guidance from the user or other systemsupplied guidance, processes should be treated the same, and no process should suffer starvation.

Enforcing priorities When processes are assigned priorities, the scheduling policy should favor higher-priority processes.

Balancing resources The scheduling policy should keep the resources of the system busy. Processes that will underutilize stressed resources should be favored. This criterion also involves medium-term and long-term scheduling.

Table 9.2

Scheduling Criteria

(Table can be found on page 403 in textbook)



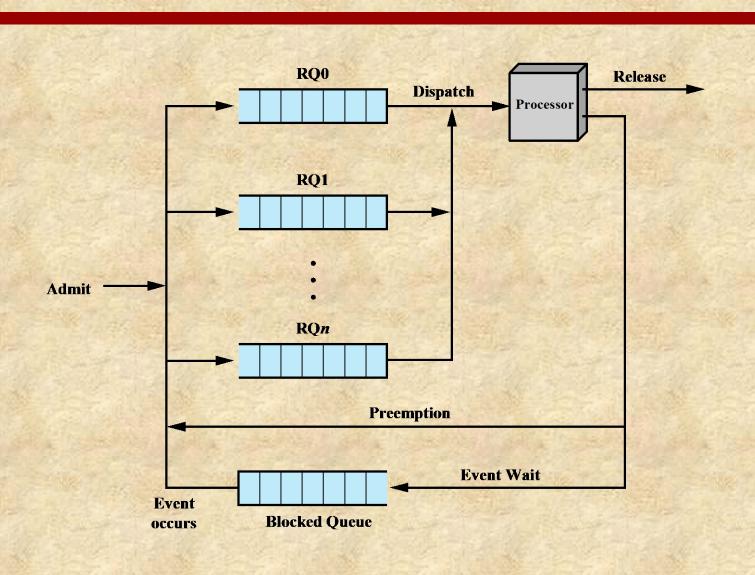


Figure 9.4 Priority Queuing



	FCFS	Round robin	SPN	SRT	HRRN	Feedback
Selection function	max[w]	constant	min[s]	min[s - e]	$\max\left(\frac{w+s}{s}\right)$	(see text)
Decision mode	Non- preemptive	(at time		Preemptive (at arrival)	Non- preemptive	Preemptive (at time quantum)
Through- Put	Not emphasized	May be low if quantum is too small			High	Not emphasized
Response time	May be high, especially if there is a large variance in process execution times	Provides good response time for short processes	Provides good response time for short processes	Provides good response time	Provides good response time	Not emphasized
Overhead	Minimum	nimum Minimum Can be high		Can be high Can be high		Can be high
Effect on processes	Penalizes short processes; penalizes I/O bound processes	Fair treatment	Penalizes long processes	Penalizes long processes	Good balance	May favor I/O bound processes
Starvation	No	No	Possible	Possible	No	Possible

Table 9.3

Characteristics of Various Scheduling Policies

(Table can be found on page 405 in textbook)



Selection Function

- Determines which process, among ready processes, is selected next for execution
- May be based on priority, resource requirements, or the execution characteristics of the process
- If based on execution characteristics, then important quantities are:
 - w = time spent in system so far, waiting
 - e = time spent in execution so far
 - s = total service time required by the process, including e; generally, this quantity must be estimated or supplied by the user



Specifies the instants in time at which the selection function is exercised

- Two categories:
 - Nonpreemptive
 - Preemptive



Nonpreemptive

 Once a process is in the running state, it will continue until it terminates or blocks itself for I/O

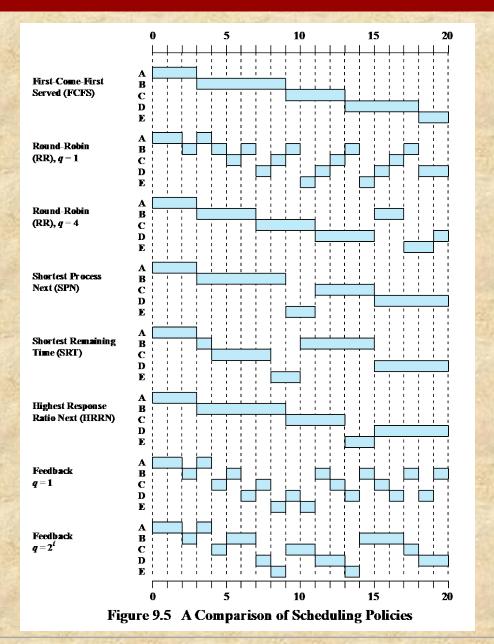
Preemptive

- Currently running process may be interrupted and moved to ready state by the OS
- Decision to preempt may be performed when a new process arrives, when an interrupt occurs that places a blocked process in the Ready state, or periodically, based on a clock interrupt

