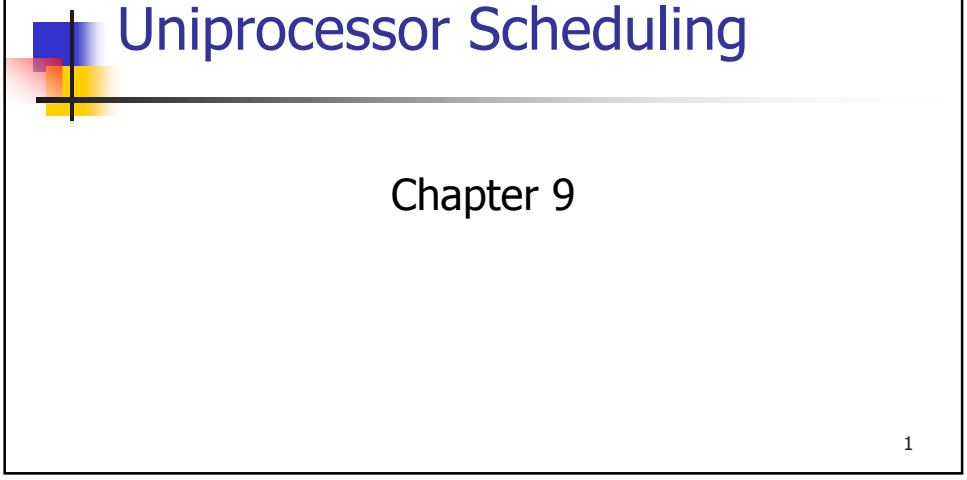
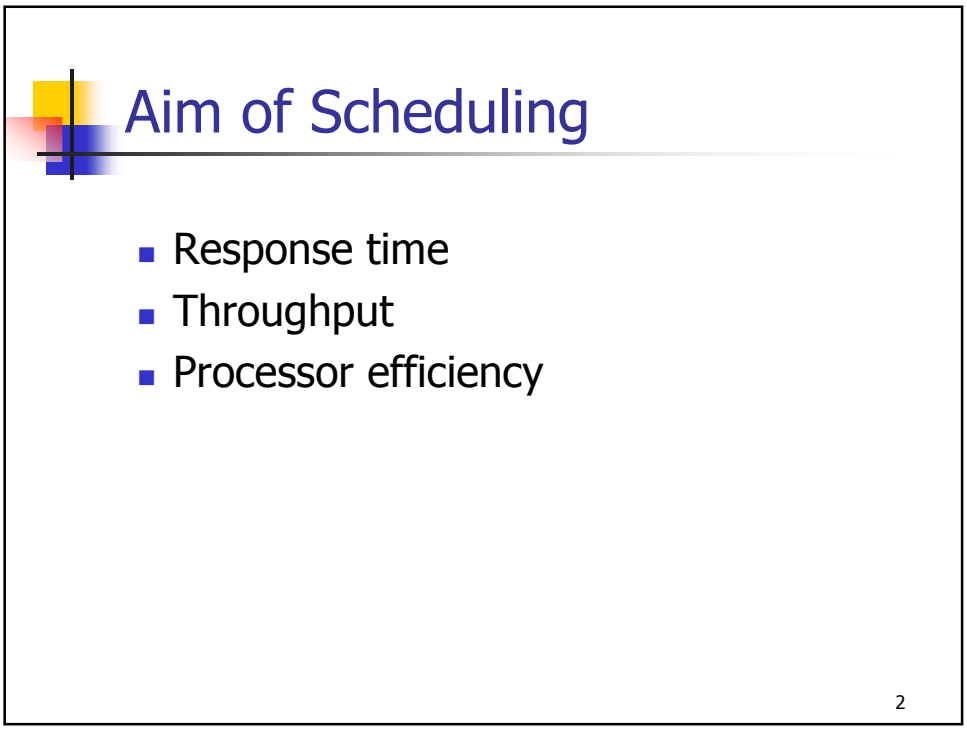

