Project 2 – solution code

- **Project 2 solution code in files for project 3:**
  - Mutex solution in Synch.c
    - **But this code has several flaws!**
    - **If you copied this, we will know!**
  - Producer/Consumer and Dining Philosophers solution in Proj2Sol.pdf

- **Make sure you study and understand the solutions!**
  - Try to identify errors in your own project 2 code
  - Email your thoughts on these to me and the TA over the next two days for extra credit
  - Important: if the solutions you handed in were copied let me know by email a.s.a.p.

Project 3

- **Part 1: The Sleeping Barber problem**
  - Use semaphores and mutex variables for thread synchronization
  - You decide how to test your code!!
    - **We're not providing the sample output for this one!**

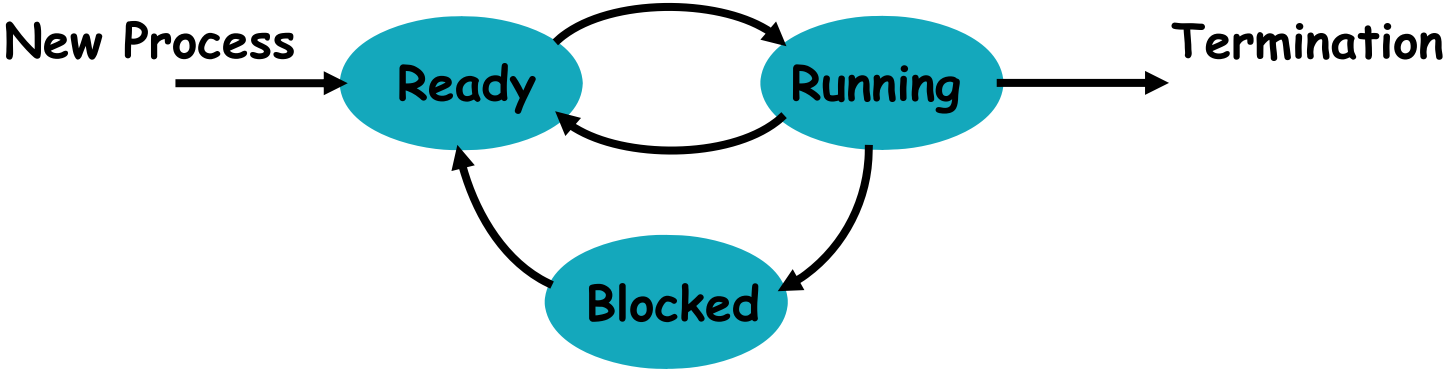
- **Part 2: The Gaming Parlor problem**
  - Use monitors and condition variables for synchronization
  - Avoid deadlock when handling resource requests
  - Sample output given in handout

CS 333
Introduction to Operating Systems

Class 8 - Scheduling

Jonathan Walpole
Computer Science
Portland State University

Process state model



CPU scheduling criteria

- **CPU Utilization** – *how busy is the CPU?*
- **Throughput** – *how many jobs finished/unit time?*
- **Turnaround Time** – *how long from job submission to job termination?*
- **Response Time** – *how long (on average) does it take to get a "response" from a "stimulus"?*
- **Missed deadlines** – *were any deadlines missed?*

Scheduler options

- **Priorities**
  - May use priorities to determine who runs next
  - Dynamic vs. Static algorithms
    - **Dynamically alter the priority of the tasks while they are in the system (possibly with feedback)**
    - **Static algorithms typically assign a fixed priority when the job is initially started.**
- **Preemptive vs. Nonpreemptive**
  - Preemptive systems allow the task to be interrupted at any time so that the O.S. can take over again.

Scheduling policies

- First-Come, First Served (FIFO)
- Shortest Job First (non-preemptive)
- Shortest Job First (with preemption)
- Round-Robin Scheduling
- Priority Scheduling
- Real-Time Scheduling

First-Come, First-Served (FIFO)

- Start jobs in the order they arrive (FIFO queue)
- Run each job until completion

First-Come, First-Served (FIFO)

- Start jobs in the order they arrive (FIFO queue)
- Run each job until completion

	Arrival	Processing
Process	Time	Time
□ 1	0	3
▨ 2	2	6
🟥 3	4	4
🟦 4	6	5
🟩 5	8	2

First-Come, First-Served (FIFO)

- Start jobs in the order they arrive (FIFO queue)
- Run each job until completion

	Arrival	Processing		Turnaround
Process	Time	Time	Delay	Time
░ 1	0	3		
▨ 2	2	6		
🟥 3	4	4		
🟦 4	6	5		
🟩 5	8	2		

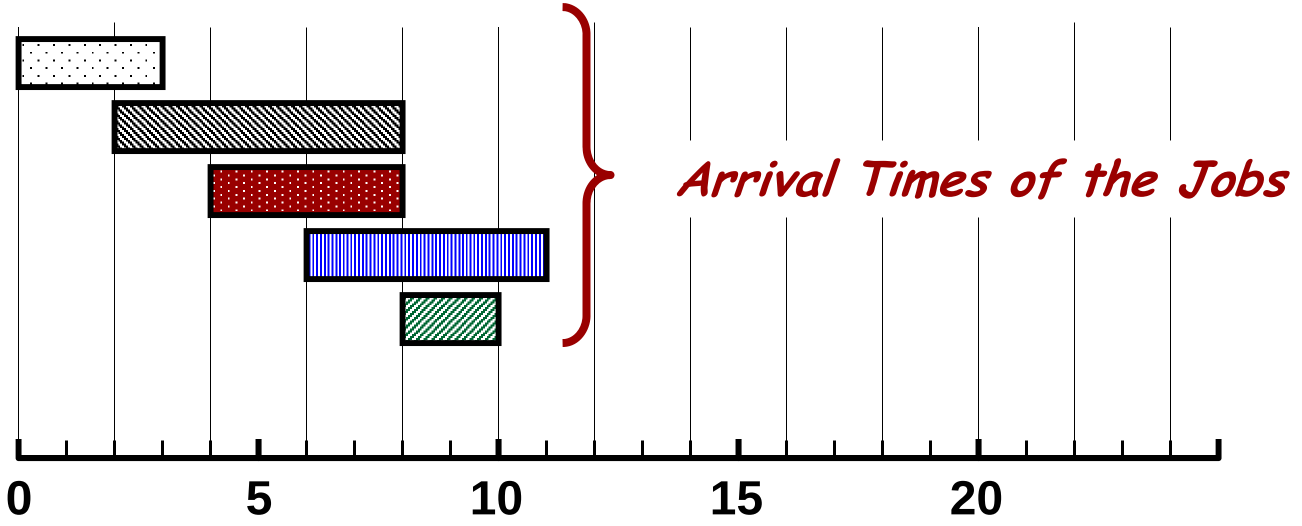
Total time taken, from submission to completion

First-Come, First-Served (FIFO)

- Start jobs in the order they arrive (FIFO queue)
- Run each job until completion

Total time taken, from submission to completion

	Arrival	Processing		Turnaround
Process	Time	Time	Delay	Time
⬚ 1	0	3		
▨ 2	2	6		
🟥 3	4	4		
🟦 4	6	5		
🟩 5	8	2		

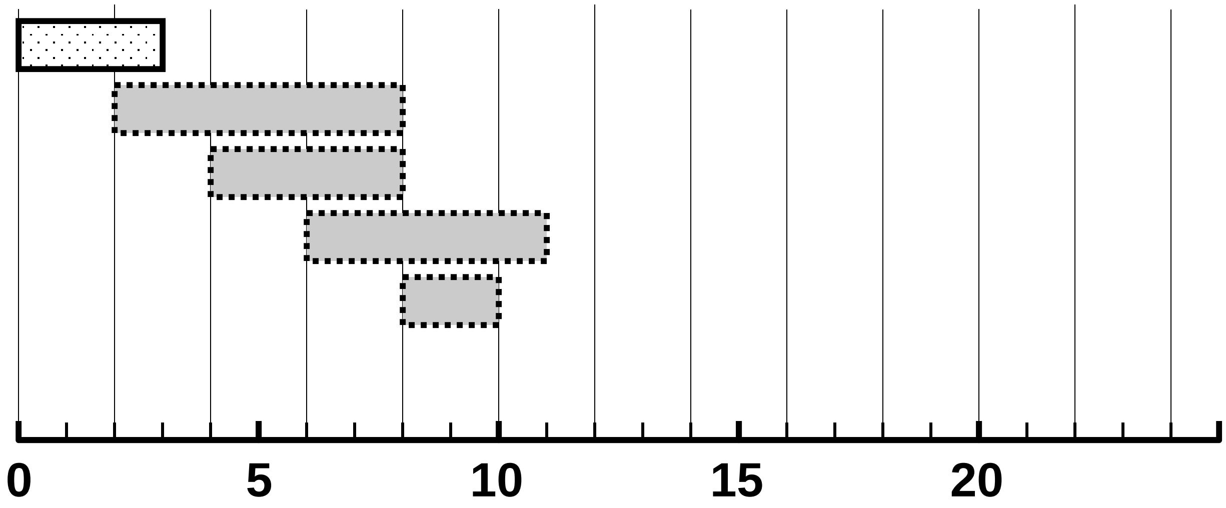


First-Come, First-Served (FIFO)

- Start jobs in the order they arrive (FIFO queue)
- Run each job until completion

Total time taken, from submission to completion

	Arrival	Processing		Turnaround
Process	Time	Time	Delay	Time
⬚ 1	0	3		
▨ 2	2	6		
🟥 3	4	4		
🟦 4	6	5		
🟩 5	8	2		

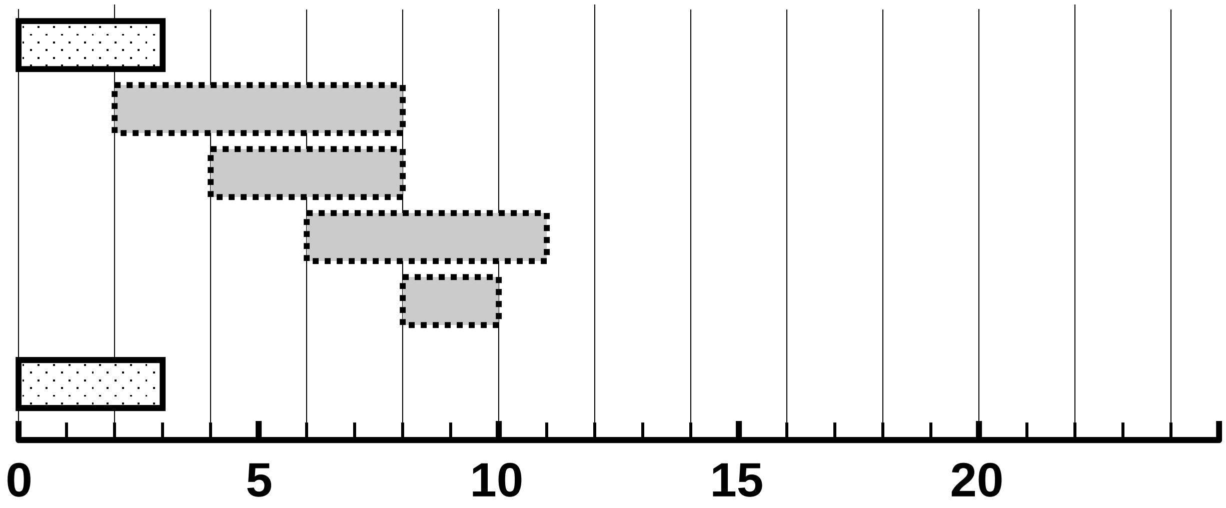


First-Come, First-Served (FIFO)