Process	Arrival Time	Service Time
A	0	3
В	2	6
С	4	4
D	6	5
Е	8	2

Table 9.4 Process Scheduling Example







Process	A	В	С	D	Е				
Arrival Time	0	2	4	6	8	2521			
Service Time (T_s)	3	6	4	5	2	Mean			
Mary Mary 18		F	CFS		19/21/2	# 0.5 May 0			
Finish Time	3	9	13	18	20				
Turnaround Time (T_r)	3	7	9	12	12	8.60			
T_{P}/T_{S}	1.00	1.17	2.25	2.40	6.00	2.56			
THE NAME OF STREET	(30-7)	RR q	=1	Carrie					
Finish Time	4	18	17	20	15				
Turnaround Time (T_r)	4	16	13	14	7	10.80			
T_{P}/T_{S}	1.33	2.67	3.25	2.80	3.50	2.71			
THE PERSON NAMED IN COLUMN TWO IS NOT THE PERSON NAMED IN COLUMN TWO IS NA	A Lower	RR q	= 4	1					
Finish Time	3	17	11	20	19				
Turnaround Time (T_r)	3	15	7	14	11	10.00			
T_{r}/T_{s}	1.00	2.5	1.75	2.80	5.50	2.71			
		SP	N		30.00				
Finish Time	3	9	15	20	11				
Turnaround Time (T_r)	3	7	11	14	3	7.60			
T_{r}/T_{s}	1.00	1.17	2.75	2.80	1.50	1.84			
	A 24.50	SR	T			ALC: NO.			
Finish Time	3	15	8	20	10				
Turnaround Time (T_r)	3	13	4	14	2	7.20			
T_{r}/T_{s}	1.00	2.17	1.00	2.80	1.00	1.59			
Market As and	1009 L	HR	RN	189 500	ella B	A DES			
Finish Time	3	9	13	20	15				
Turnaround Time (T_r)	3	7	9	14	7	8.00			
T_{r}/T_{s}	1.00	1.17	2.25	2.80	3.5	2.14			
Carrie Direction		FB q			18 NO.				
Finish Time	4	20	16	19	11				
Turnaround Time (T_r)	4	18	12	13	3	10.00			
T_r/T_s	1.33	3.00	3.00	2.60	1.5	2.29			
FB q = 2i									
Finish Time	4	17	18	20	14				
Turnaround Time (T_r)	4	15	14	14	6	10.60			
T_r/T_s	1.33	2.50	3.50	2.80	3.00	2.63			

Table 9.5

A Comparison of Scheduling Policies

(Table is on page 408 in textbook)



First-Come-First-Served (FCFS)

- Simplest scheduling policy
- Also known as first-in-first-out (FIFO) or a strict queuing scheme
- As each process becomes ready, it joins the ready queue
- When the currently running process ceases to execute, the process that has been in the ready queue the longest is selected for running

- Performs much better for long processes than short ones
- Tends to favor processor-bound processes over I/O-bound processes

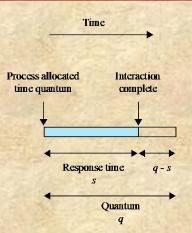


Round Robin

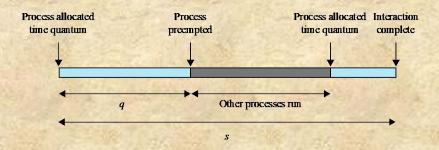
- Uses preemption based on a clock
- Also known as time slicing because each process is given a slice of time before being preempted
- Principal design issue is the length of the time quantum, or slice, to be used