Uniprocessor Scheduling
Chapter 9
1


Aim of Scheduling
Response time
Throughput
Processor efficiency
2

Types of Scheduling

Long-term scheduling	The decision to add to the pool of processes to be executed
Medium-term scheduling	The decision to add to the number of processes that are partially or fully in main memory
Short-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

3

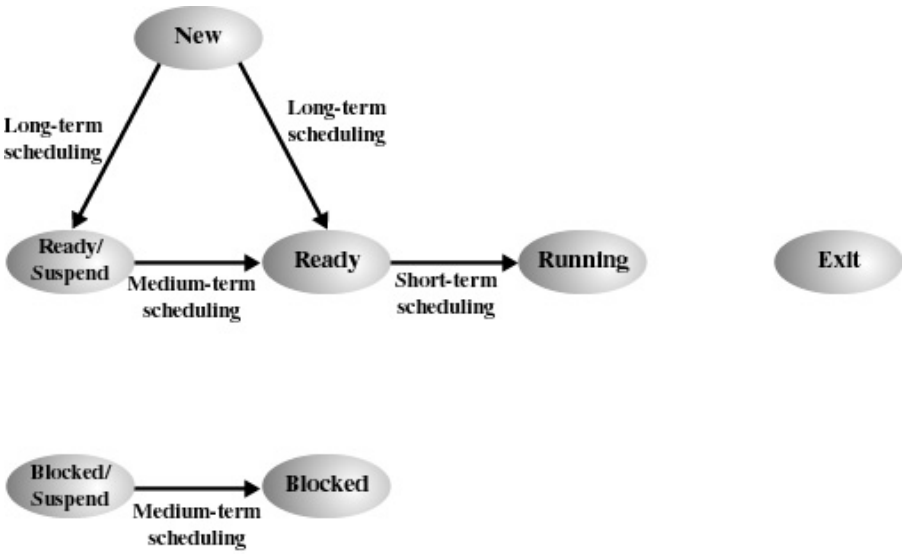


Figure 9.1 Scheduling and Process State Transitions

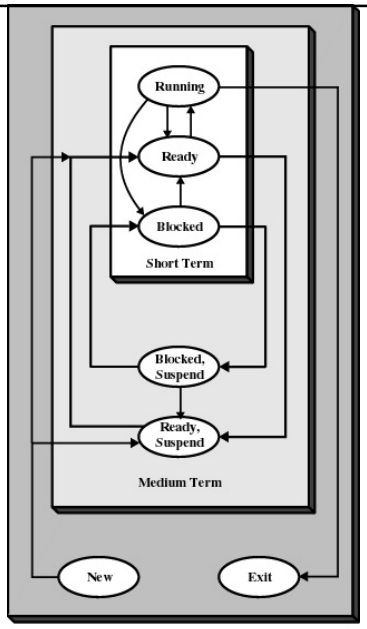


Figure 9.2 Levels of Scheduling

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

6

Medium-Term Scheduling

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

7

Short-Term Scheduling

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

8

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

9

Short-Term Scheduling Criteria

- Performance-related
  - Quantitative
  - Measurable such as response time and throughput
- Not performance related
  - Qualitative
  - Predictability