- Start jobs in the order they arrive (FIFO queue)
- Run each job until completion

Total time taken, from submission to completion

Process		Arrival Time	Processing Time	Delay	Turnaround Time
⬚	1	0	3		
▨	2	2	6		
🟥	3	4	4		
🟦	4	6	5		
🟩	5	8	2		

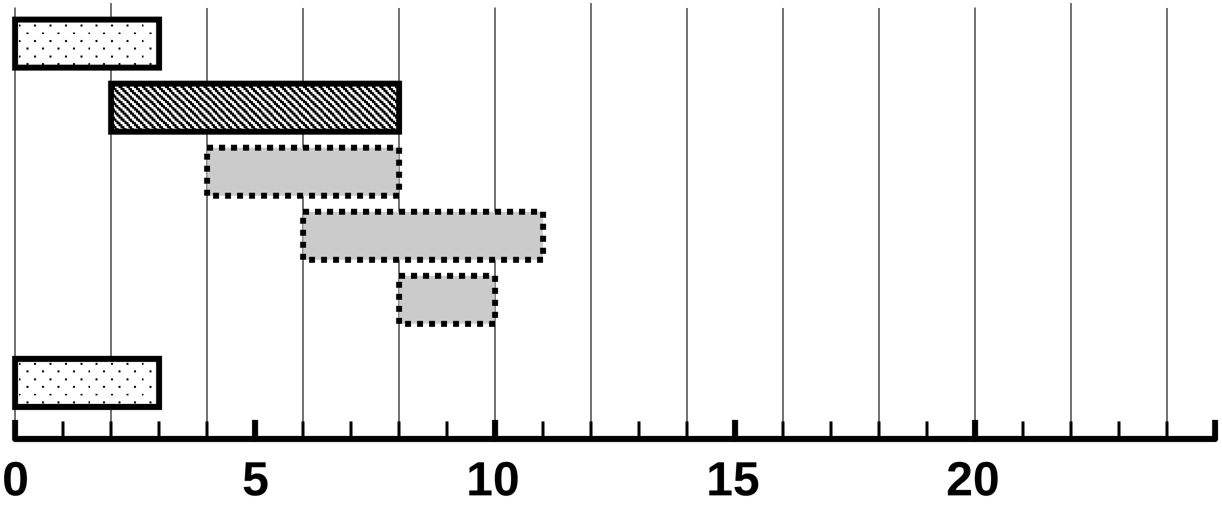


First-Come, First-Served (FIFO)

- Start jobs in the order they arrive (FIFO queue)
- Run each job until completion

Total time taken, from submission to completion

	Arrival	Processing		Turnaround
Process	Time	Time	Delay	Time
░ 1	0	3		
▨ 2	2	6		
🟥 3	4	4		
🟦 4	6	5		
🟩 5	8	2		

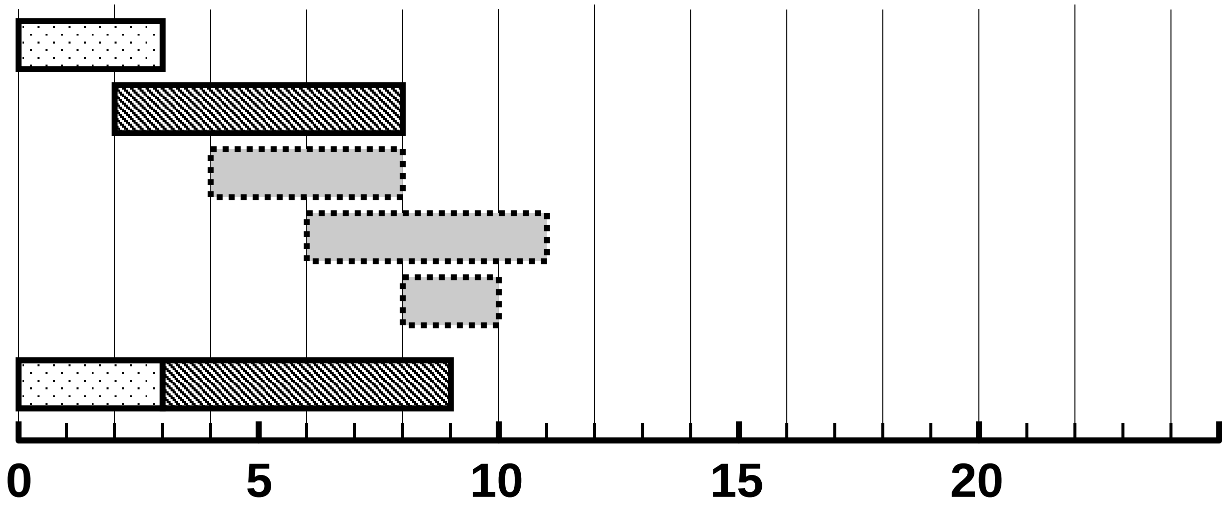


First-Come, First-Served (FIFO)

- Start jobs in the order they arrive (FIFO queue)
- Run each job until completion

Total time taken, from submission to completion

Process	Arrival Time	Processing Time	Delay	Turnaround Time
⬚ 1	0	3		
▨ 2	2	6		
🟥 3	4	4		
🟦 4	6	5		
🟩 5	8	2		

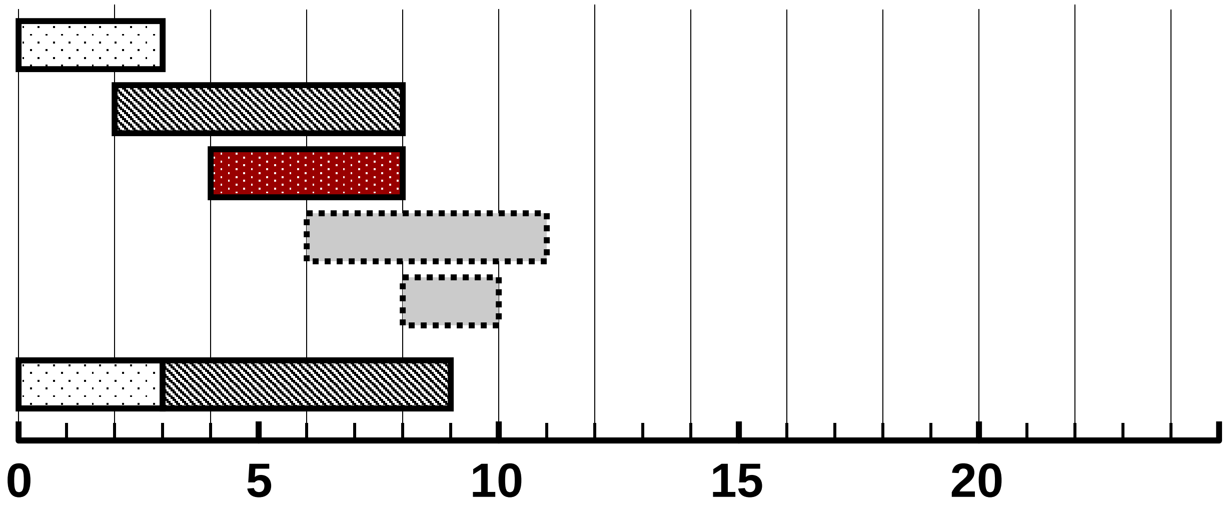


First-Come, First-Served (FIFO)

- Start jobs in the order they arrive (FIFO queue)
- Run each job until completion

Total time taken, from submission to completion

	Process	Arrival	Processing		Turnaround
		Time	Time	Delay	Time
░	1	0	3		
▨	2	2	6		
🟥	3	4	4		
▥	4	6	5		
🟩	5	8	2		

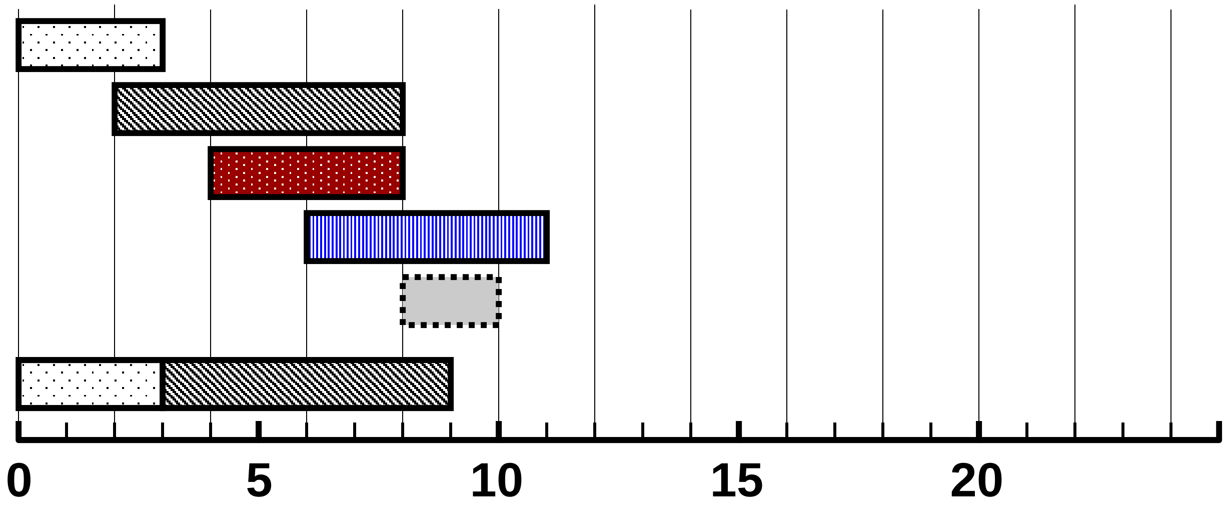


First-Come, First-Served (FIFO)

- Start jobs in the order they arrive (FIFO queue)
- Run each job until completion

Total time taken, from submission to completion

	Arrival	Processing		Turnaround
Process	Time	Time	Delay	Time
⬚ 1	0	3		
▨ 2	2	6		
🟥 3	4	4		
🟦 4	6	5		
🟩 5	8	2		