- Particularly effective in a general-purpose time-sharing system or transaction processing system
- One drawback is its relative treatment of processor-bound and I/O-bound processes





(a) Time quantum greater than typical interaction



(b) Time quantum less than typical interaction

Figure 9.6 Effect of Size of Preemption Time Quantum



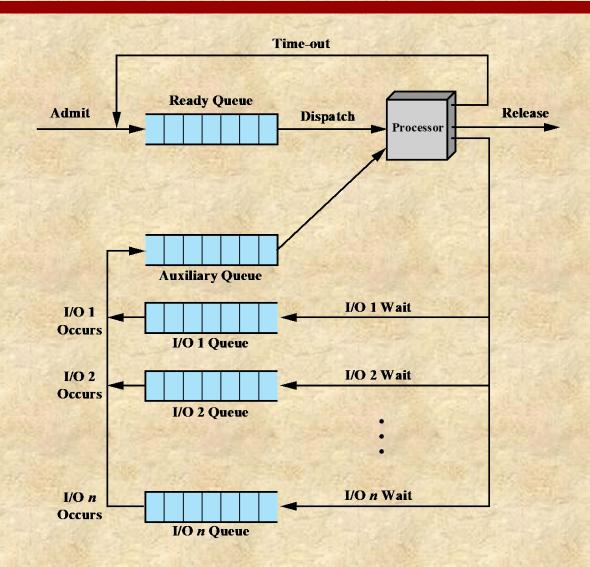


Figure 9.7 Queuing Diagram for Virtual Round-Robin Scheduler

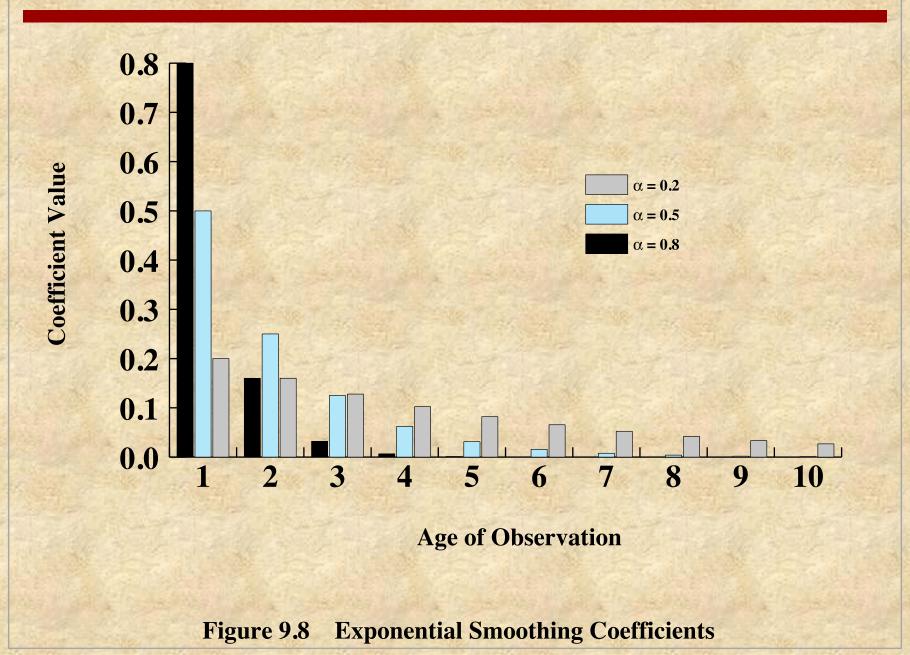


Shortest Process Next (SPN)