10

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

11

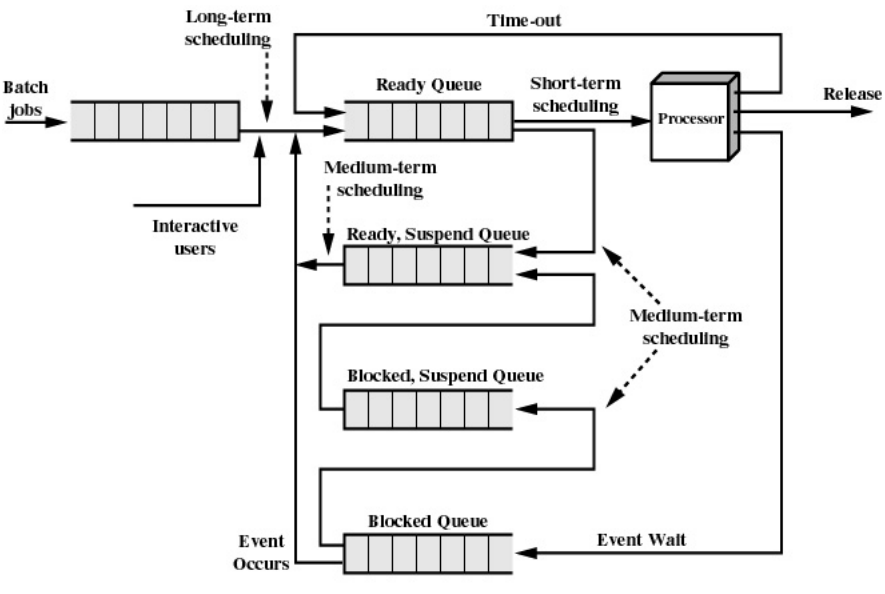


Figure 9.3 Queuing Diagram for Scheduling

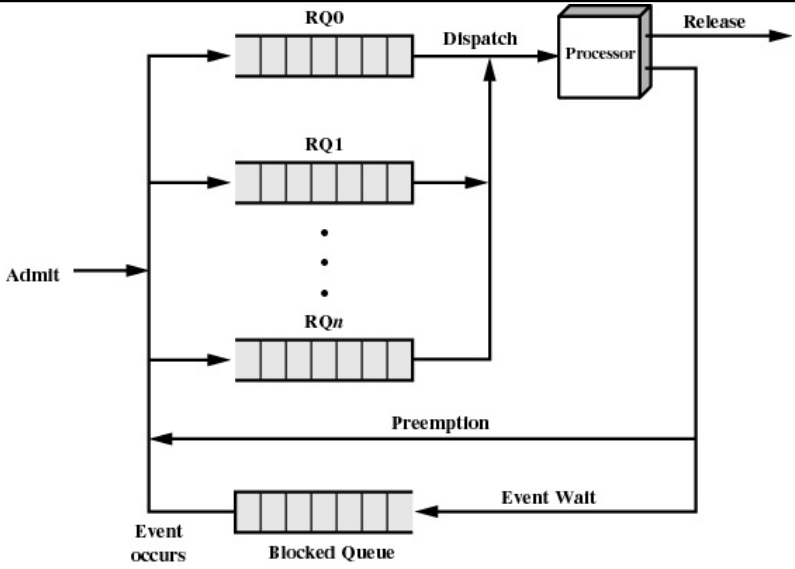


Figure 9.4 Priority Queuing

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

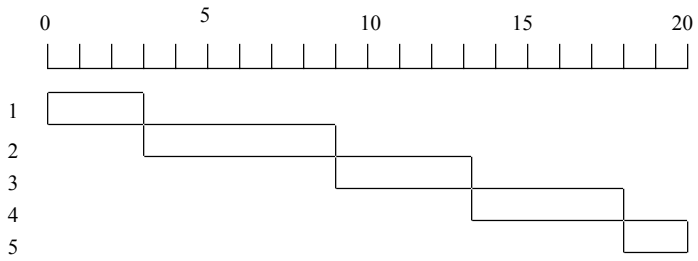
14

Process Scheduling Example

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

15

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

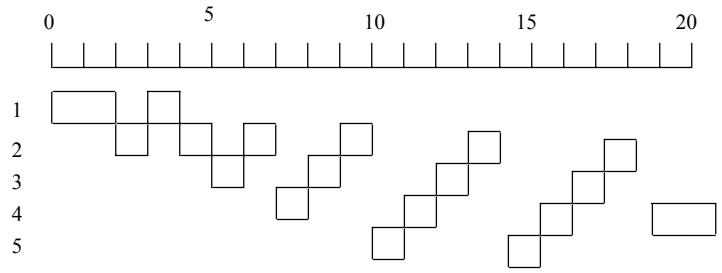
16

First-Come-First-Served (FCFS)

