Today's class

- Scheduling

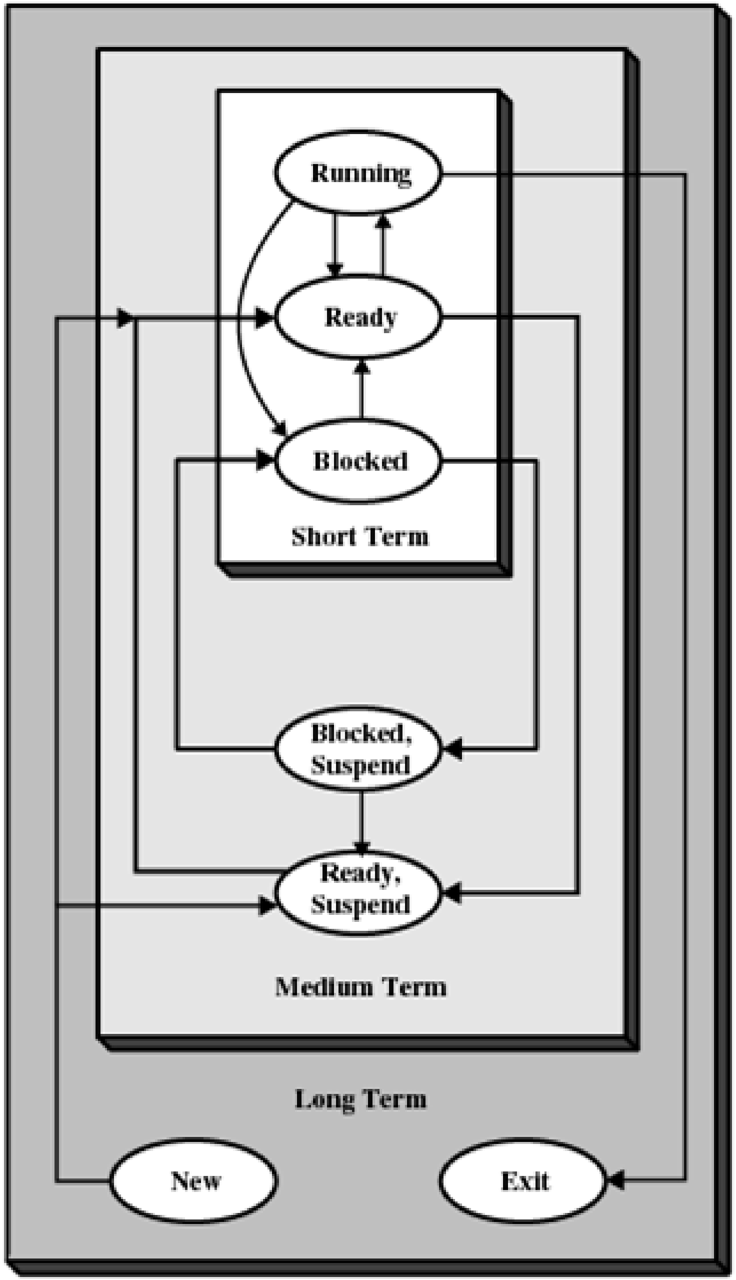
UPPSALA UNIVERSITET

Informationsteknologi

Aim of Scheduling

- Assign processes to be executed by the processor(s)
- Need to meet system objectives regarding:
  - Response time
  - Throughput
  - Processor efficiency

Levels of Scheduling



Long-Term Scheduling

- Determines which programs are admitted to the system for processing
- Controls the degree of multiprogramming
- More processes, smaller percentage of time each process is executed

UPPSALA UNIVERSITET

Informationsteknologi

Medium-Term Scheduling

- Part of the swapping function
- Based on the need to manage the degree of multiprogramming

UPPSALA UNIVERSITET

Informationsteknologi

Short-Term Scheduling

- Known as the dispatcher
- Executes most frequently
- Invoked when an event occurs
  - Clock interrupts
  - I/O interrupts
  - Operating system calls
  - Signals

UPPSALA UNIVERSITET

Informationsteknologi

Short-Tem Scheduling Criteria

- User-oriented
  - Response time - elapsed time between the submission of a request until there is output
- System-oriented
  - Effective and efficient utilization of the processor
- Performance-related
  - Quantitative and generally measurable, such as response time and throughput