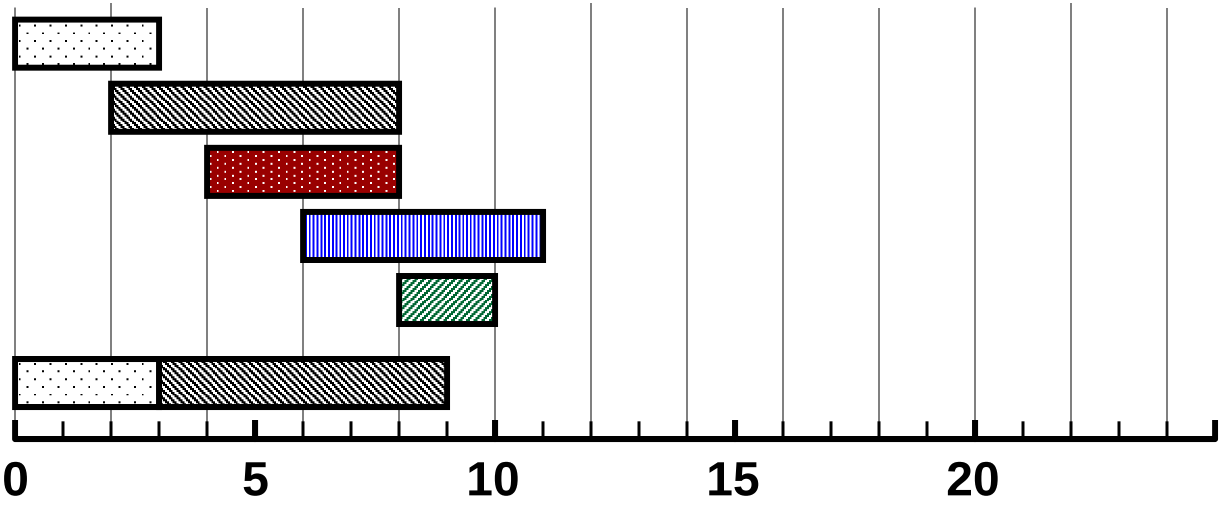


First-Come, First-Served (FIFO)

- Start jobs in the order they arrive (FIFO queue)
- Run each job until completion

Total time taken, from submission to completion

Process	Arrival Time	Processing Time	Delay	Turnaround Time
░ 1	0	3		
▨ 2	2	6		
🟥 3	4	4		
🟦 4	6	5		
🟩 5	8	2		

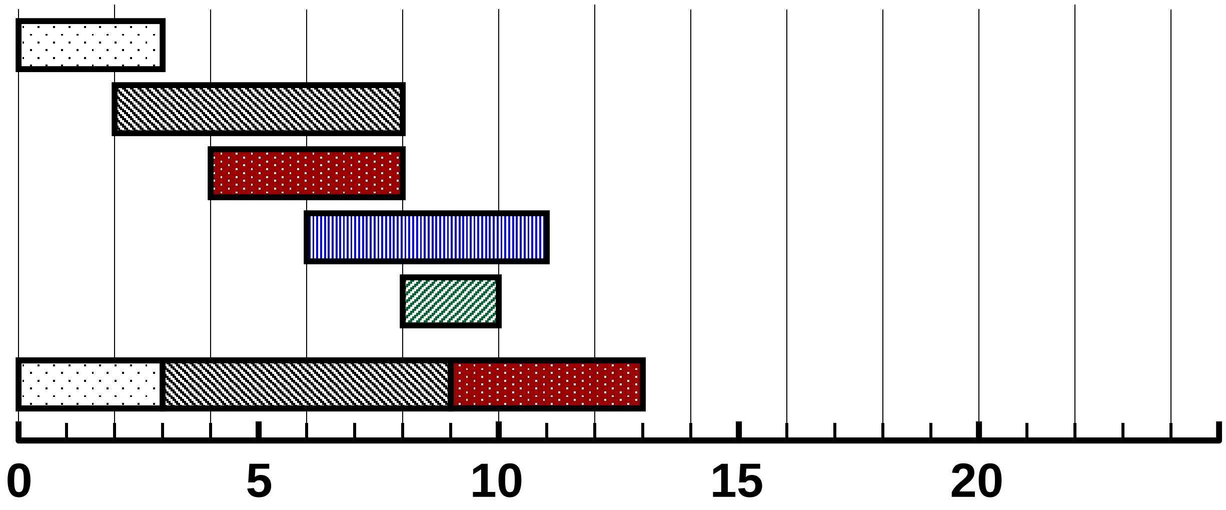


First-Come, First-Served (FIFO)

- Start jobs in the order they arrive (FIFO queue)
- Run each job until completion

Total time taken, from submission to completion

	Arrival	Processing		Turnaround
Process	Time	Time	Delay	Time
⬚ 1	0	3		
▨ 2	2	6		
🟥 3	4	4		
🟦 4	6	5		
🟩 5	8	2		

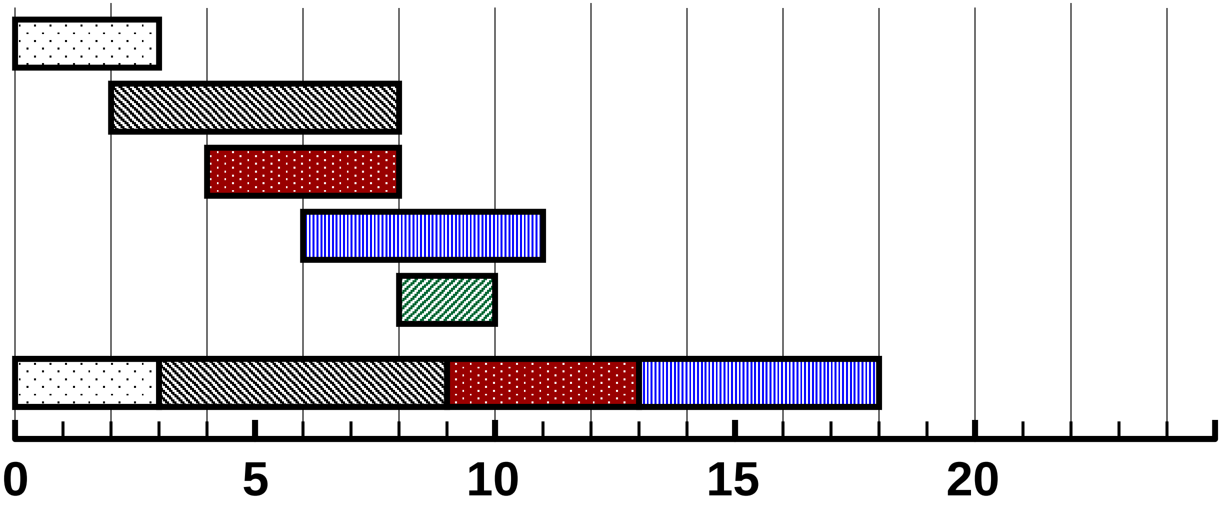


First-Come, First-Served (FIFO)

- Start jobs in the order they arrive (FIFO queue)
- Run each job until completion

Total time taken, from submission to completion

Process	Arrival Time	Processing Time	Delay	Turnaround Time
⬚ 1	0	3		
▨ 2	2	6		
🟥 3	4	4		
🟦 4	6	5		
🟩 5	8	2		

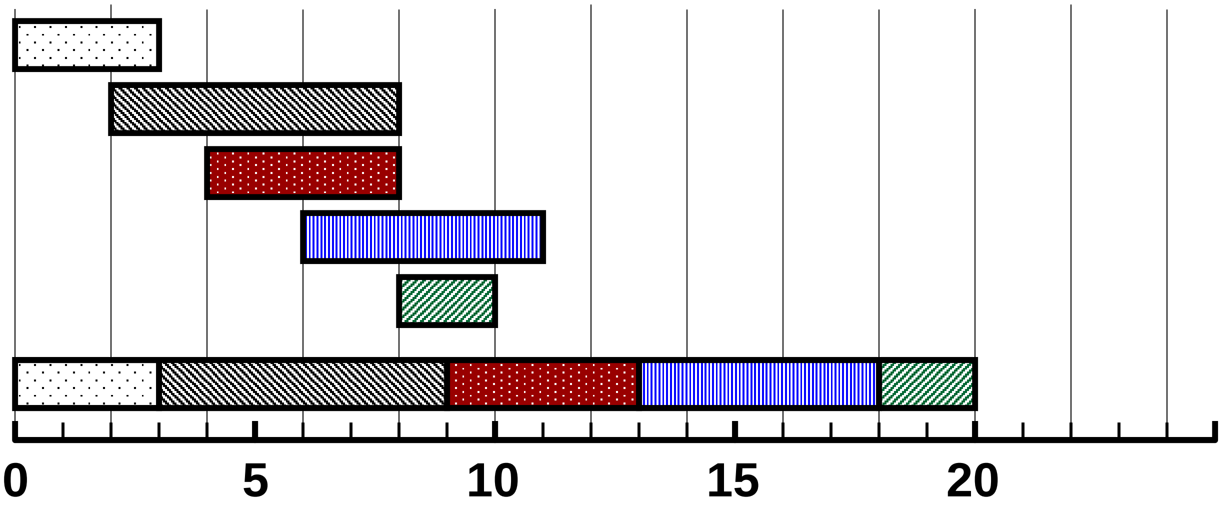


First-Come, First-Served (FIFO)

- Start jobs in the order they arrive (FIFO queue)
- Run each job until completion

Total time taken, from submission to completion

		Arrival	Processing		Turnaround
	Process	Time	Time	Delay	Time
░	1	0	3		
▨	2	2	6		
🟥	3	4	4		
🟦	4	6	5		
🟩	5	8	2		

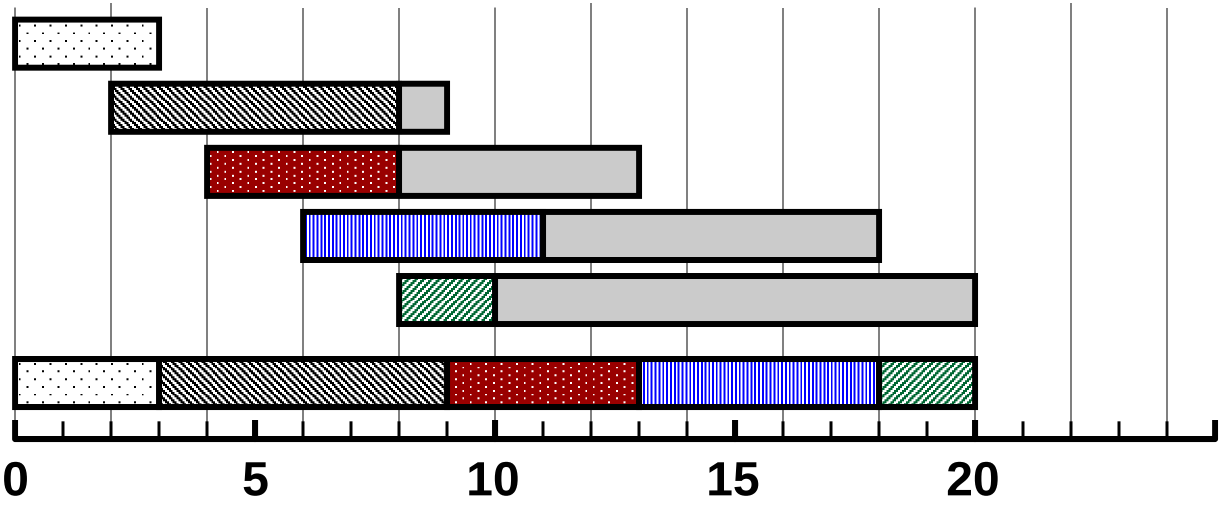


First-Come, First-Served (FIFO)

- Start jobs in the order they arrive (FIFO queue)
- Run each job until completion

Total time taken, from submission to completion

Process		Arrival Time	Processing Time	Delay	Turnaround Time
⬚	1	0	3		
▨	2	2	6		
🟥	3	4	4		
🟦	4	6	5		
🟩	5	8	2		