- A 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

17

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

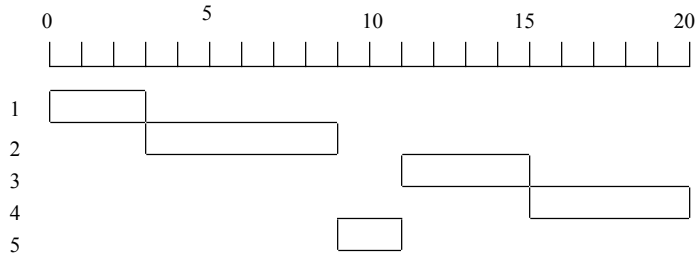
18

Round-Robin

- Clock interrupt is generated at periodic intervals
- When an interrupt occurs, the currently running process is placed in the ready queue
  - Next ready job is selected
- Known as time slicing

19

Shortest Process Next



- Nonpreemptive policy
- Process with shortest expected processing time is selected next
- Short process jumps ahead of longer processes

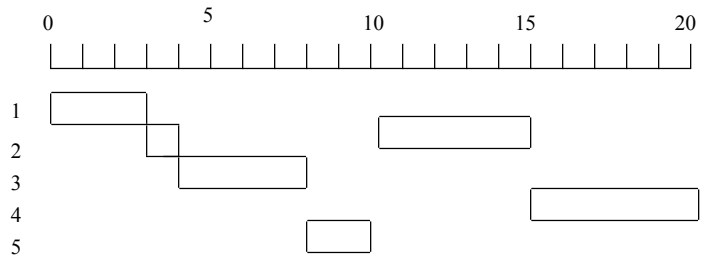
20

Shortest Process Next

- Predictability of longer processes is reduced
- If estimated time for process not correct, the operating system may abort it
- Possibility of starvation for longer processes

21

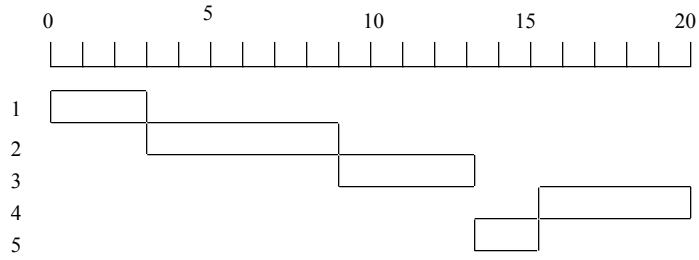
Shortest Remaining Time



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

22

Highest Response Ratio Next (HRRN)