UPPSALA UNIVERSITET

Informationsteknologi

Priorities

- Scheduler will always choose a process of higher priority over one of lower priority
- Have multiple ready queues to represent each level of priority
- Lower-priority may suffer starvation
  - Allow a process to change its priority based on its age or execution history

UPPSALA UNIVERSITET

Informationsteknologi

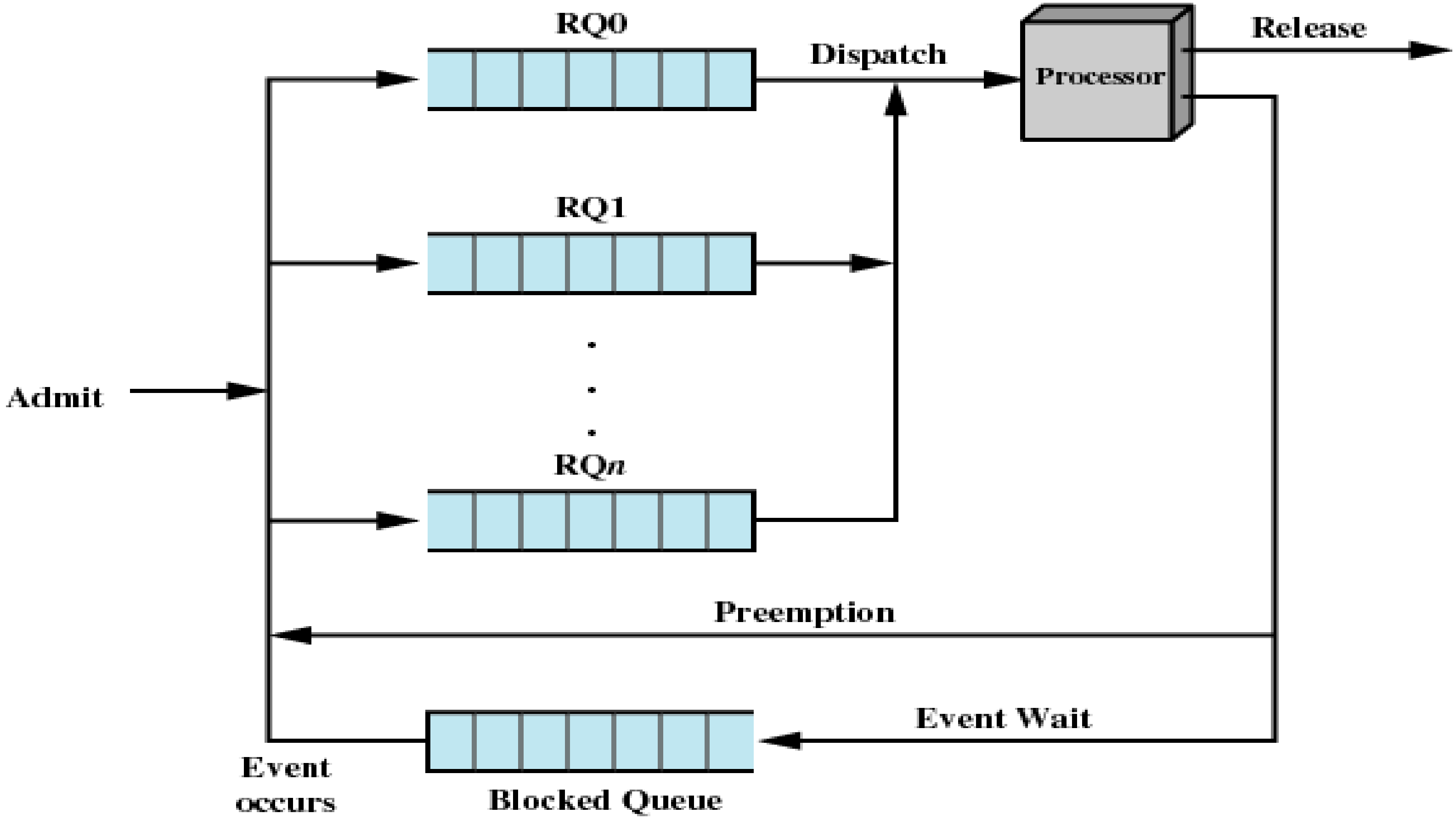


Figure 9.4 Priority Queuing

UPPSALA UNIVERSITET

Informationsteknologi

Decision Mode

- 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 the Ready state by the operating system
  - Allows for better service since any one process cannot monopolize the processor for very long