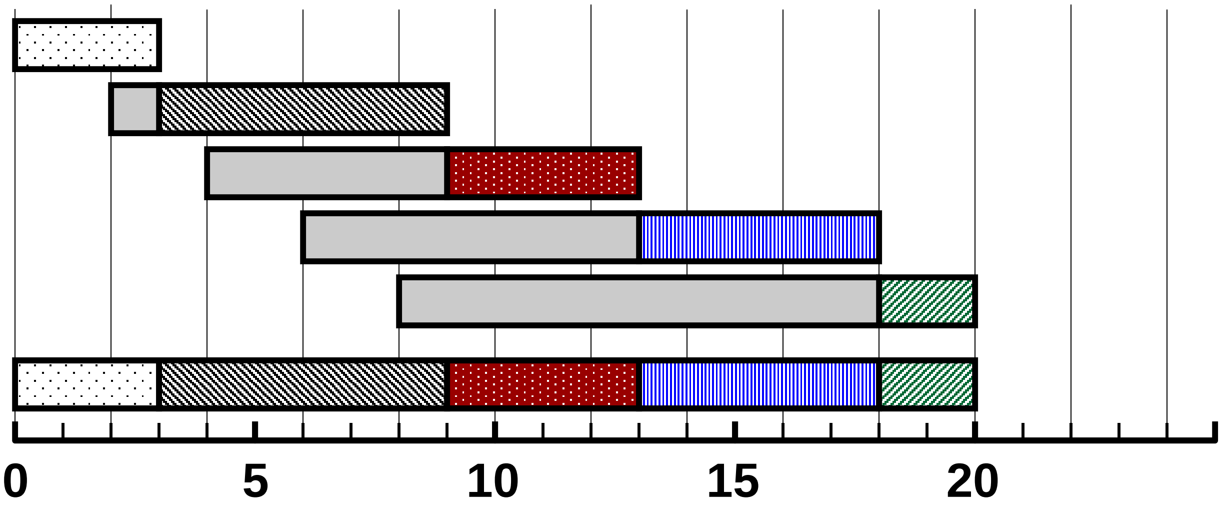


First-Come, First-Served (FIFO)

- Start jobs in the order they arrive (FIFO queue)
- Run each job until completion

Total time taken, from submission to completion

	Arrival	Processing		Turnaround
Process	Time	Time	Delay	Time
⬚ 1	0	3		
▨ 2	2	6		
🟥 3	4	4		
🟦 4	6	5		
🟩 5	8	2		

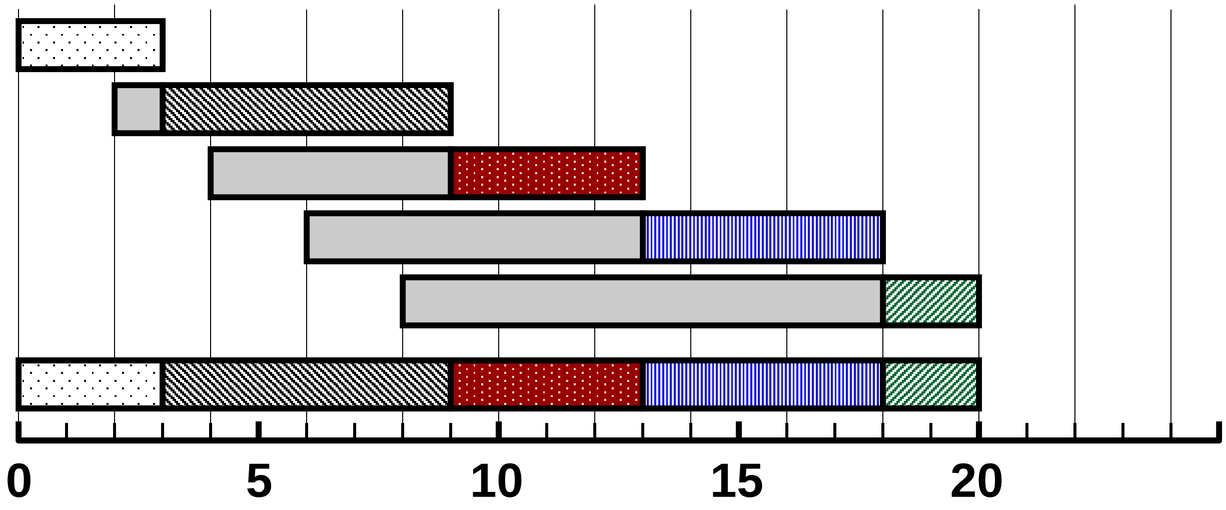


First-Come, First-Served (FIFO)

- Start jobs in the order they arrive (FIFO queue)
- Run each job until completion

Total time taken, from submission to completion

Process		Arrival Time	Processing Time	Delay	Turnaround Time
⬚	1	0	3	0	
▨	2	2	6	1	
🟥	3	4	4	5	
🟦	4	6	5	7	
🟩	5	8	2	10	

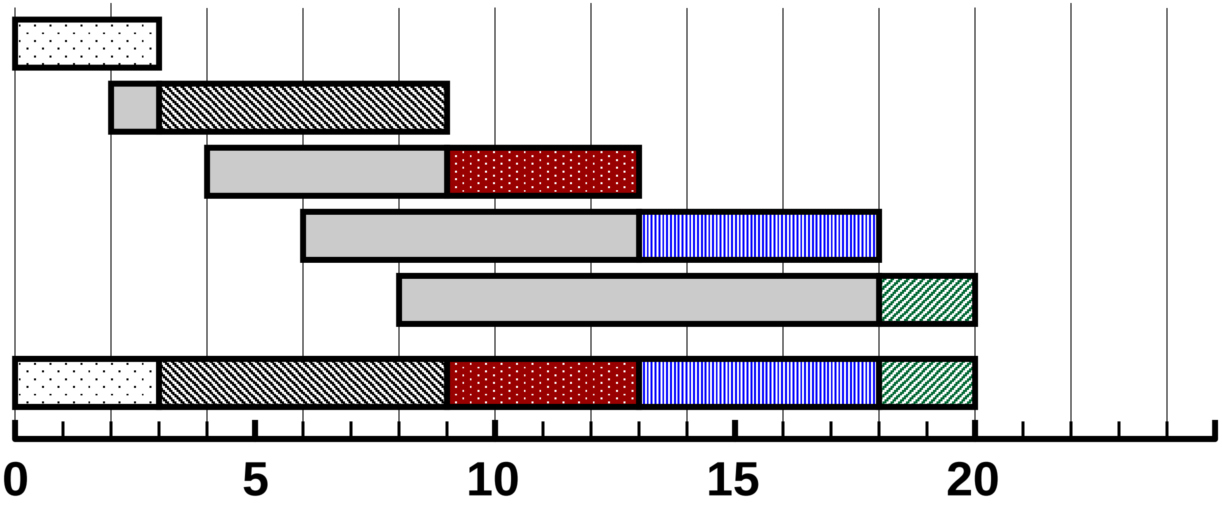


First-Come, First-Served (FIFO)

- Start jobs in the order they arrive (FIFO queue)
- Run each job until completion

Process		Arrival Time	Processing Time	Delay	Turnaround Time
⬚	1	0	3	0	3
▧	2	2	6	1	7
🟥	3	4	4	5	9
🟦	4	6	5	7	12
🟩	5	8	2	10	12

Total time taken, from submission to completion



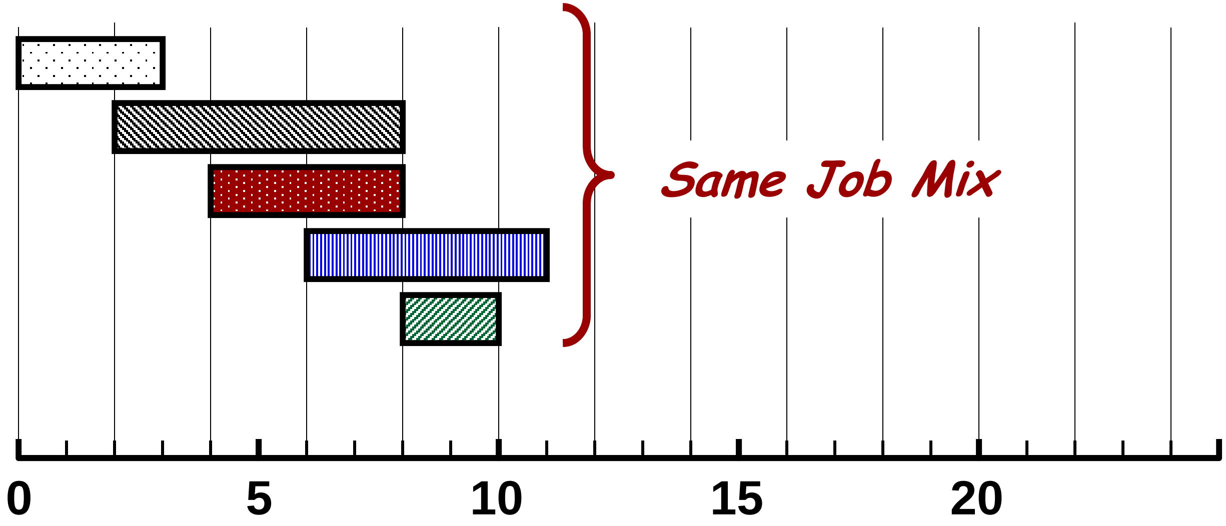
Shortest Job First

- Select the job with the shortest (expected) running time
- Non-Preemptive

Shortest Job First

- Select the job with the shortest (expected) running time
- Non-Preemptive

Process		Arrival Time	Processing Time	Delay	Turnaround Time
⬚	1	0	3		
▨	2	2	6		
🟥	3	4	4		
🟦	4	6	5		
🟩	5	8	2		