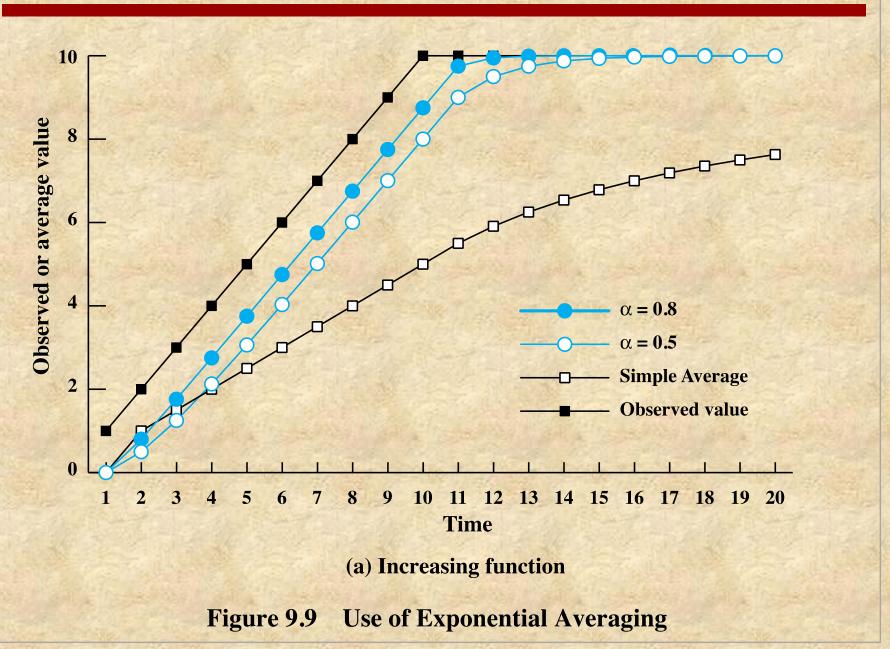
- Nonpreemptive policy in which the process with the shortest expected processing time is selected next
- A short process will jump to the head of the queue
- Possibility of starvation for longer processes

- One difficulty is the need to know, or at least estimate, the required processing time of each process
- If the programmer's estimate is substantially under the actual running time, the system may abort the job

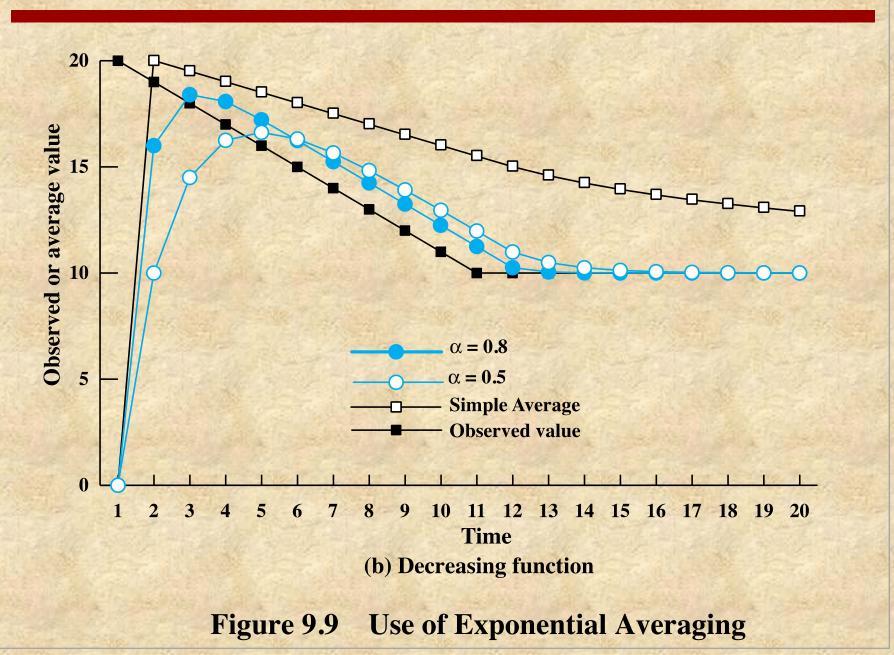














Shortest Remaining Time (SRT)

- Preemptive version of SPN
- Scheduler always chooses the process that has the shortest expected remaining processing time
- Risk of starvation of longer processes

Should give superior turnaround time performance to SPN because a short job is given immediate preference to a running longer job



Highest Response Ratio Next (HRRN)

- Chooses next process with the greatest ratio
- Attractive because it accounts for the age of the process

• While shorter jobs are favored, aging without service increases the ratio so that a longer process will eventually get past competing shorter jobs

 $Ratio = \frac{time\ spent\ waiting + expected\ service\ time}{expected\ service\ time}$



