TI III: Operating & Communication Systems

Scheduling

Types of Scheduling,
Decision Modes,
Process Priorities,
Scheduling Policies

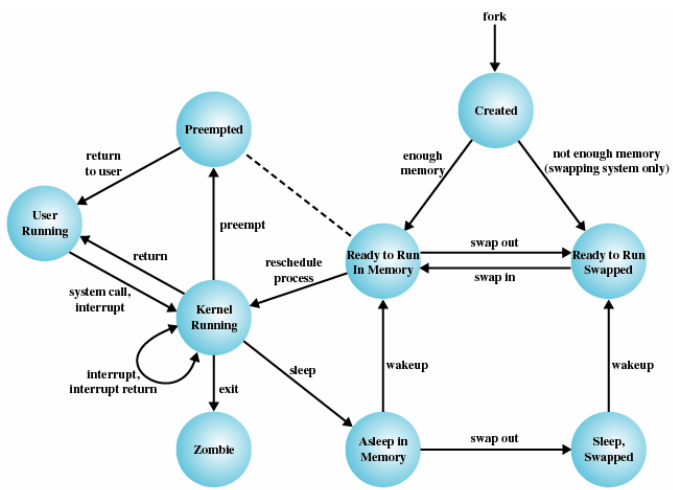


Figure 3.17 UNIX Process State Transition Diagram

Content (1)

Definition and Goals

- Assign processes to be executed by the processor(s)
- More general: assign consumers to resources
- Goals:
  - Response time
  - Throughput
  - Processor efficiency

Types of Scheduling

- Long-term scheduling:
  - Whether to add new process to running queue and execute it, or not
- Medium-term scheduling:
  - Whether to add existing process that is only partially in primary memory
- Short-term scheduling:
  - Which one of fully available processes to run
- I/O scheduling:
  - Which I/O request (of which process) to dispatch to I/O device for handling