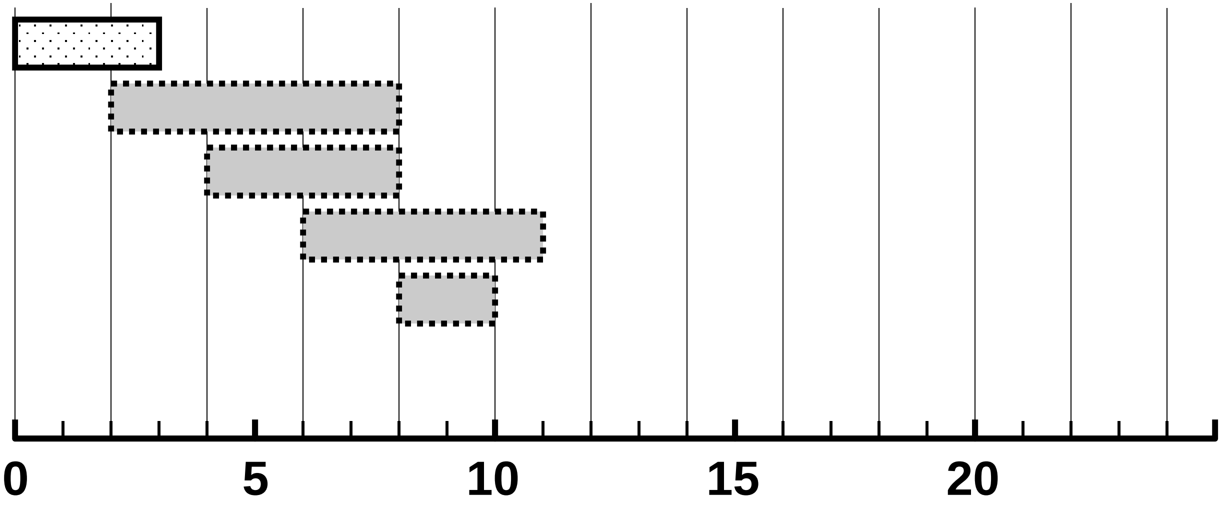


Same Job Mix

Shortest Job First

- Select the job with the shortest (expected) running time
- Non-Preemptive

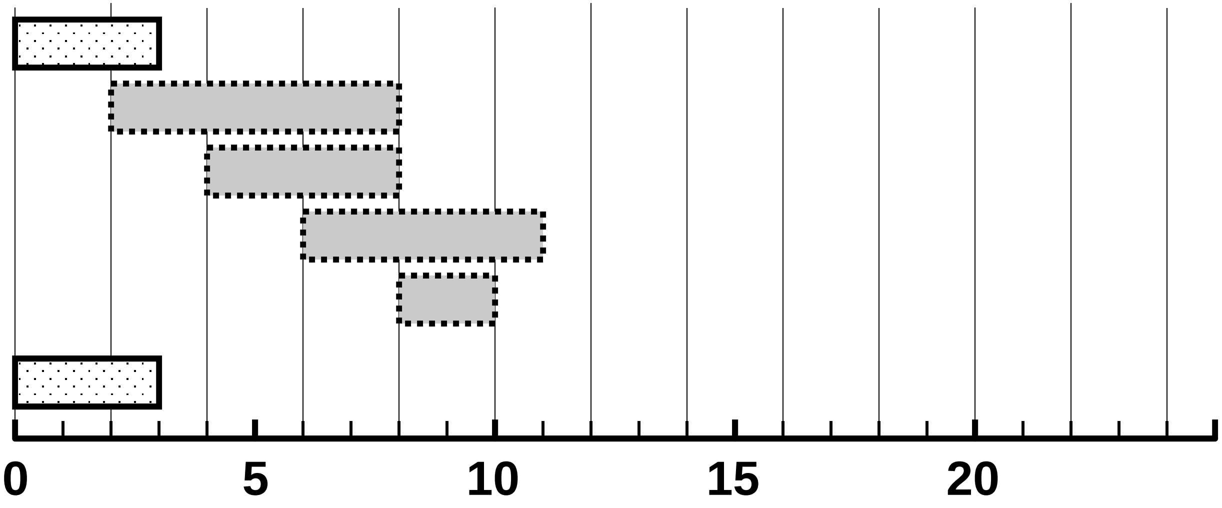
	Arrival	Processing		Turnaround
Process	Time	Time	Delay	Time
░ 1	0	3		
▨ 2	2	6		
🟥 3	4	4		
▥ 4	6	5		
🟩 5	8	2		



Shortest Job First

- Select the job with the shortest (expected) running time
- Non-Preemptive

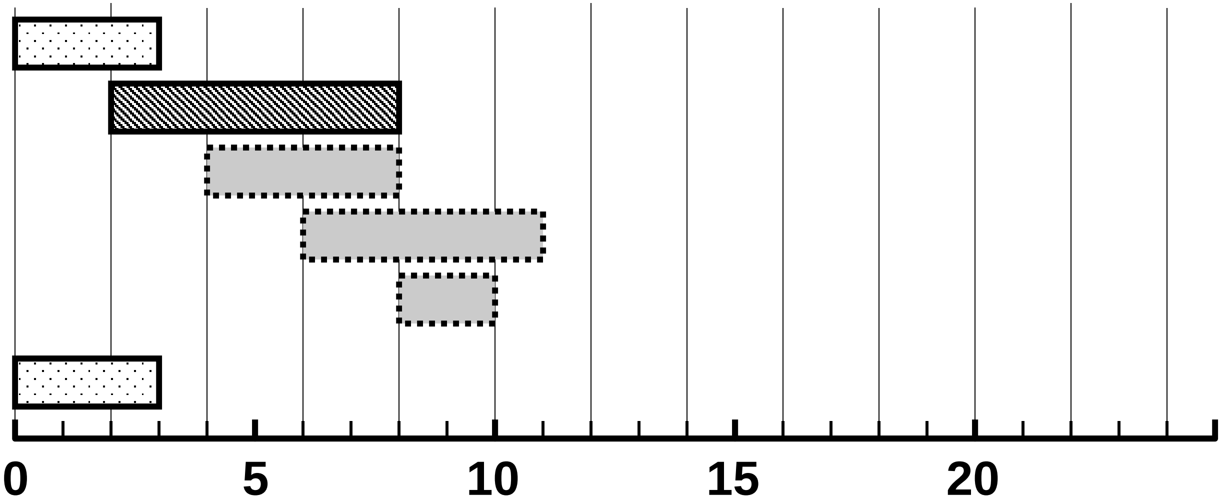
	Arrival	Processing		Turnaround
Process	Time	Time	Delay	Time
⬚ 1	0	3		
▨ 2	2	6		
🟥 3	4	4		
🟦 4	6	5		
🟩 5	8	2		



Shortest Job First

- Select the job with the shortest (expected) running time
- Non-Preemptive

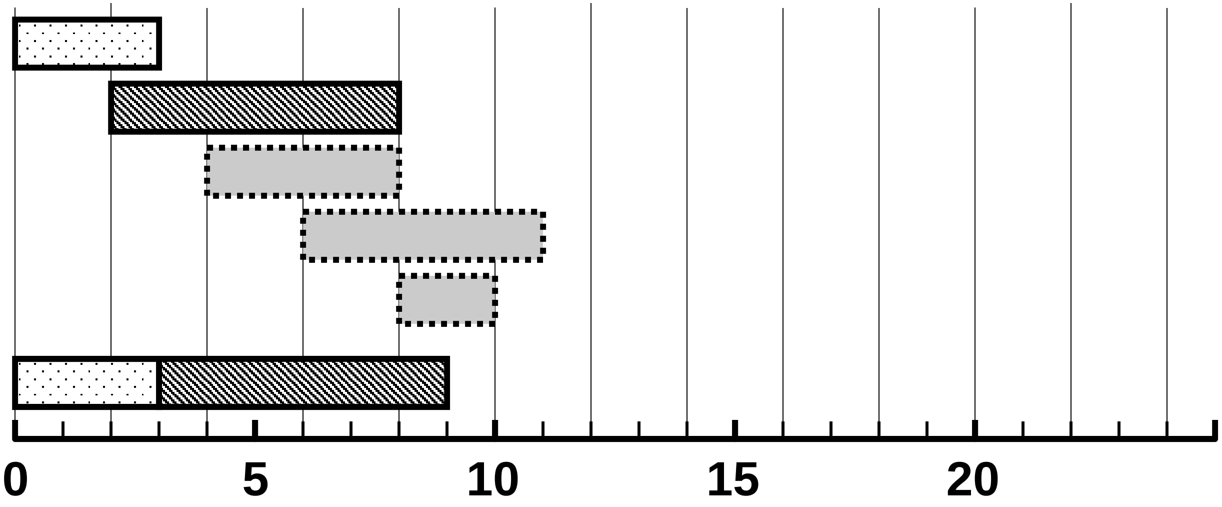
	Arrival	Processing		Turnaround
Process	Time	Time	Delay	Time
⬚ 1	0	3		
▧ 2	2	6		
🟥 3	4	4		
🟦 4	6	5		
🟩 5	8	2		



Shortest Job First

- Select the job with the shortest (expected) running time
- Non-Preemptive

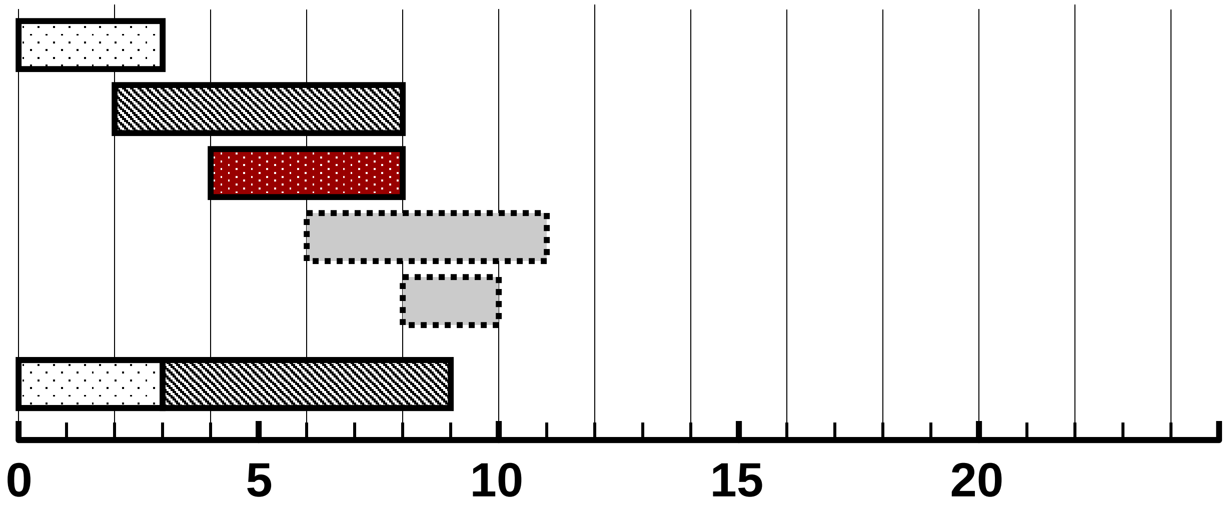
Process	Arrival Time	Processing Time	Delay	Turnaround Time
░ 1	0	3		
▧ 2	2	6		
🟥 3	4	4		
🟦 4	6	5		
🟩 5	8	2		



Shortest Job First

- Select the job with the shortest (expected) running time
- Non-Preemptive

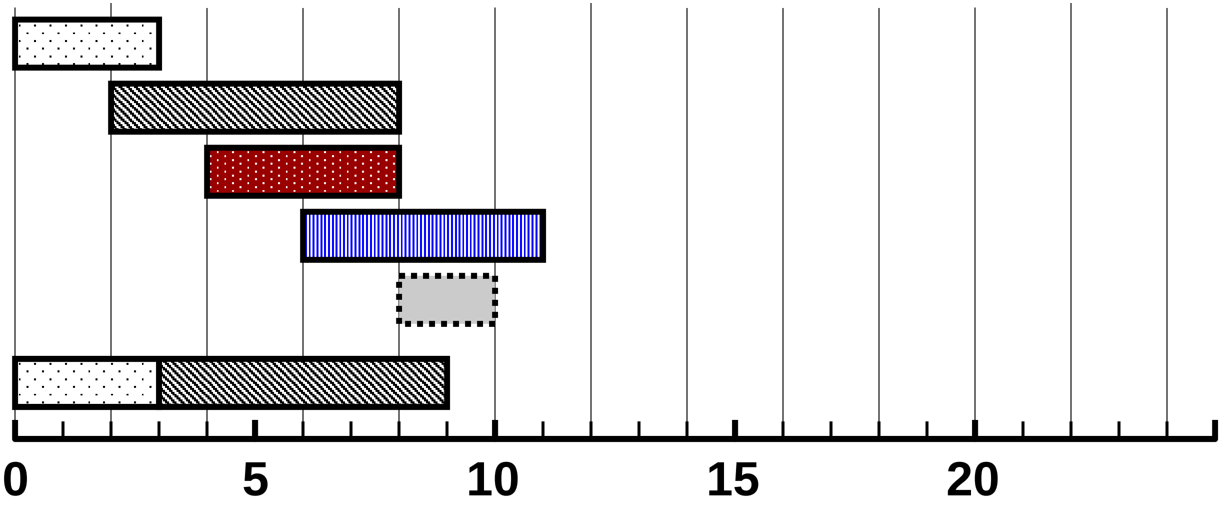
Process		Arrival Time	Processing Time	Delay	Turnaround Time
⬚	1	0	3		
▨	2	2	6		
🟥	3	4	4		
▥	4	6	5		
▧	5	8	2		