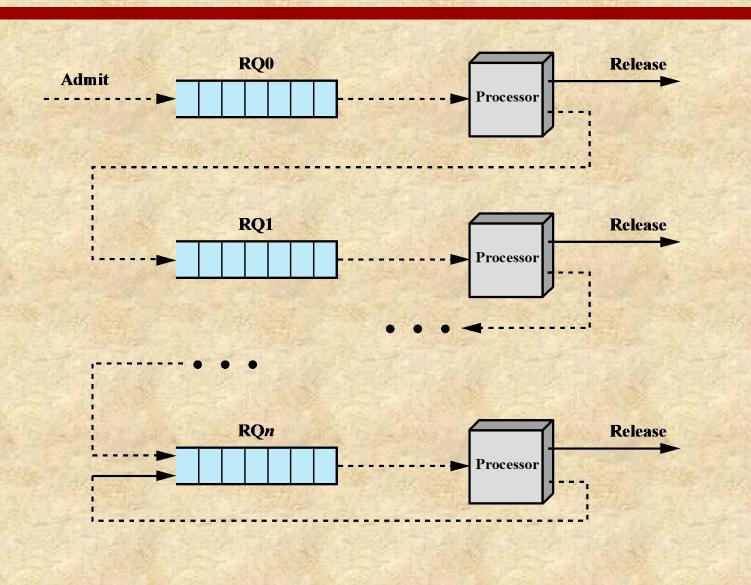


Figure 9.10 Feedback Scheduling



Performance Comparison

■ Any scheduling discipline that chooses the next item to be served independent of service time obeys the relationship:

$$\frac{T_r}{T_s} = \frac{1}{1 - \rho}$$

where

 T_r = turnaround time or residence time; total time in system, waiting plus execution

 T_s = average service time; average time spent in Running state

 ρ = processor utilization



Table 9.6

Formulas
for SingleServer
Queues
with Two
Priority
Categories

Assumptions: 1. Poisson arrival rate.

- **2.** Priority 1 items are serviced before priority 2 items.
- **3.** First-come-first-served dispatching for items of equal priority.
- **4.** No item is interrupted while being served.
- **5.** No items leave the queue (lost calls delayed).

(a) General formulas

$$\lambda = \lambda_1 + \lambda_2$$

$$\rho_1 = \lambda_1 T_{s1}; \quad \rho_2 = \lambda_2 T_{s2}$$

$$\rho = \rho_1 + \rho_2$$

$$T_s = \frac{\lambda_1}{\lambda} T_{s1} + \frac{\lambda_2}{\lambda} T_{s2}$$

$$T_r = \frac{\lambda_1}{\lambda} T_{r1} + \frac{\lambda_2}{\lambda} T_{r2}$$

(b) No interrupts; exponential service times

$$T_{r1} = T_{s1} + \frac{\rho_1 T_{s1} + \rho_2 T_{s2}}{1 - \rho_1}$$
$$T_{r2} = T_{s2} + \frac{T_{r1} - T_{s1}}{1 - \rho}$$

(c) Preemptive-resume queuing discipline; exponential service times

$$T_{r1} = T_{s1} + \frac{\rho_1 T_{s1}}{1 - \rho_1}$$

$$T_{r2} = T_{s2} + \frac{1}{1 - \rho_1} \left(\rho_1 T_{s2} + \frac{\rho T_s}{1 - \rho} \right)$$