Process States and Scheduling

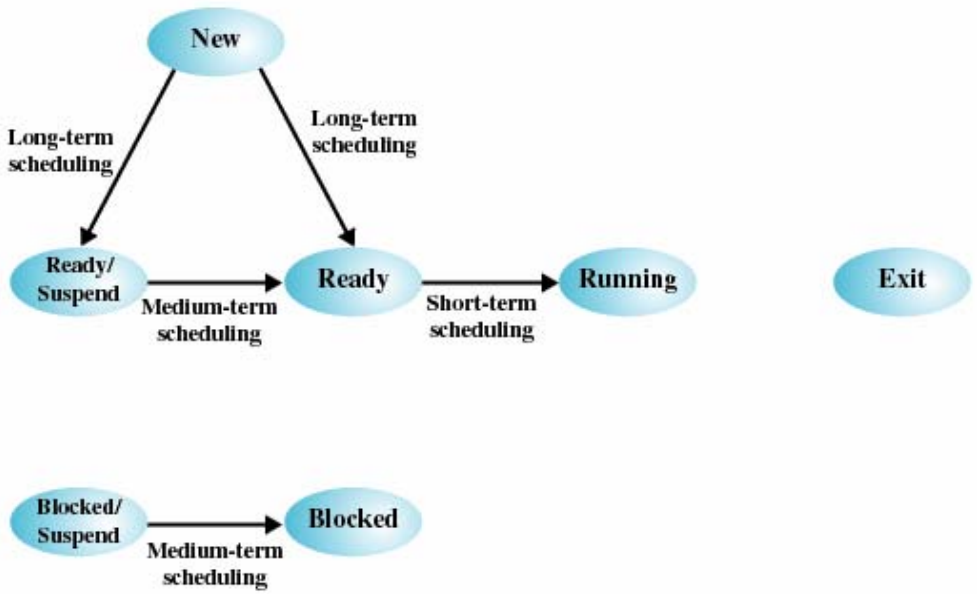


Figure 9.1 Scheduling and Process State Transitions

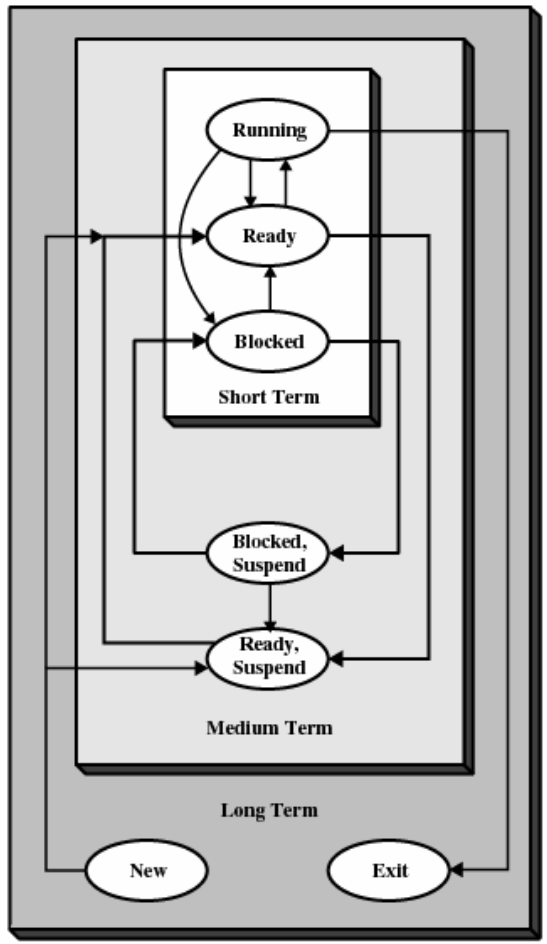


Figure 9.2 Levels of Scheduling

Types of Scheduling in Detail

- Long-term scheduling:
  - Determines which programs are admitted to system for processing
  - Controls degree of multiprogramming
  - The more processes, the smaller percentage of time each process is executed
- Medium-term scheduling:
  - Part of swapping function
  - Based on the need to manage degree of multiprogramming
- Short-term scheduling:
  - Known as the dispatcher
  - Executes most frequently
  - Invoked when an event occurs (clock interrupts, I/O interrupts, operating system calls, signals)

Short-Term Scheduling Criteria

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

Short-Term Scheduling Criteria

	Performance-related	Other
User-oriented	•Turnaround time •Response time •Deadlines	•Predictability
System-oriented	•Throughput •Processor utilization	•Fairness •Enforcing priorities •Balancing resources

Queuing and Scheduling

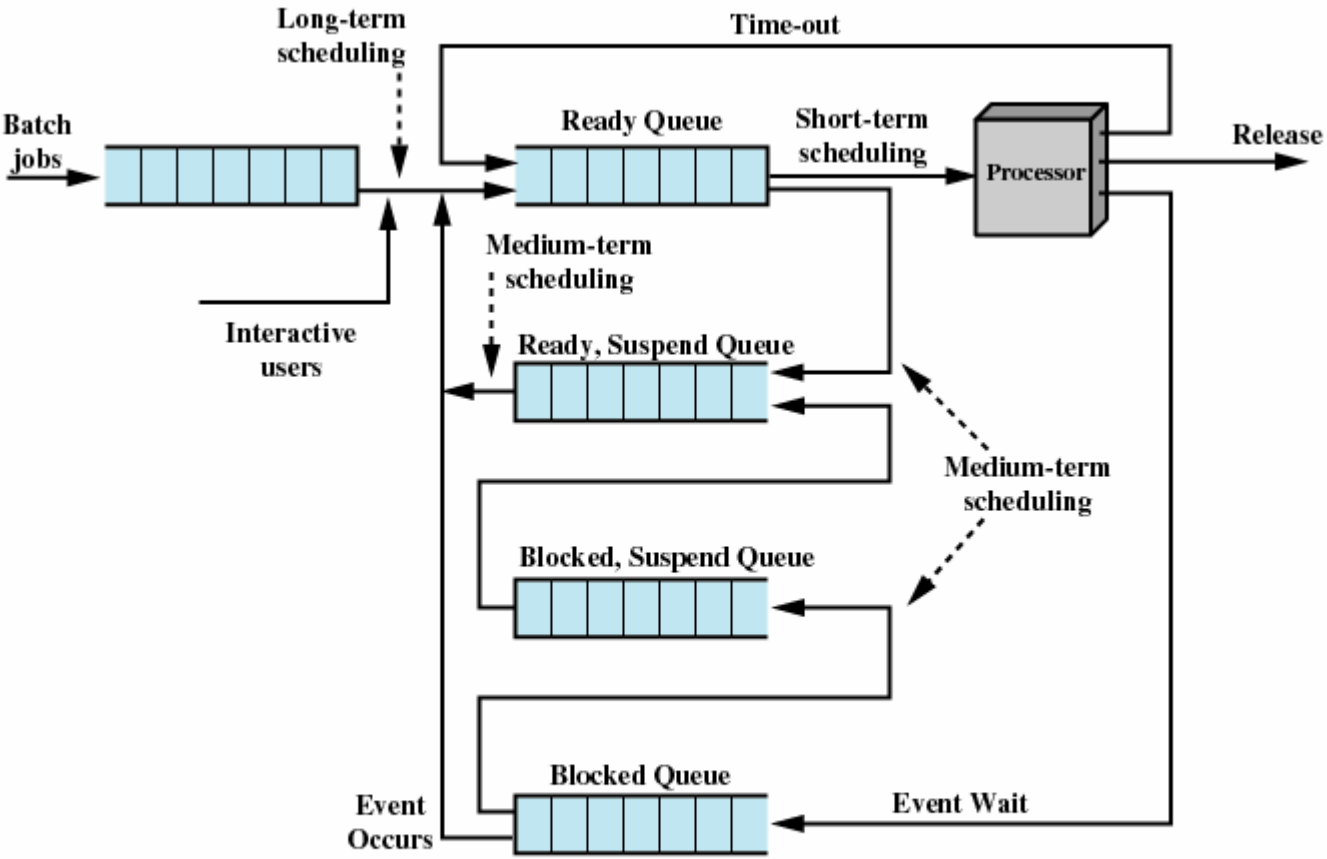


Figure 9.3 Queuing Diagram for Scheduling

Priorities

- Scheduling is controlled by per-process 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