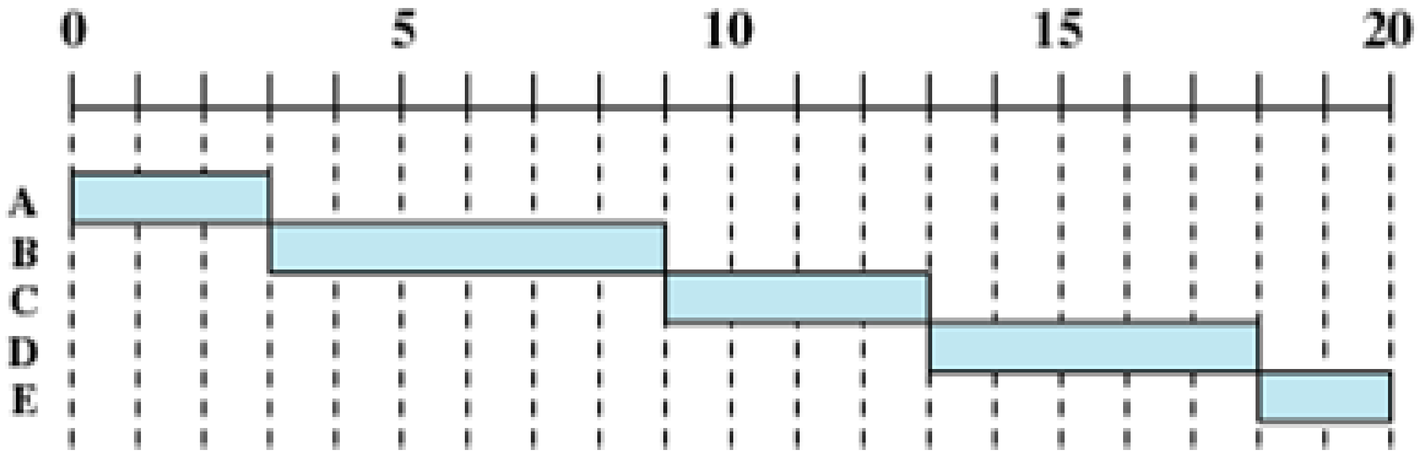
UPPSALA UNIVERSITET

Informationsteknologi

Process Scheduling Example

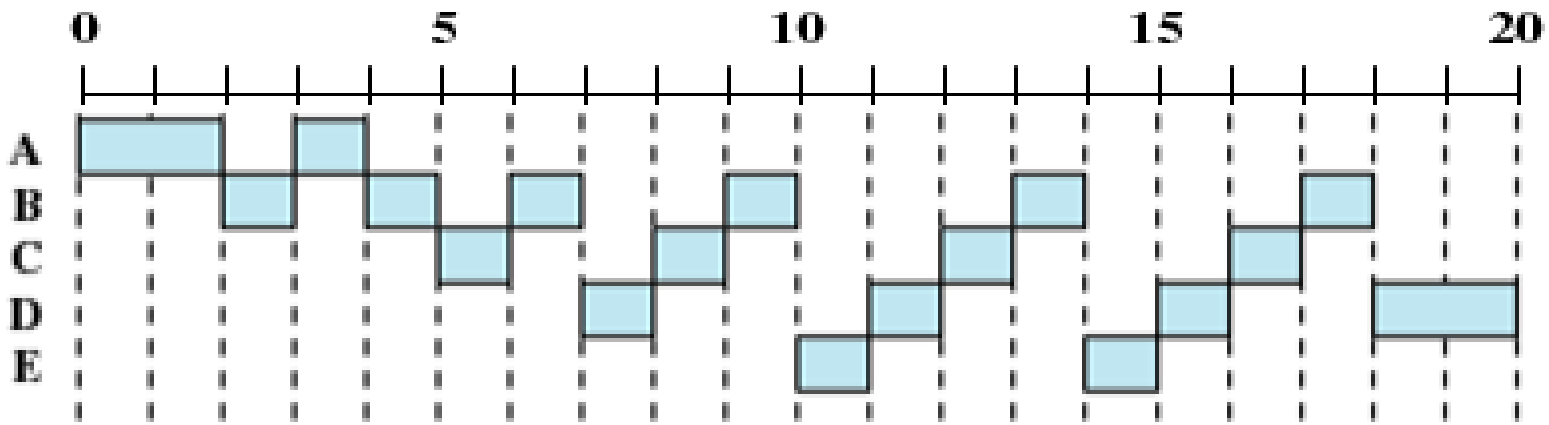
Process	Arrival Time	Service Time
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