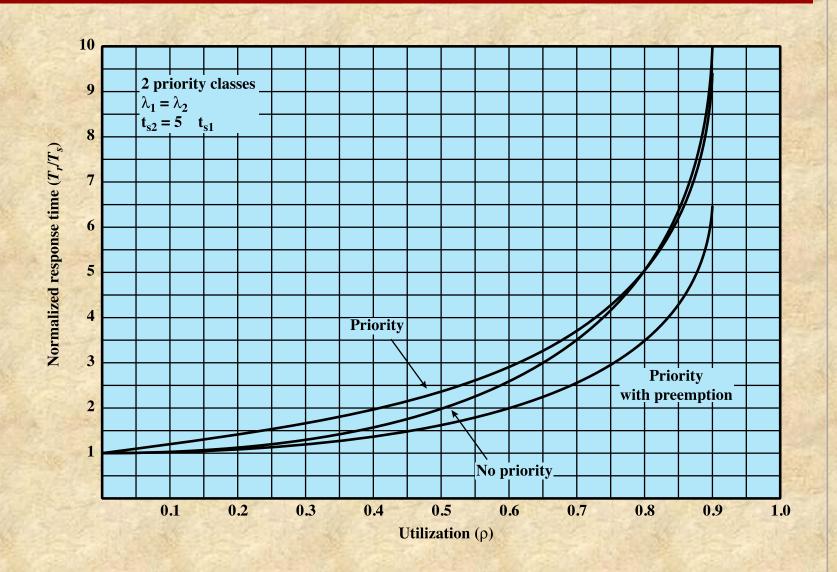


Figure 9.11 Overall Normalized Response Time



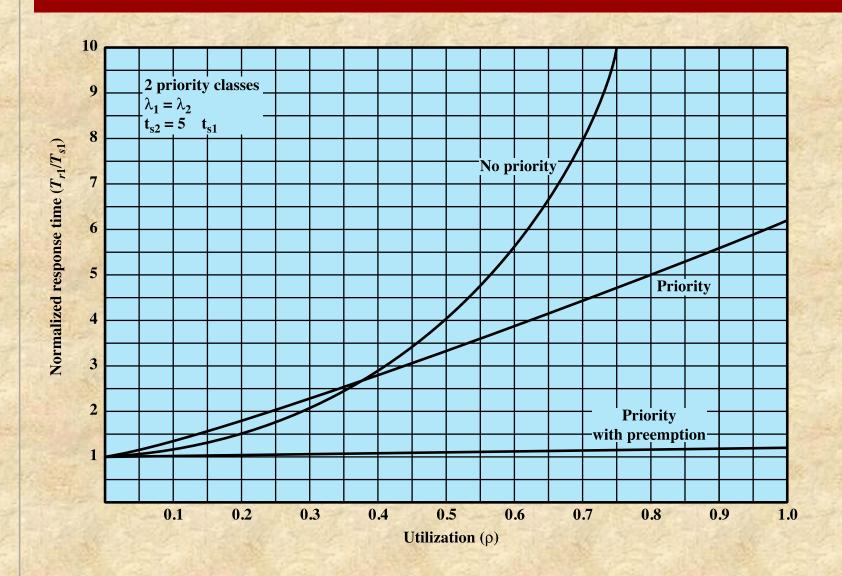


Figure 9.12 Normalized Response Time for Shorter Processes



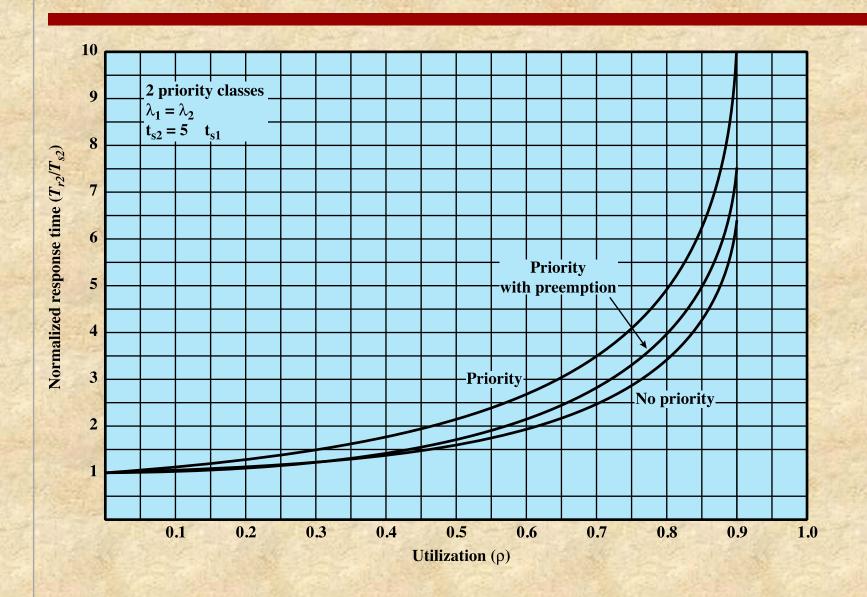


Figure 9.13 Normalized Response Time for Longer Processes



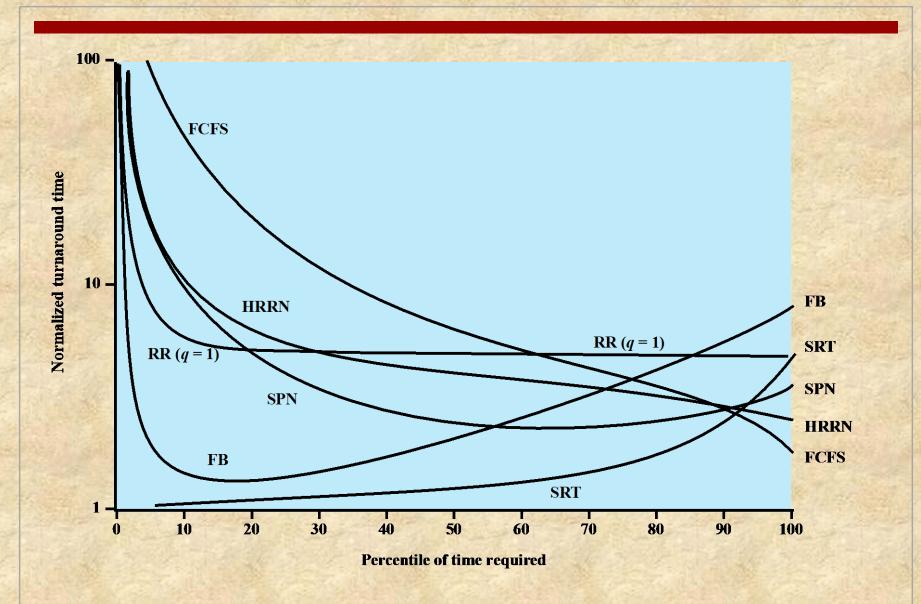


Figure 9.14 Simulation Results for Normalized Turnaround Time