Priority Queuing

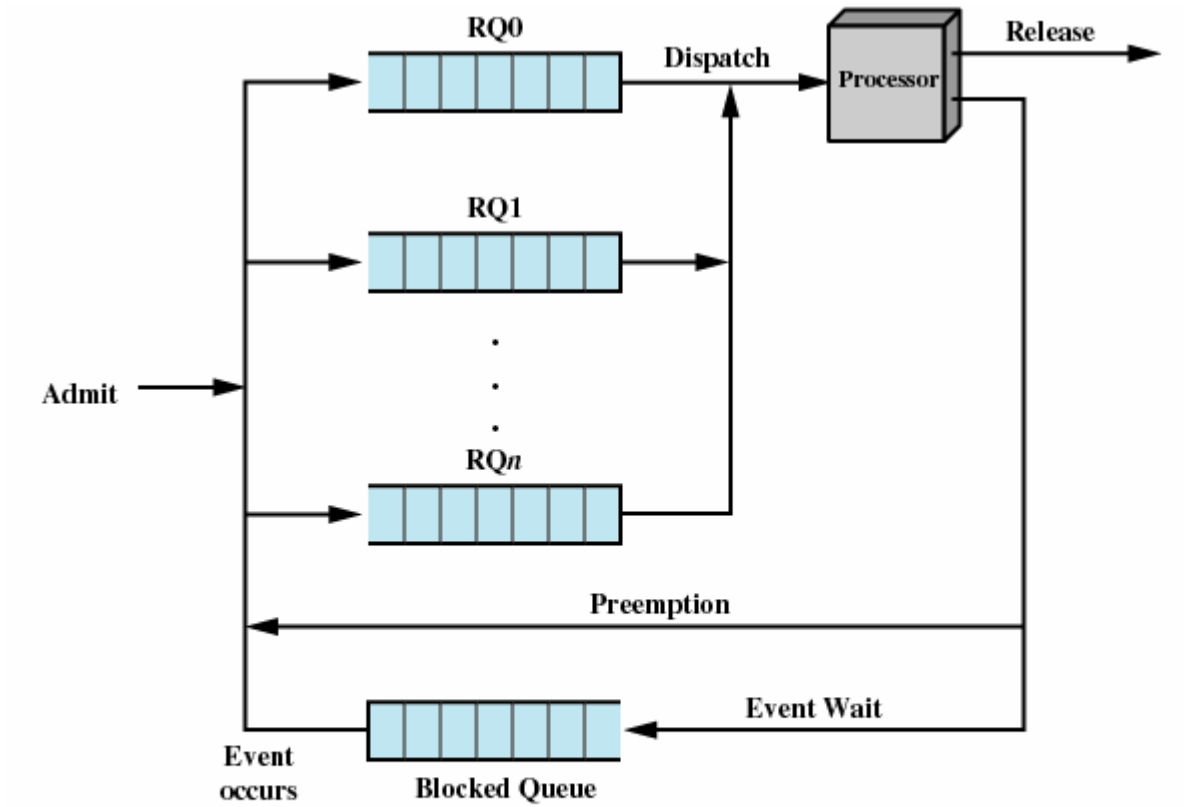
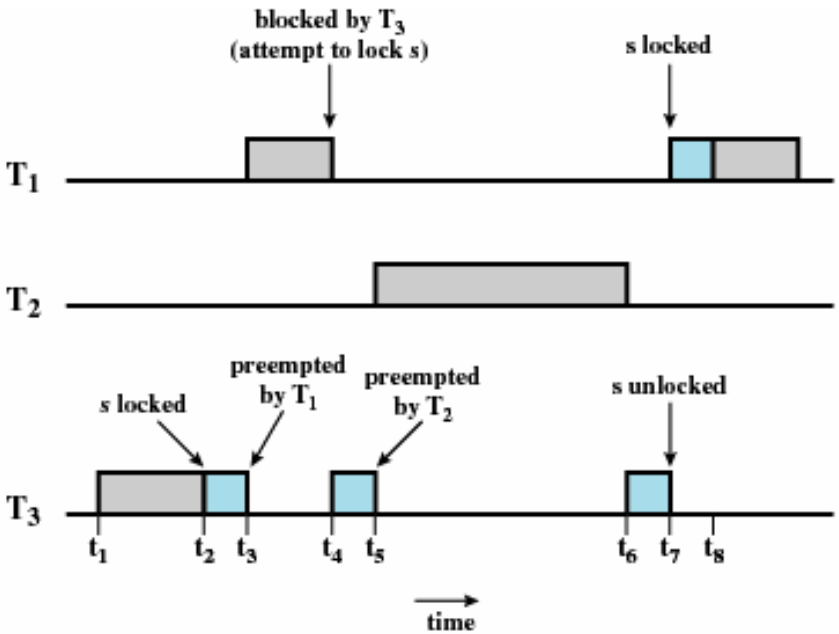