Shortest Job First

- Select the job with the shortest (expected) running time
- Non-Preemptive

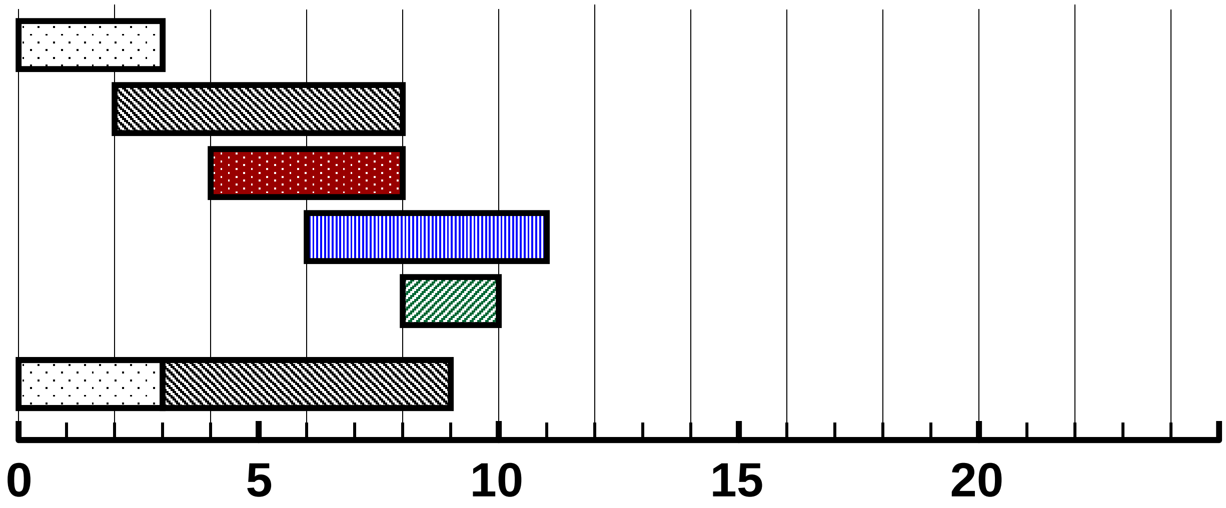
	Arrival	Processing		Turnaround
Process	Time	Time	Delay	Time
□ 1	0	3		
▨ 2	2	6		
🟥 3	4	4		
🟦 4	6	5		
🟩 5	8	2		



Shortest Job First

- Select the job with the shortest (expected) running time
- Non-Preemptive

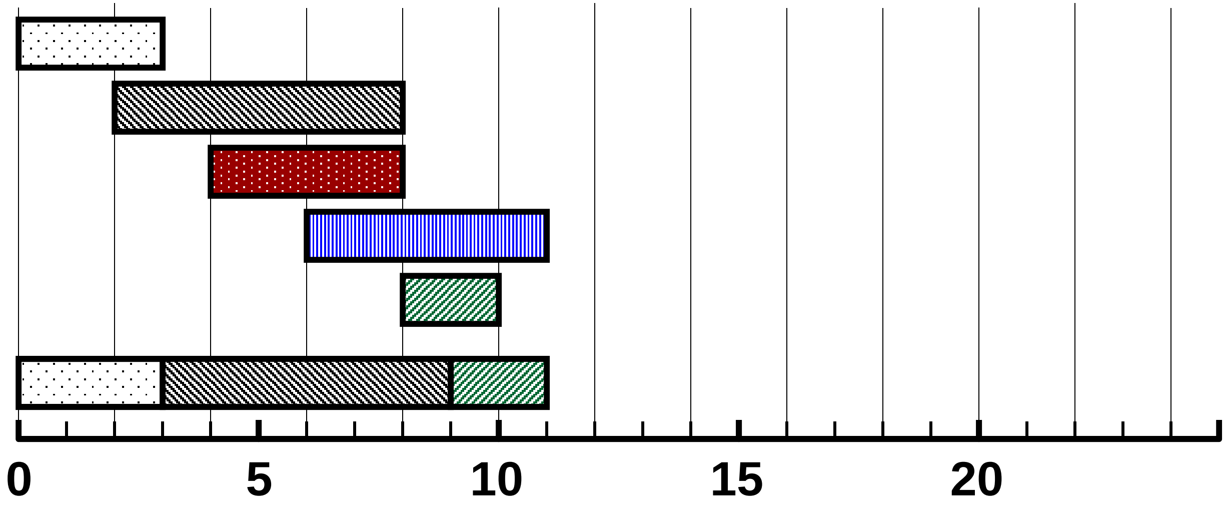
Process	Arrival Time	Processing Time	Delay	Turnaround Time
⬚ 1	0	3		
▨ 2	2	6		
🟥 3	4	4		
🟦 4	6	5		
🟩 5	8	2		



Shortest Job First

- Select the job with the shortest (expected) running time
- Non-Preemptive

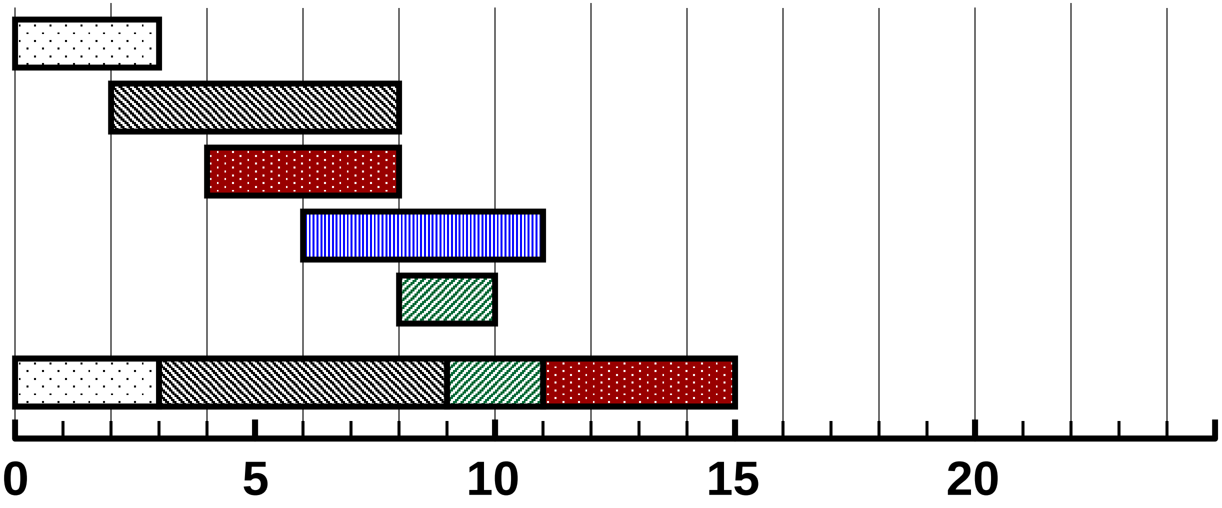
	Arrival	Processing		Turnaround
Process	Time	Time	Delay	Time
⬚ 1	0	3		
▨ 2	2	6		
🟥 3	4	4		
🟦 4	6	5		
🟩 5	8	2		



Shortest Job First

- Select the job with the shortest (expected) running time
- Non-Preemptive

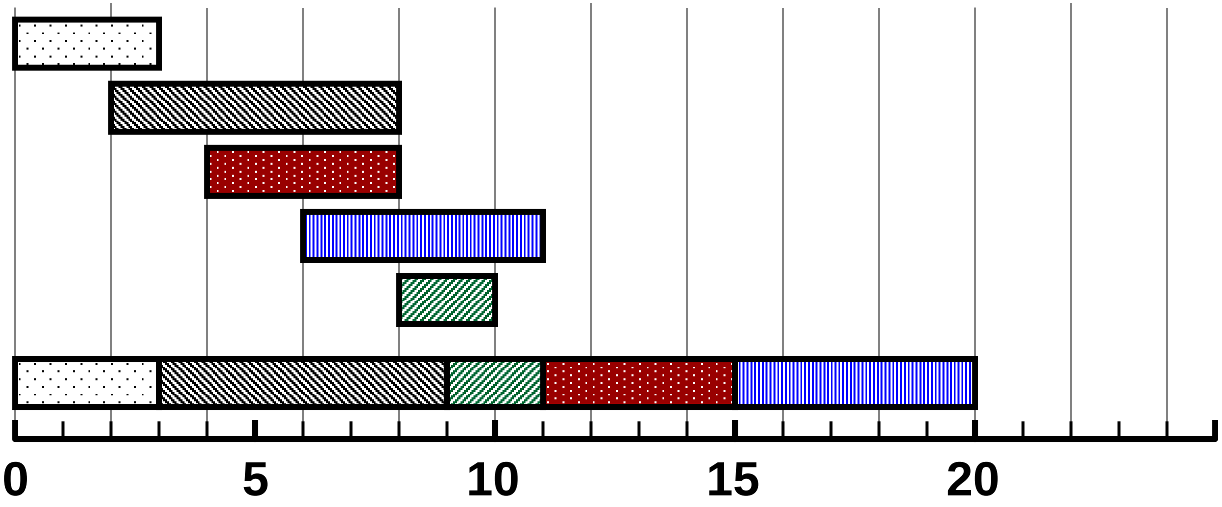
Process	Arrival Time	Processing Time	Delay	Turnaround Time
░ 1	0	3		
▧ 2	2	6		
🟥 3	4	4		
🟦 4	6	5		
🟩 5	8	2		