First-Come-First-Served (FCFS)



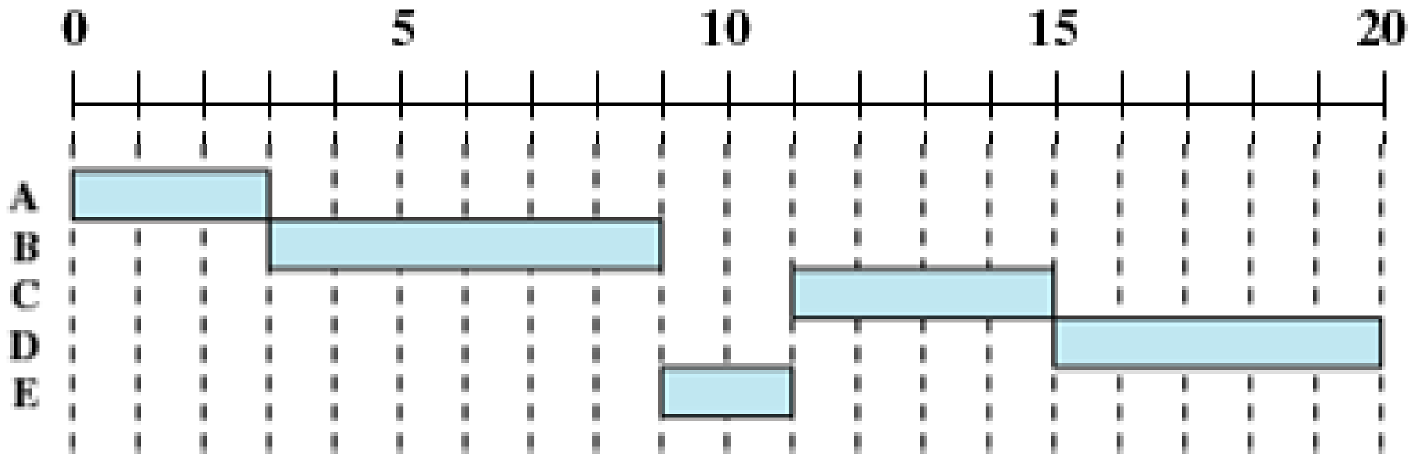
- Each process joins the Ready queue
- When the current process ceases to execute, the oldest process in the Ready queue is selected

Round-Robin



- Uses preemption based on a clock
- An amount of time is determined that allows each process to use the processor for that length of time