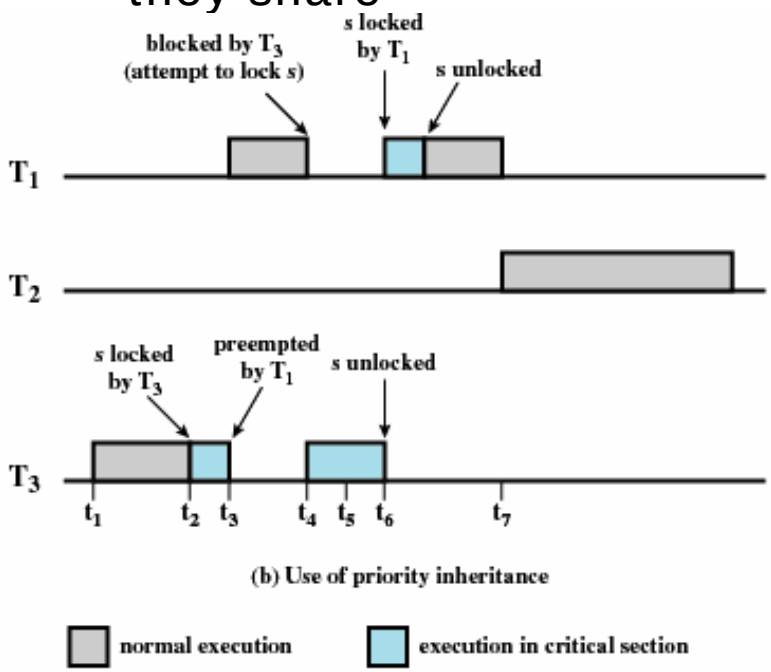
Figure 9.4 Priority Queuing

Priority Inversion and Inheritance

- Problem: Priority Inversion
  - Occurs when circumstances within the system force a higher priority task to wait for a lower priority task
- Solution: Priority Inheritance
  - Lower-priority task inherits priority of any higher priority task pending on a resource they share



(a) Unbounded priority inversion



(b) Use of priority inheritance

Figure 10.9 Priority Inversion

Scheduling Decision Modes

- Non-preemptive
  - Once a process is in 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

Scheduling Algorithms

- First-Come-First-Served (FCFS)
- Round-Robin
- Shortest Process Next
- Shortest Remaining Time
- Highest Response Ratio Next (HRRN)
- Feedback

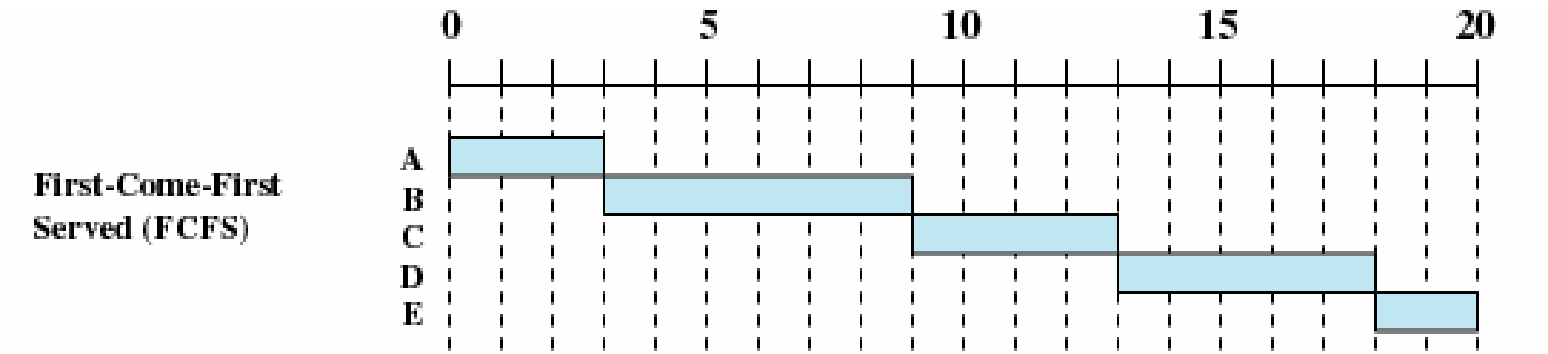
- Example:

Table 9.4 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)

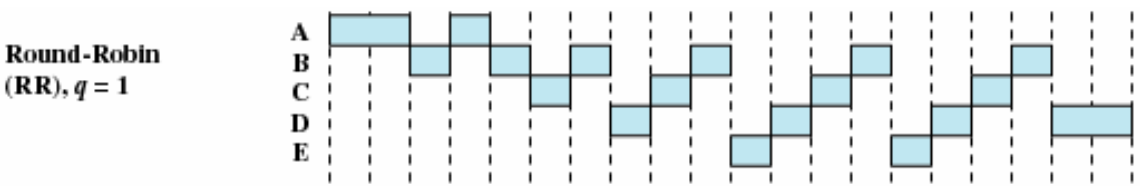
- Each process joins the Ready queue
- When current process ceases to execute, oldest process in the Ready queue is selected



- Short process may have to wait a very long time before it can execute
- Favors CPU-bound processes
  - I/O processes have to wait until CPU-bound process completes