Shortest Job First

- Select the job with the shortest (expected) running time
- Non-Preemptive

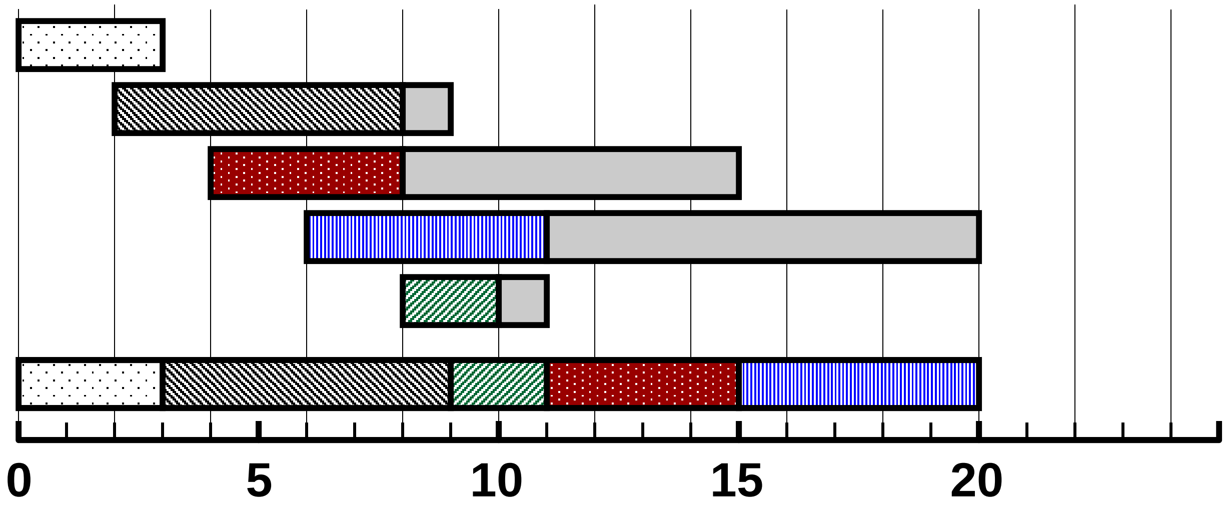
Process		Arrival Time	Processing Time	Delay	Turnaround Time
░	1	0	3		
▨	2	2	6		
🟥	3	4	4		
🟦	4	6	5		
🟩	5	8	2		



Shortest Job First

- Select the job with the shortest (expected) running time
- Non-Preemptive

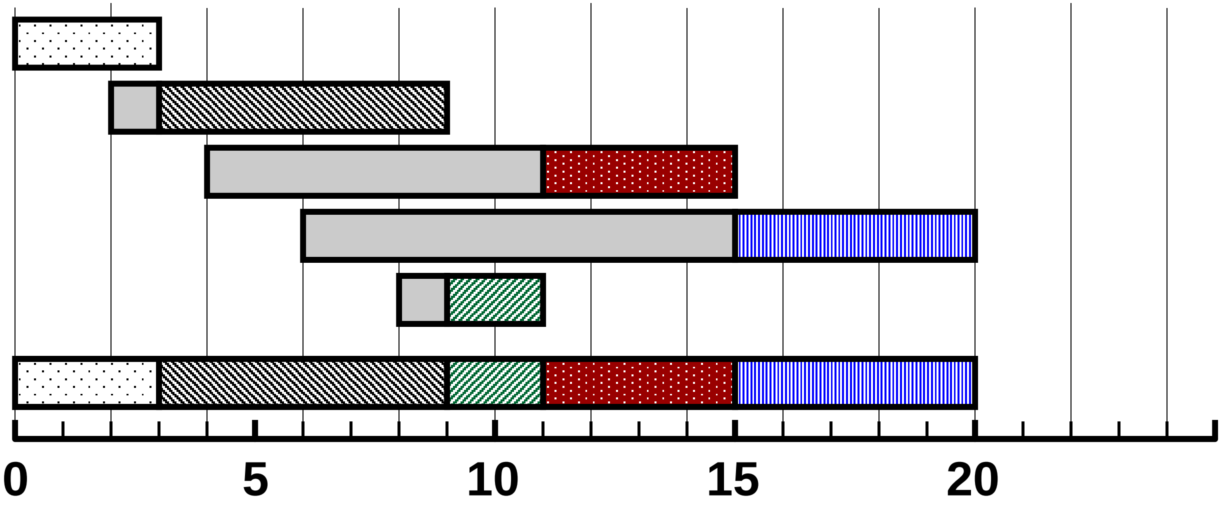
	Arrival	Processing		Turnaround
Process	Time	Time	Delay	Time
░ 1	0	3		
▨ 2	2	6		
🟥 3	4	4		
🟦 4	6	5		
🟩 5	8	2		



Shortest Job First

- Select the job with the shortest (expected) running time
- Non-Preemptive

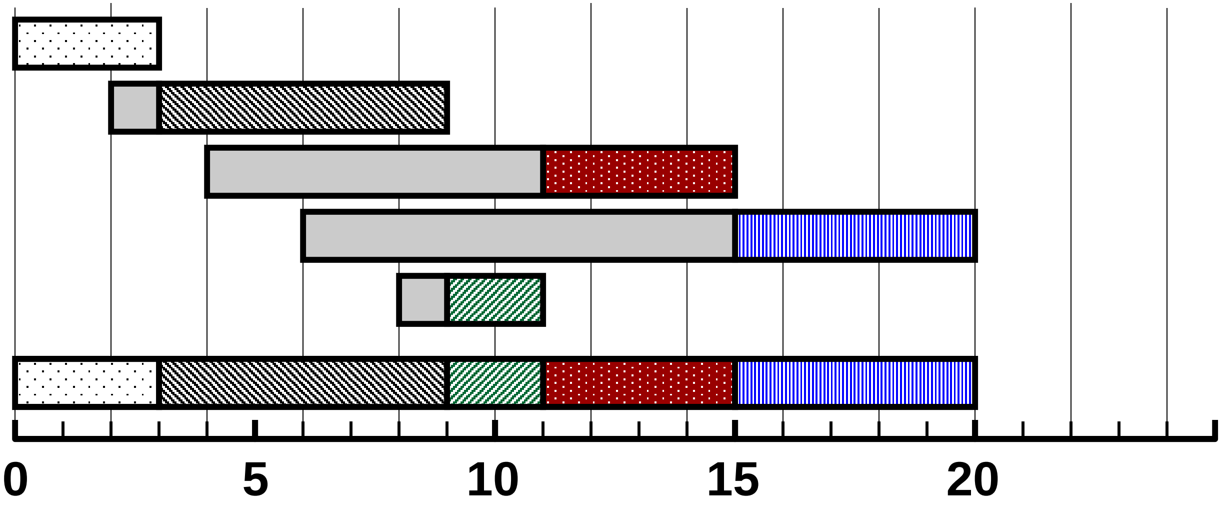
	Process	Arrival Time	Processing Time	Delay	Turnaround Time
⬚	1	0	3		
▨	2	2	6		
🟥	3	4	4		
🟦	4	6	5		
🟩	5	8	2		



Shortest Job First

- Select the job with the shortest (expected) running time
- Non-Preemptive

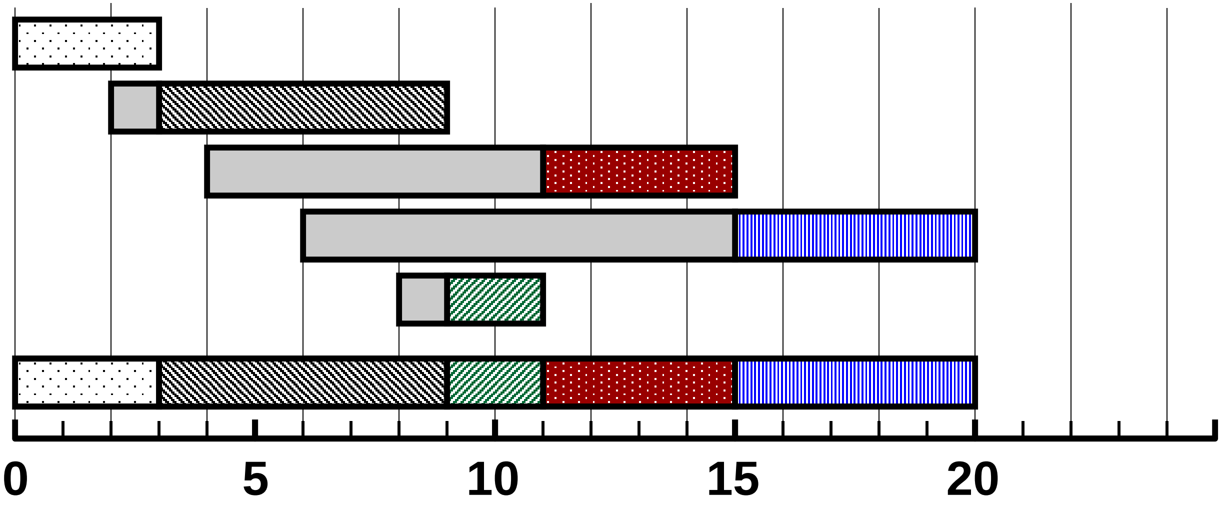
		Arrival	Processing		Turnaround
Process		Time	Time	Delay	Time
⬚	1	0	3	0	
▨	2	2	6	1	
🟥	3	4	4	7	
🟦	4	6	5	9	
🟩	5	8	2	1	



Shortest Job First

- Select the job with the shortest (expected) running time
- Non-Preemptive

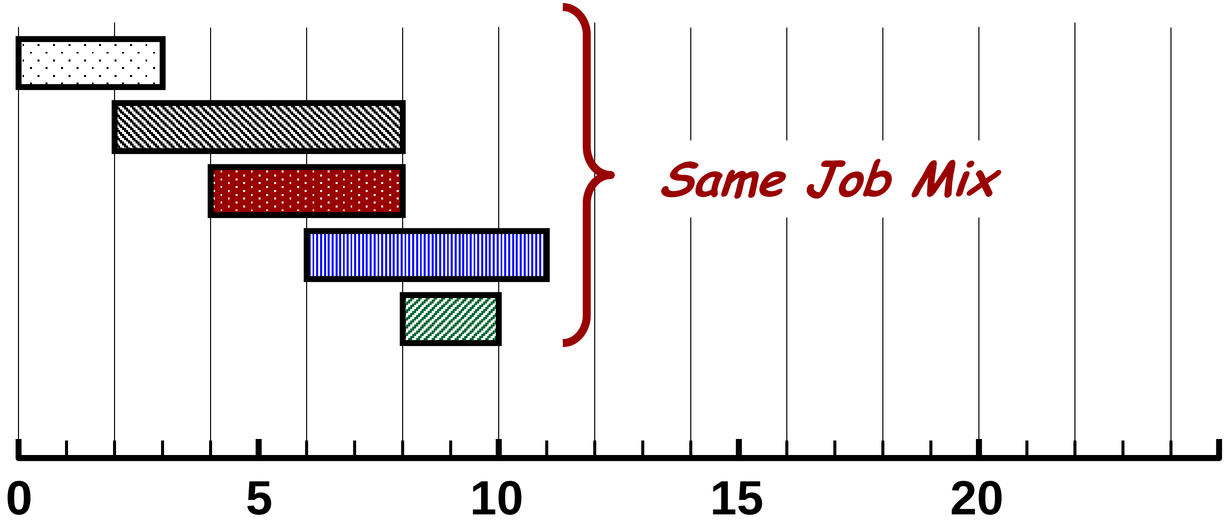
	Arrival	Processing		Turnaround
Process	Time	Time	Delay	Time
⬚ 1	0	3	0	3
▨ 2	2	6	1	7
🟥 3	4	4	7	11
🟦 4	6	5	9	14
🟩 5	8	2	1	3