Shortest Process Next

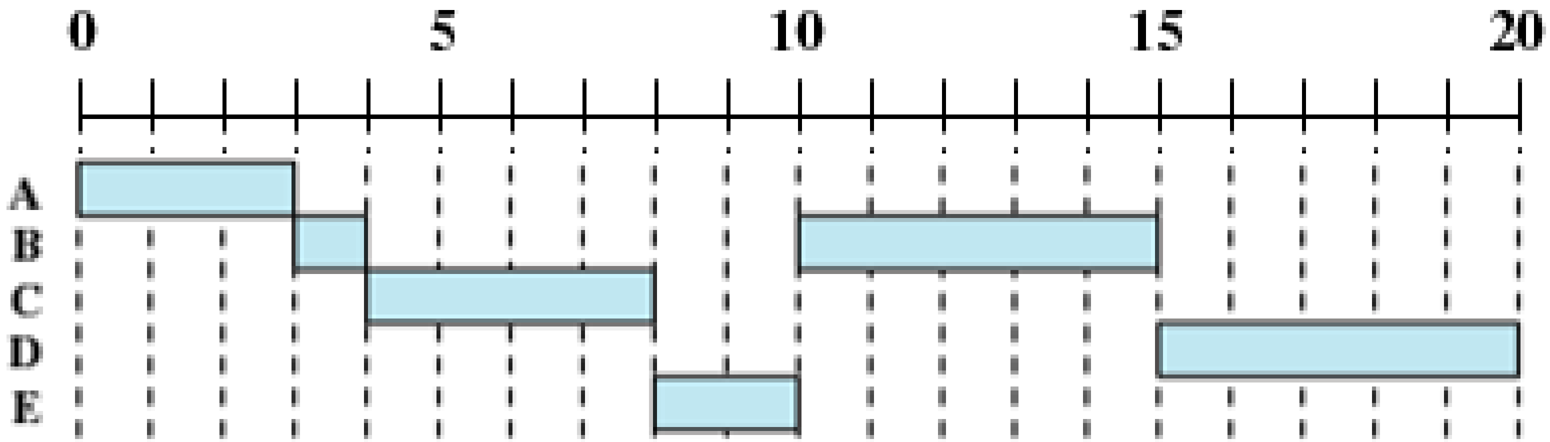


- Nonpreemptive policy
- Process with shortest expected processing time is selected next
- Short processes jump ahead of longer processes
- Possibility of starvation for longer processes

UPPSALA UNIVERSITET

Informationsteknologi

Shortest Remaining Time

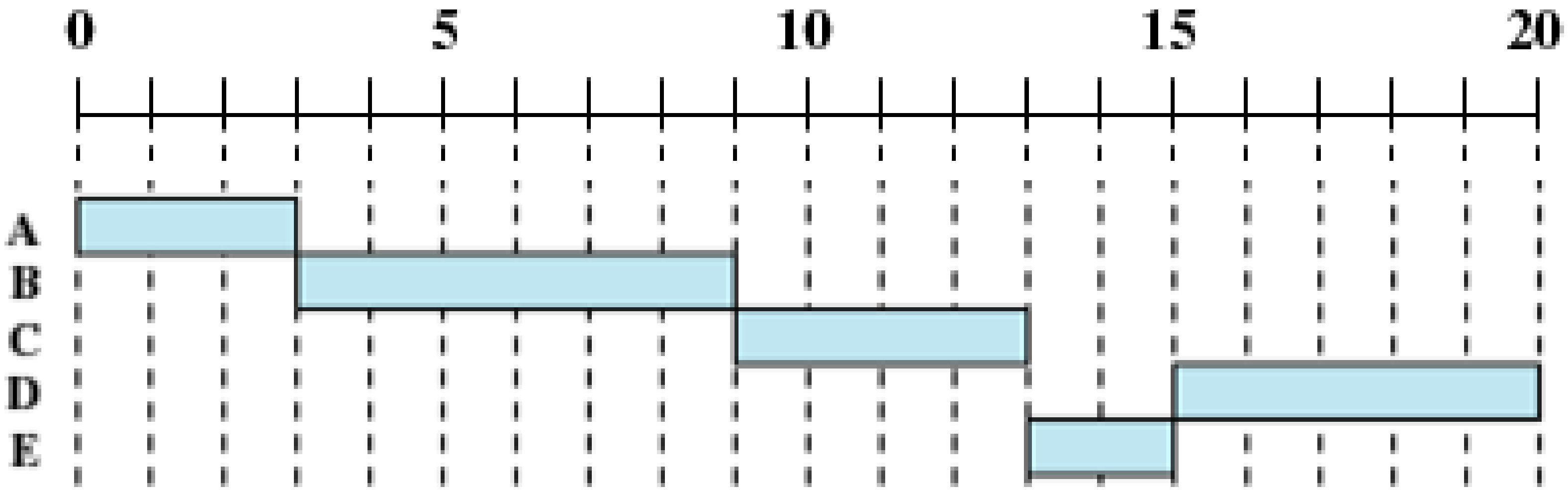


- Preemptive version of shortest process next policy
- Must estimate processing time