Round-Robin

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

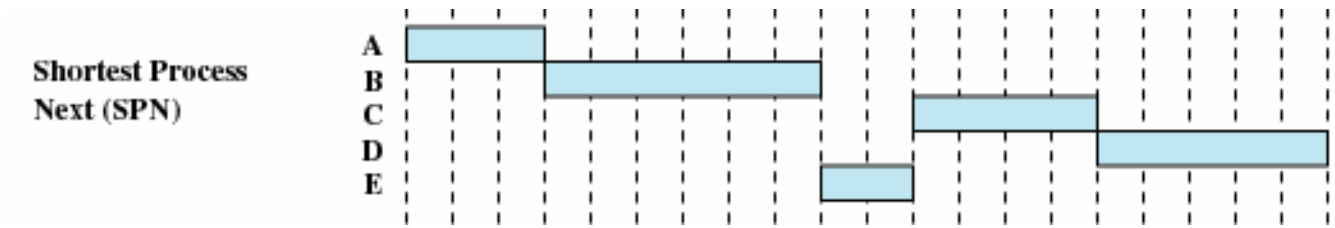


- Clock interrupt is generated at periodic intervals
- When interrupt occurs, currently running process is placed in the read queue, next ready job is selected

➢ Time slicing

Shortest Process Next

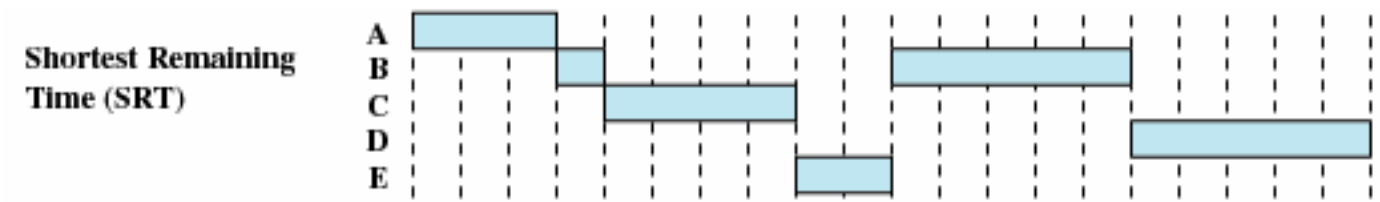
- Process with shortest expected processing time is selected
- Short process jumps ahead of longer processes
- If estimated time for process not correct, the operating system may abort it



- Non-preemptive policy
- Predictability of longer processes is reduced
- Possibility of starvation for longer processes

Shortest Remaining Time

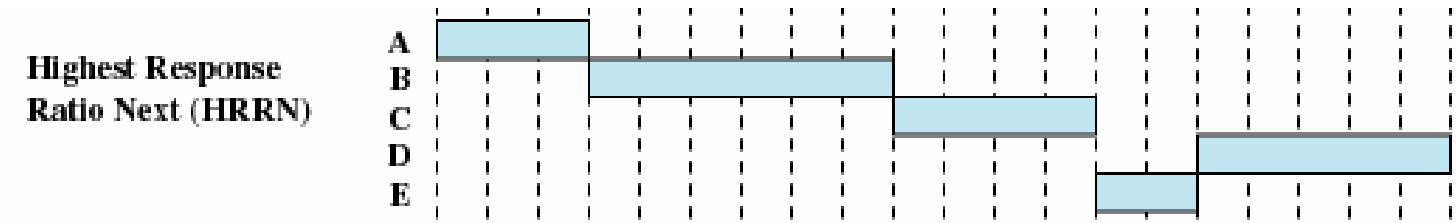
- Preemptive version of shortest process next policy
- Must estimate processing time



Highest Response Ratio Next (HRRN)

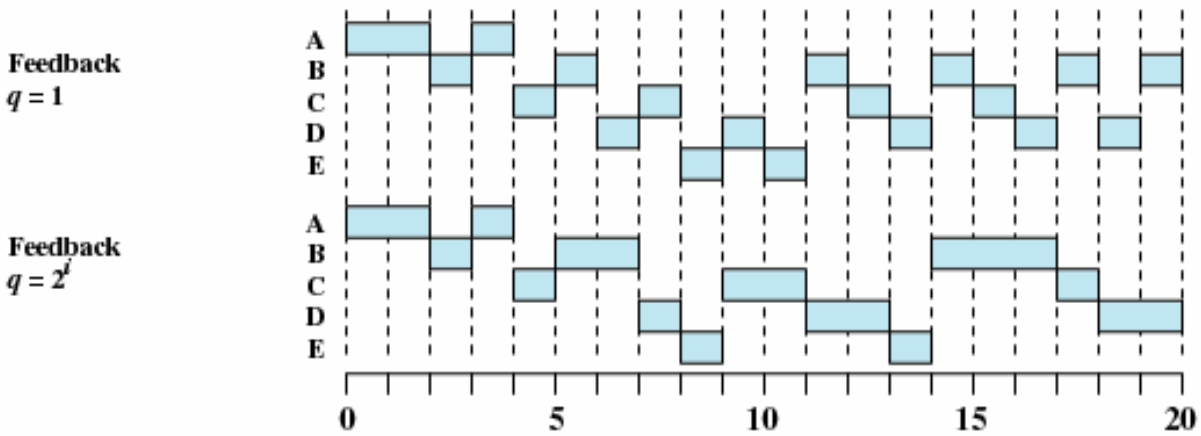
- Choose next process with the greatest ratio

$$\frac{\textit{time spent waiting + expected service time}}{\textit{expected service time}}$$