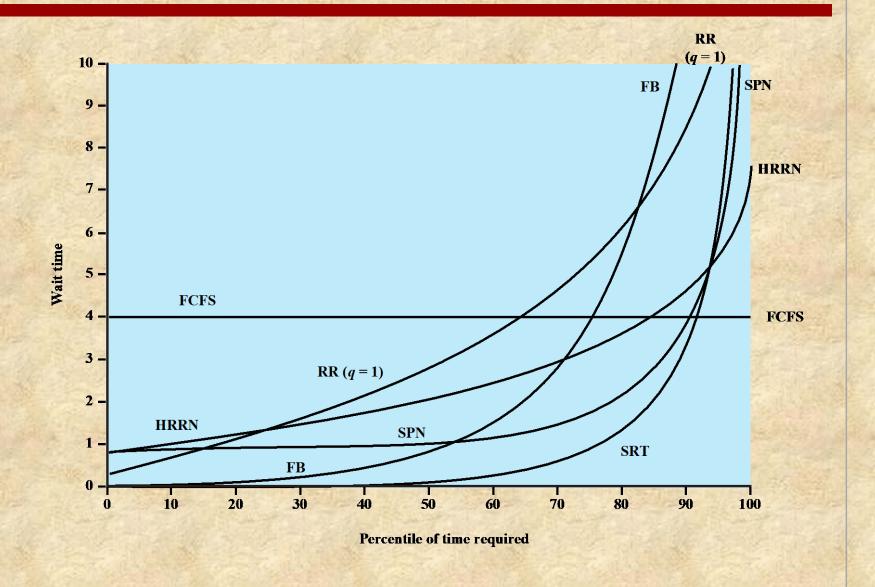


Figure 9.15 Simulation Results for Waiting Time



Fair-Share Scheduling

- Scheduling decisions based on the process sets
- Each user is assigned a share of the processor
- Objective is to monitor usage to give fewer resources to users who have had more than their fair share and more to those who have had less than their fair share

Fair Share Scheduler (FFS)