UPPSALA UNIVERSITET

Informationsteknologi

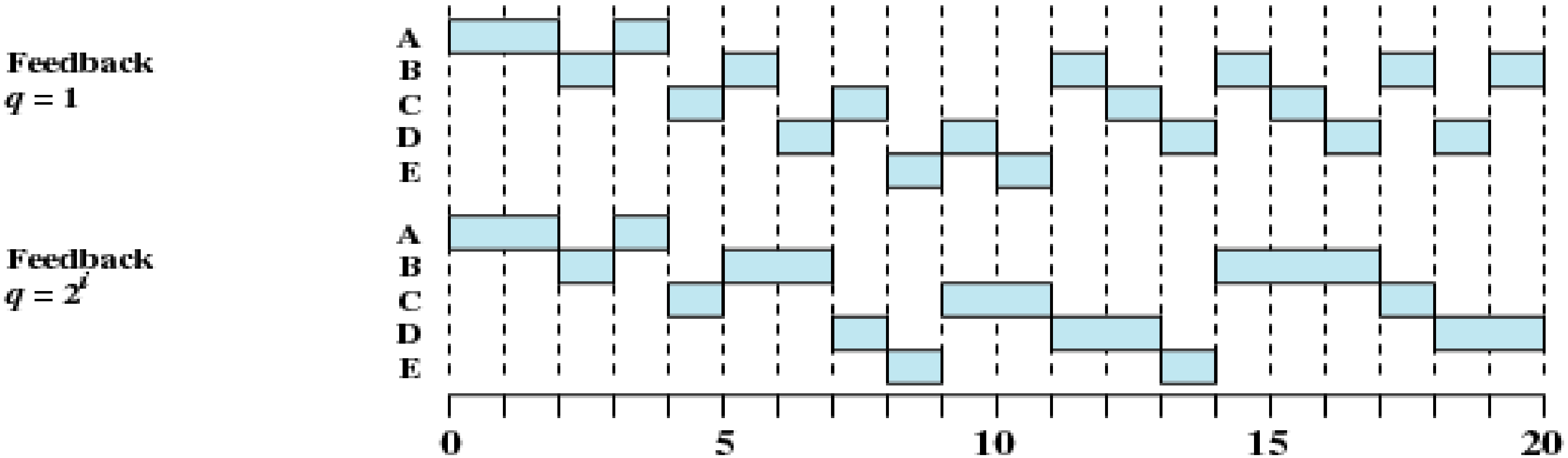
Highest Response Ratio Next (HRRN)



- Choose next process with the greatest ratio

$$R = \frac{w + s}{s}$$

Feedback



- Penalize jobs that have been running longer
- Don’t know remaining time process needs to execute

UPPSALA UNIVERSITET

Informationsteknologi

Table 9.3 Characteristics of Various Scheduling Policies

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

w = time spent waiting
e = time spent in execution so far
s = total service time required by the process, including e

UPPSALA UNIVERSITET

Table 9.5 A Comparison of Scheduling Policies

	Process	A	B	C	D	E	
	Arrival Time	0	2	4	6	8	
	Service Time (T_s)	3	6	4	5	2	Mean
FCFS	Finish Time	3	9	13	18	20	
	Turnaround Time (T_r)	3	7	9	12	12	8.60
	T_r/T_s	1.00	1.17	2.25	2.40	6.00	2.56
RR $q = 1$	Finish Time	4	18	17	20	15	
	Turnaround Time (T_r)	4	16	13	14	7	10.80
	T_r/T_s	1.33	2.67	3.25	2.80	3.50	2.71
RR $q = 4$	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
SPN	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
SRT	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
HRRN	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
FB $q = 1$	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 $q = 2^i$	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

UPPSALA UNIVERSITET

Informationsteknologi

Fair-Share Scheduling

- In a multiuser system there is a structure to the collection of processes not recognized by a traditional scheduler
- Each user's application runs as a collection of processes (threads)
- User is concerned about the performance of the application
- Need to make scheduling decisions based on process sets

UPPSALA UNIVERSITET

Informationsteknologi

Classifications of Multiprocessor Systems

- Loosely coupled or distributed multiprocessor, or cluster
  - Each processor has its own memory and I/O channels
- Functionally specialized processors
  - Such as I/O processor
  - Controlled by a master processor
- Tightly coupled multiprocessing
  - Processors share main memory
  - Controlled by operating system