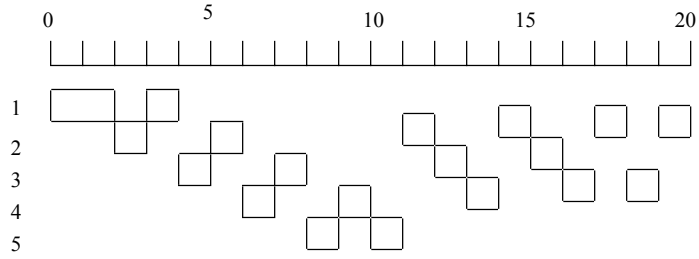


- Choose next process with the lowest ratio

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

23

Feedback



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

24

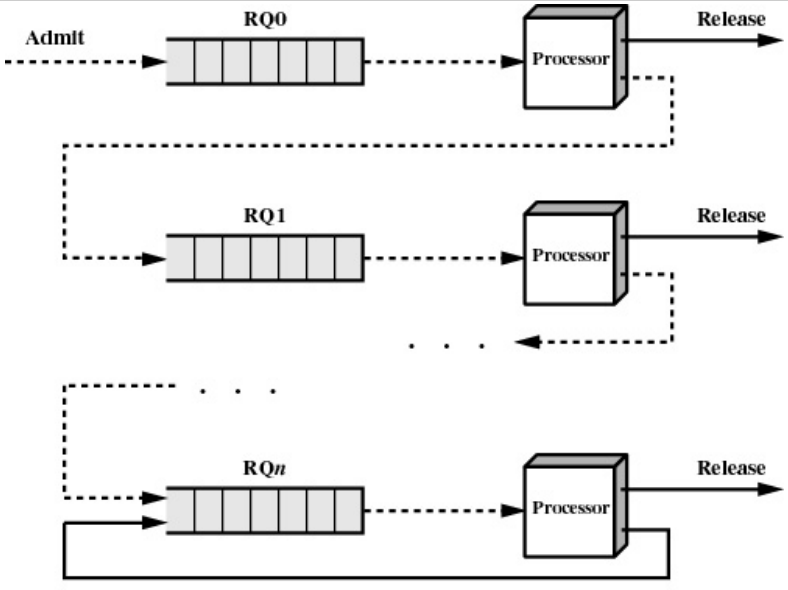
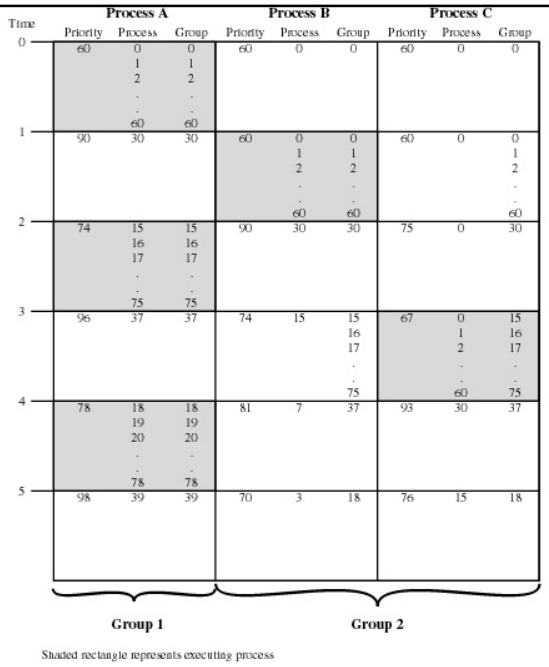


Figure 9.10 Feedback Scheduling

Fair-Share Scheduling

- 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

26



Time	Process A Priority	Process A Process	Process A Group	Process B Priority	Process B Process	Process B Group	Process C Priority	Process C Process	Process C Group
0	60	0	0	60	0	0	60	0	0
		1	1						
		2	2						
		.	.						
		.	.						
		60	60						
1	90	30	30	60	0	0	60	0	0
					1	1			1
					2	2			2
					.	.			.
					.	.			.
					60	60			60
2	74	15	15	90	30	30	75	0	30
		16	16						
		17	17						
		.	.						
		.	.						
		75	75						
3	96	37	37	74	15	15	67	0	15
						16		1	16
						17		2	17
						.		.	.
						.		.	.
						75		60	75
4	78	18	18	81	7	37	93	30	37
		19	19						
		20	20						
		.	.						
		.	.						
		78	78						
5	98	39	39	70	3	18	76	15	18

Group 1 Group 2

Shaded rectangle represents executing process

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

Traditional UNIX Scheduling

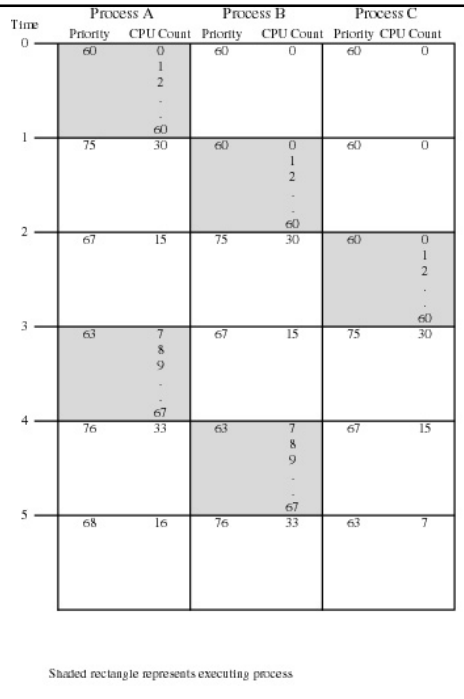
- Multilevel feedback using round robin within each of the priority queues
- Priorities are recomputed once per second
- Base priority divides all processes into fixed bands of priority levels
- Adjustment factor used to keep process in its assigned band

28

Bands

- Decreasing order of priority
  - Swapper
  - Block I/O device control
  - File manipulation
  - Character I/O device control
  - User processes

29



Shaded rectangle represents executing process

Figure 9.17 Example of Traditional UNIX Process Scheduling