Feedback

- Multiple queues with different priorities
- Processes start in the queue with highest priority RQ0 and move to queues with lower priority after each time slice
- For fairness, allow longer time slices *q* for queues RQ*q*



- Penalize jobs that have been running longer
- Don't need to know remaining execution time of process

Comparison of Scheduling Policies (1)

Freie Universität Berlin

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

Comparison of Scheduling Policies (2)

Freie Universität Berlin

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

Traditional UNIX Scheduling

- Multilevel feedback using round robin within each of priority queues
- If running process does not block or complete within one second, it is preempted
- Priorities are recomputed once per second
- Base priority divides all processes into fixed bands of priority levels

Multiprocessor Scheduling

- Assignment of processes to processors
  - ➢ Treat processors as a pooled resource and assign process to processors on demand
  - ➢ Permanently assign process to a processor
  - Architectures:
    - Global queue – schedule to any available processor
    - Master / slave – Key kernel functions always run on particular processor, master is responsible for scheduling
    - Peer – Operating system can execute on any processor, each processor does self-scheduling
- Use of multiprogramming on individual processors
- Actual dispatching of processes

Real-Time Scheduling (1)

- Correctness of system depends
  - on logical result of the computation
  - **AND** on time at which the results are produced
- Tasks or processes attempt to control or react to events that take place in outside world
- Examples:
  - Control of laboratory experiments
  - Process control in industrial plants
  - Robotics
  - Air traffic control
  - Telecommunications
  - Military command and control systems
- Real-time applications are not concerned with speed but with completing tasks

Real-Time Scheduling (2)