UPPSALA UNIVERSITET

Informationsteknologi

Independent Parallelism

- No synchronization among processes
- Each process represents a separate application or job
- Example is a time-sharing system

UPPSALA UNIVERSITET

Informationsteknologi

Coarse and Very Coarse-Grained Parallelism

- Synchronization among processes at a very gross level
- Good for concurrent processes running on a multiprogrammed uniprocessor
- Can by supported on a multiprocessor with little or no change
- An example is a recompilation of several files to build a program

UPPSALA UNIVERSITET

Informationsteknologi

Medium-Grained Parallelism

- Single application is a collection of threads
- Threads usually interact frequently

UPPSALA UNIVERSITET

Informationsteknologi

Fine-Grained Parallelism

- Highly parallel applications
- This is a specialized and fragmented area that has many different approaches

UPPSALA UNIVERSITET

Informationsteknologi

Scheduling on a Multiprocessor

- Assignment of processes to processors
- Use of multiprogramming on individual processors
- Actual dispatching of a process

UPPSALA UNIVERSITET

Informationsteknologi

Assignment of Processes to Processors

- Treat processors as a pooled resource and assign process to processors on demand
- Permanently assign process to a processor
  - Known as group or gang scheduling
  - Dedicate short-term queue for each processor
  - Less overhead
  - Processor could be idle while another processor has a backlog

UPPSALA UNIVERSITET

Informationsteknologi

Assignment of Processes to Processors

- Global queue
  - Schedule to any available processor
- Master/slave architecture
  - Key kernel functions always run on a particular processor
  - Master is responsible for scheduling
  - Slave sends service request to the master
  - Disadvantages
    - Failure of master brings down whole system
    - Master can become a performance bottleneck

UPPSALA UNIVERSITET

Informationsteknologi

Assignment of Processes to Processors

- Peer architecture
  - Operating system can execute on any processor
  - Each processor does self-scheduling
  - Complicates the operating system
    - Make sure two processors do not choose the same process

UPPSALA UNIVERSITET

Informationsteknologi

Process Scheduling

- Single queue for all processes
- Multiple queues are used for priorities
- All queues feed to the common pool of processors

UPPSALA UNIVERSITET

Informationsteknologi

Thread Scheduling

- Executes separate from the rest of the process
- An application can be a set of threads that cooperate and execute concurrently in the same address space
- Threads running on separate processors yield a dramatic gain in performance

Multiprocessor Thread Scheduling

- Load sharing
  - Processes are not assigned to a particular processor
- Gang scheduling
  - A set of related threads is scheduled to run on a set of processors at the same time

UPPSALA UNIVERSITET

Informationsteknologi

Multiprocessor Thread Scheduling

- Dedicated processor assignment
  - Threads are assigned to a specific processor
- Dynamic scheduling
  - Number of threads can be altered during course of execution

UPPSALA UNIVERSITET

Informationsteknologi

Load Sharing

- Load is distributed evenly across the processors
- No centralized scheduler required
- Use global queues

Disadvantages of Load Sharing

- Central queue needs mutual exclusion
  - May be a bottleneck when more than one processor looks for work at the same time
- Preemptive threads are unlikely to resume execution on the same processor
  - Cache use is less efficient
- If all threads are in the global queue, all threads of a program will not gain access to the processors at the same time

UPPSALA UNIVERSITET

Informationsteknologi

Gang Scheduling

- Simultaneous scheduling of threads that make up a single process
- Useful for applications where performance severely degrades when any part of the application is not running
- Threads often need to synchronize with each other

UPPSALA UNIVERSITET

Informationsteknologi

Dedicated Processor Assignment

- When application is scheduled, its threads are assigned to a processor
- Some processors may be idle
- No multiprogramming of processors

UPPSALA UNIVERSITET

Informationsteknologi

Dynamic Scheduling

- Number of threads in a process are altered dynamically by the application
- Operating system adjust the load to improve utilization
  - Assign idle processors
  - New arrivals may be assigned to a processor that is used by a job currently using more than one processor
  - Hold request until processor is available
  - Assign processor a jog in the list that currently has no processors (i.e., to all waiting new arrivals)