The following formulas apply for process j in group k:

$$CPU_{j}(i) = \frac{CPU_{j}(i-1)}{2}$$

$$GCPU_{k}(i) = \frac{GCPU_{k}(i-1)}{2}$$

$$P_{j}(i) = Base_{j} + \frac{CPU_{j}(i)}{2} + \frac{GCPU_{k}(i)}{4 \times W_{k}}$$

where

CPU_j(i) = measure of processor utilization by process j through interval i, GCPU_k(i) = measure of processor utilization of group k through interval i, P_j(i) = priority of process j at beginning of interval i; lower values equal higher priorities, Base_j = base priority of process j, and W_k = weighting assigned to group k, with the constraint that and $0 < W_k \le 1$ and $\sum W_k = 1$.



	Process A				Process B			Process C		
		Process CPU	Group CPU		Process CPU	Group CPU		Process CPU	Group CPU	
Time	Priority	count	count	Priority	count	count	Priority	count	count	
0	60	0	0	60	0	0	60	0	0	
James !		1	1							
		2	2							
400		•	•							
1000		60	60							
1	90	30	30	60	0	0	60	0	0	
9 3 4					1	1			1	
					2	2			2	
					•	•			•	
100					60	60			60	
2	74	15	15	90	30	30	75	0	30	
1,952		16	16							
48.50		17	17							
- 19		•	•							
		• 75	• 75							
3 —	96	37	37	74	15	15	67	0	15	
12000						16		1	16	
						17		2	17	
						•		•	•	
						• 75		• 60	75	
4 —	78	18	18	81	7	37	93	30	37	
		19	19							
The sale		20	20							
100		•	•							
1		• 78	• 78							
5 —	98	39	39	70	3	18	76	15	18	
1000										
2 54										
		71-1				0.3-15	September 2	200		
	103364		30,100					ALC: N		
		Group 1				Gro	up 2			

Colored rectangle represents executing process

Figure 9.16 Example of Fair Share Scheduler—Three Processes, Two Groups



Traditional UNIX Scheduling

- Used in both SVR3 and 4.3 BSD UNIX
 - These systems are primarily targeted at the time-sharing interactive environment
- Designed to provide good response time for interactive users while ensuring that low-priority background jobs do not starve
- Employs multilevel feedback using round robin within each of the priority queues
- Makes use of one-second preemption
- Priority is based on process type and execution history

Scheduling Formula

$$CPU_{j}(i) = \frac{CPU_{j}(i-1)}{2}$$

$$P_{j}(i) = Base_{j} + \frac{CPU_{j}(i)}{2} + nice_{j}$$

where

 $CPU_j(i)$ = measure of processor utilization by process j through interval i

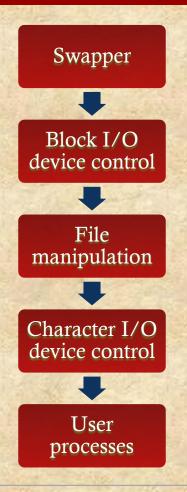
 $P_j(i)$ = priority of process j at beginning of interval i; lower values equal higher priorities

 $Base_j$ = base priority of process j

 $nice_i$ = user-controllable adjustment factor



- Used to optimize access to block devices and to allow the operating system to respond quickly to system calls
- In decreasing order of priority, the bands are:





TP:	Proc	ess A	Proc	ess B	Process C		
Time	Priority	CPU count	Priority	CPU count	Priority (CPU count	
0 —	60	0 1 2 •	60	0	60	0	
2 —	75	30	60	0 1 2 •	60	0	
	67	15	75	30	60	0 1 2 • •	
3 —	63	7 8 9 • •	67	15	75	30	
5 —	76	33	63	7 8 9 • •	67	15	
	68	16	76	33	63	7	

Colored rectangle represents executing process

Figure 9.17 Example of Traditional UNIX Process Scheduling



Summary

- Types of processor scheduling
 - Long-term scheduling
 - Medium-term scheduling
 - Short-term scheduling
- Traditional UNIX scheduling

- Scheduling algorithms
 - Short-term scheduling criteria
 - The use of priorities
 - Alternative scheduling policies
 - Performance comparison
 - Fair-share scheduling