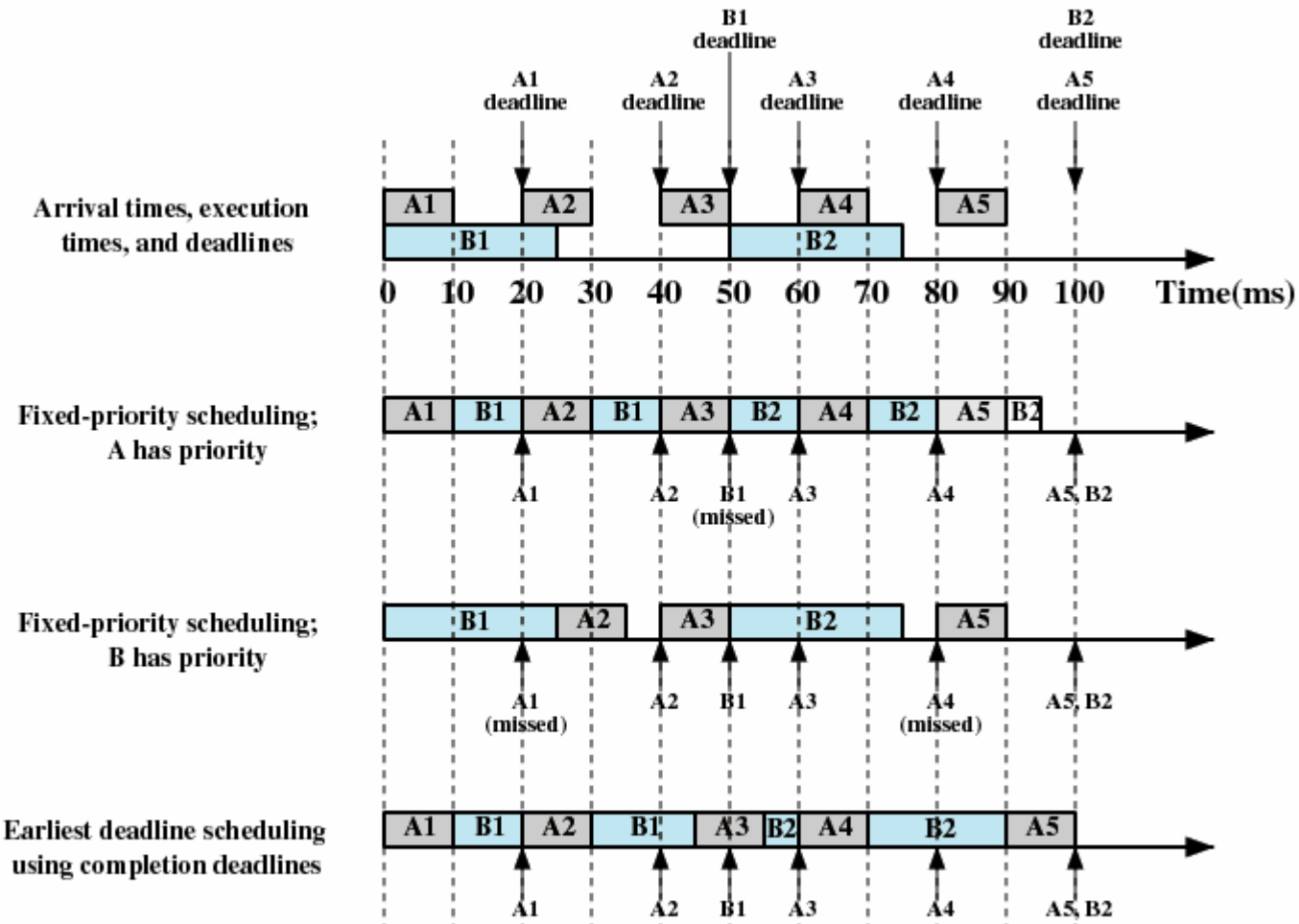


Figure 10.5 Scheduling of Periodic Real-time Tasks with Completion Deadlines (based on Table 10.2)

UNIX SVR4 Scheduling

- Highest preference to real-time processes
- Next-highest to kernel-mode processes
- Lowest preference to other user-mode processes

- Preemptable static priority scheduler
- Introduction of a set of 160 priority levels divided into three priority classes
- Insertion of preemption points

- SVR4 Priority Classes:
  - Real time (159 – 100)
  - Kernel (99 – 60)
  - Time-shared (59-0)

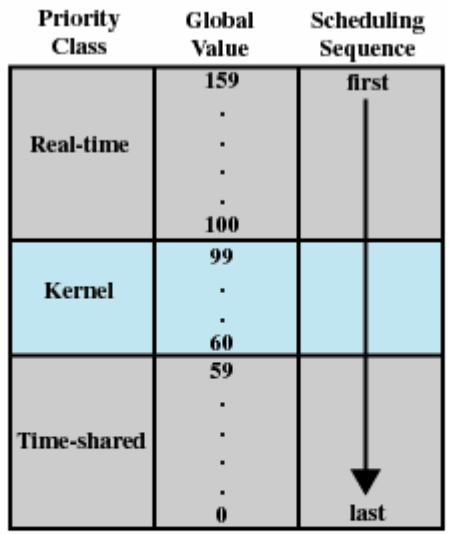


Figure 10.12 SVR4 Priority Classes

Microsoft Windows Scheduling

- Priorities organized into two bands or classes
  - Real time
  - Variable
- Priority-driven preemptive scheduler

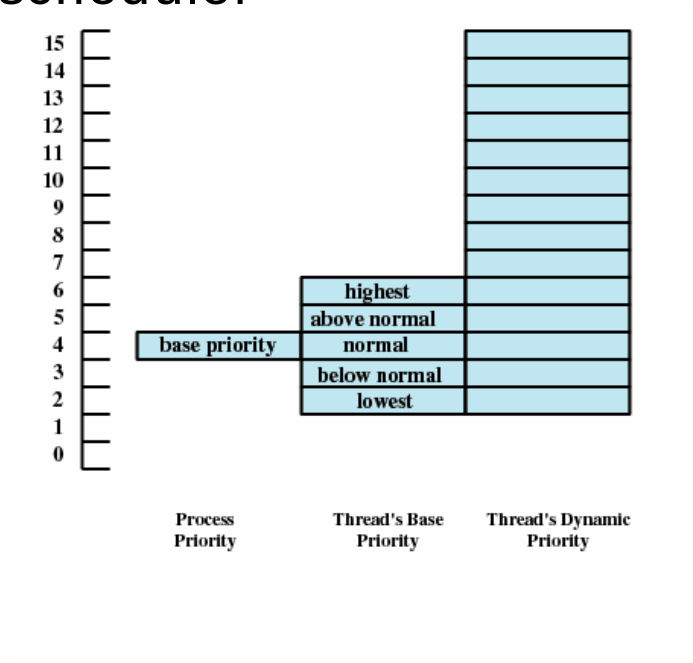


Figure 10.15 Example of Windows Priority Relationship

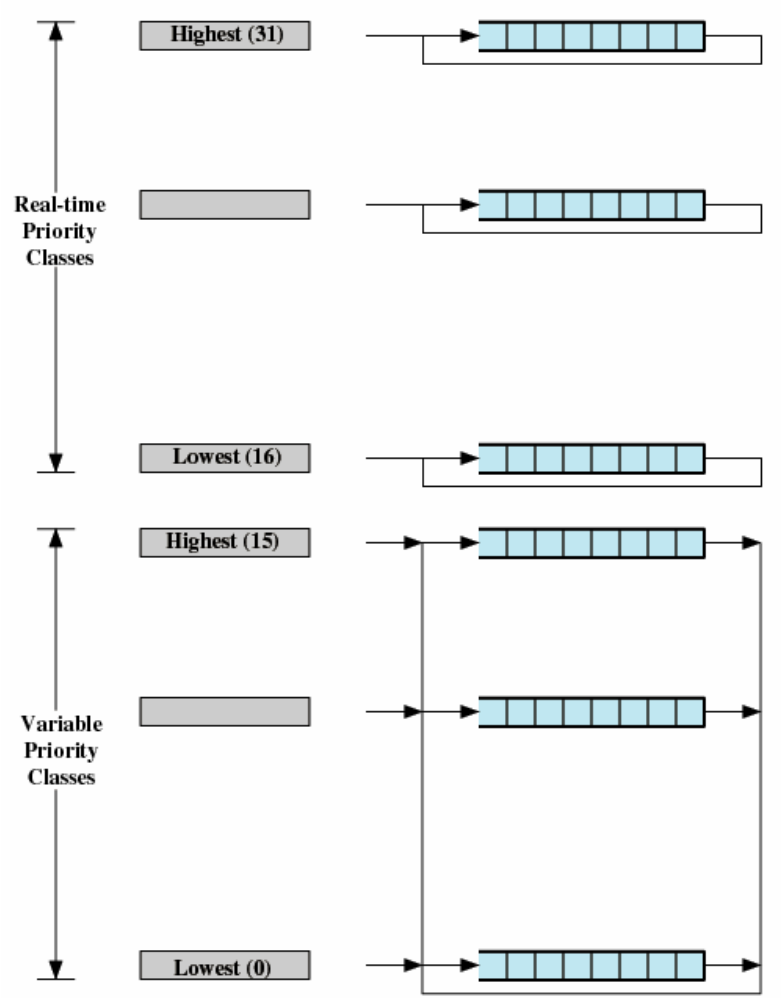


Figure 10.14 Windows Thread Dispatching Priorities

I/O Scheduling (1)

- For a single disk there will be a number of I/O requests
- Seek time is the reason for differences in performance

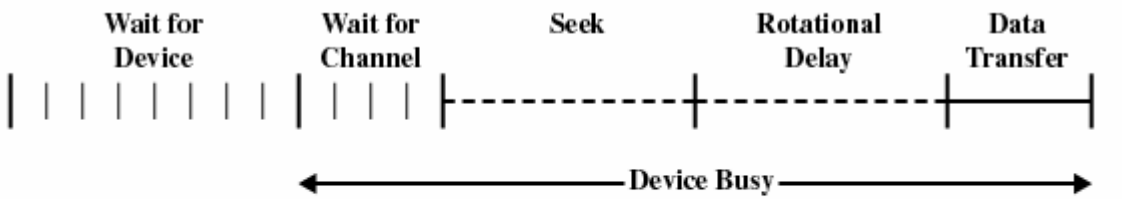


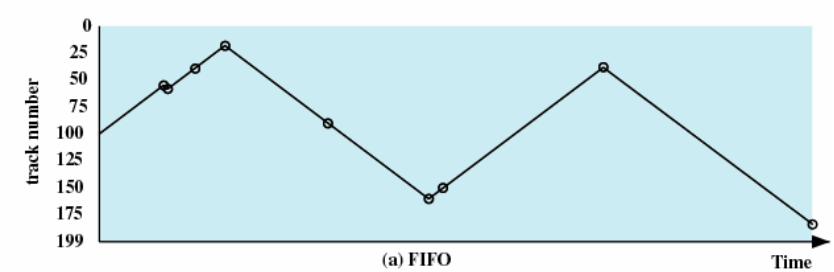
Figure 11.6 Timing of a Disk I/O Transfer

- Access time
  - Sum of seek time and rotational delay
  - The time it takes to get in position to read or write
- Data transfer occurs as the sector moves under the head

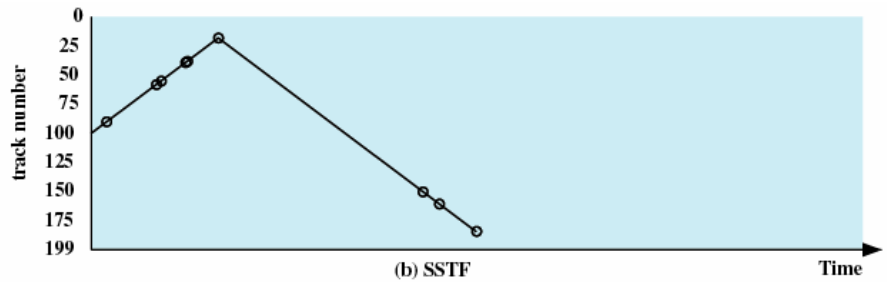
➢ Reorder I/O requests according to current state of disk

I/O Scheduling (2)

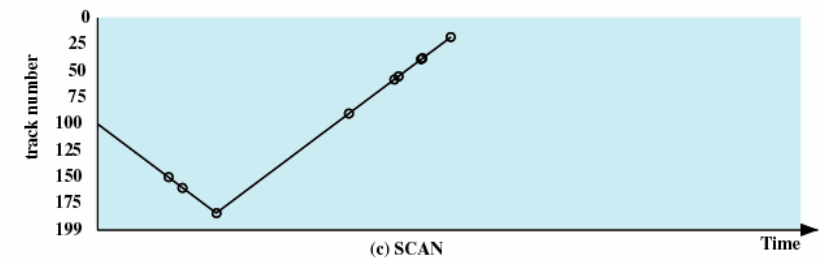
- First-in, first-out (FIFO)



- Shortest Service Time First



- SCAN



- C-SCAN