Shortest Remaining Time

- Preemptive version of SJF

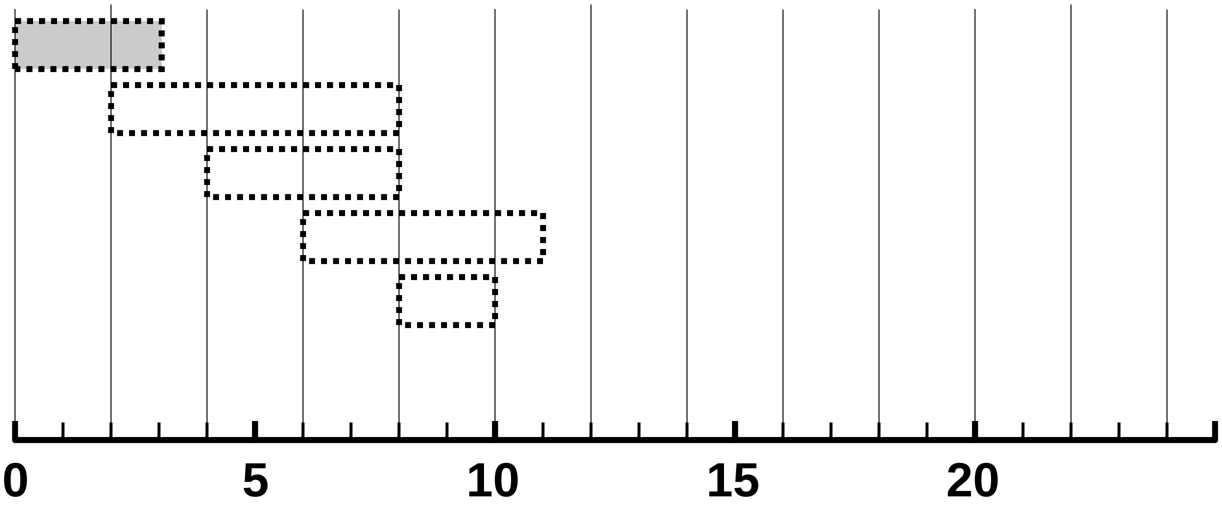
		Arrival	Processing		Turnaround
Process		Time	Time	Delay	Time
░	1	0	3		
▨	2	2	6		
🟥	3	4	4		
🟦	4	6	5		
🟩	5	8	2		



Shortest Remaining Time

- Preemptive version of SJF

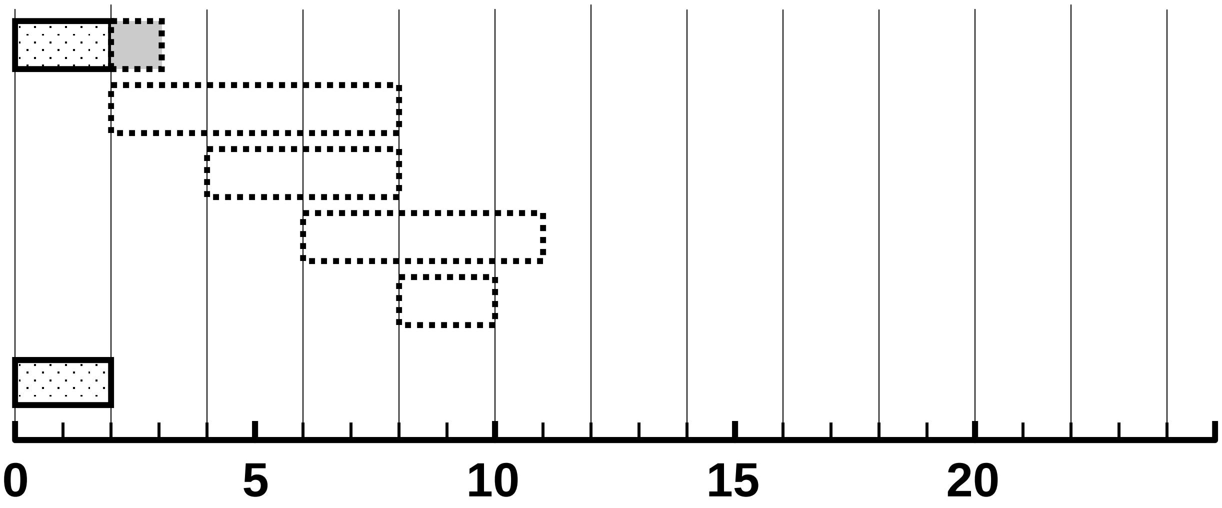
Process		Arrival Time	Processing Time	Delay	Turnaround Time
⬚	1	0	3		
▨	2	2	6		
🟥	3	4	4		
🟦	4	6	5		
🟩	5	8	2		



Shortest Remaining Time

- Preemptive version of SJF

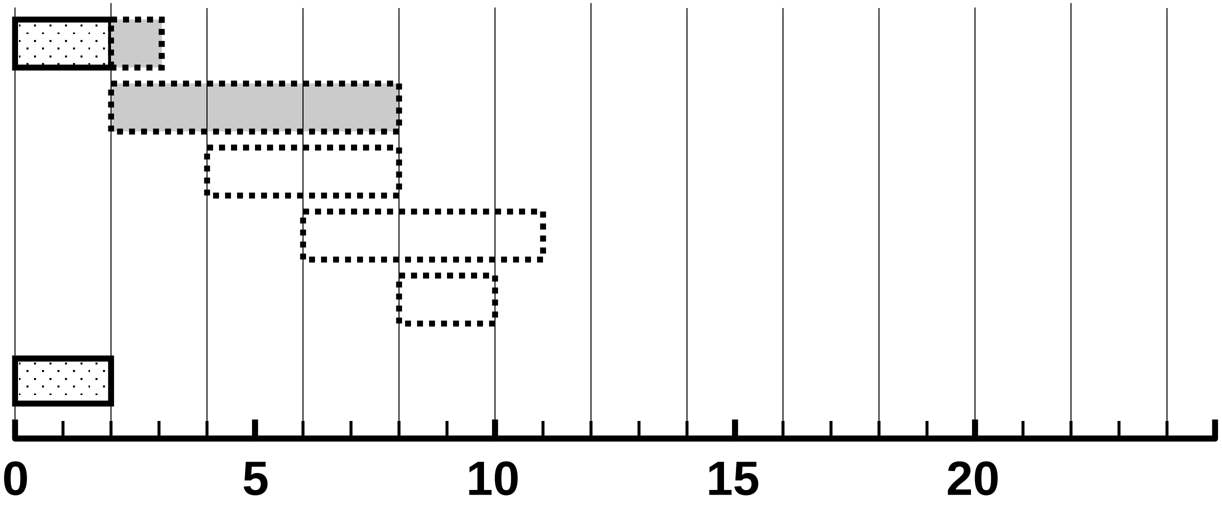
	Process	Arrival Time	Processing Time	Delay	Turnaround Time
⬚	1	0	3		
▨	2	2	6		
🟥	3	4	4		
🟦	4	6	5		
🟩	5	8	2		



Shortest Remaining Time

- Preemptive version of SJF

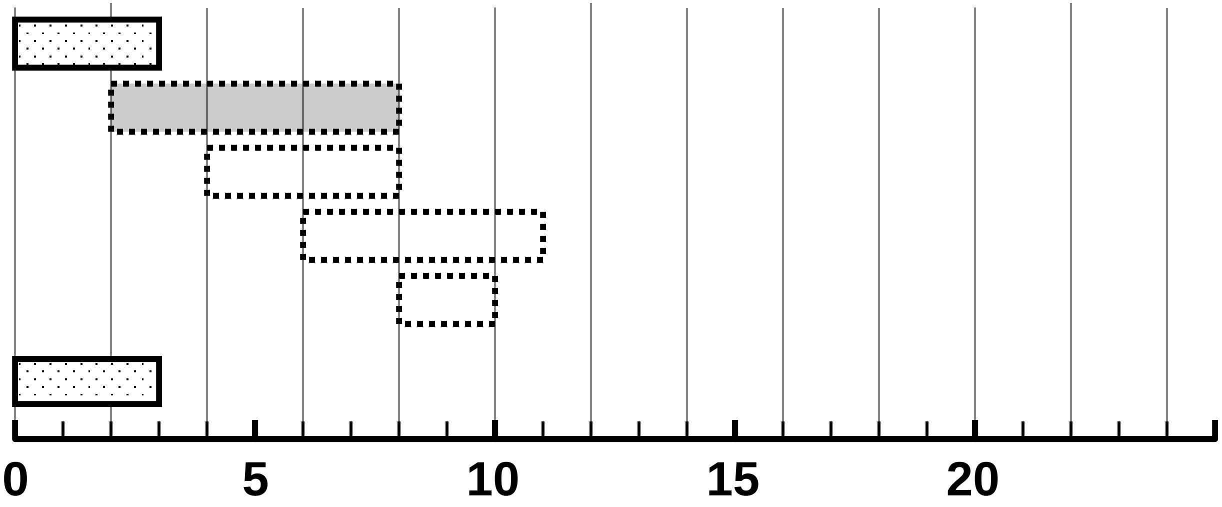
Process		Arrival Time	Processing Time	Delay	Turnaround Time
░	1	0	3		
▧	2	2	6		
🟥	3	4	4		
▥	4	6	5		
🟩	5	8	2		



Shortest Remaining Time

- Preemptive version of SJF

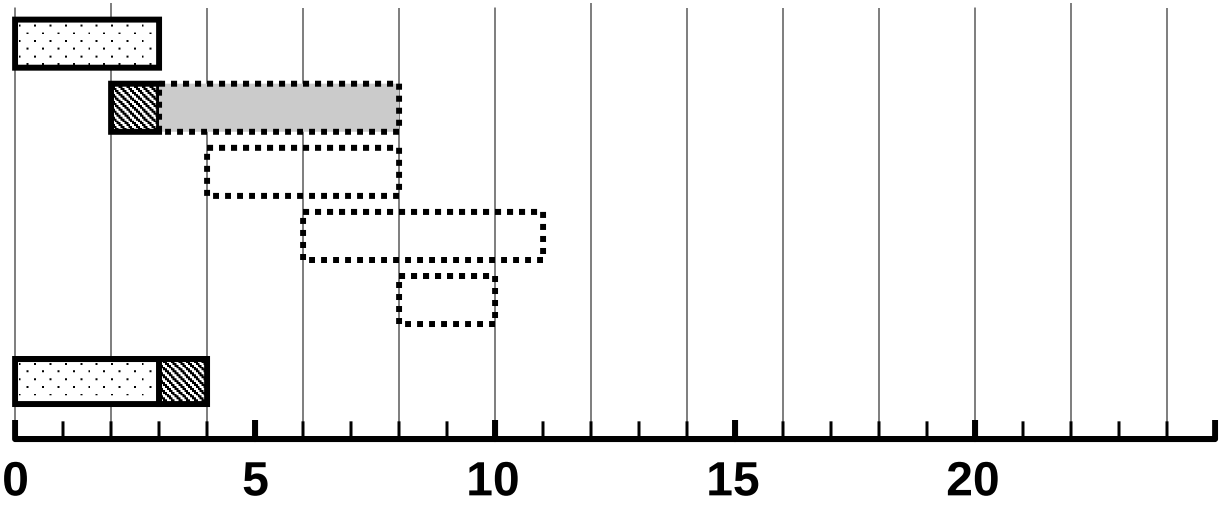
Process		Arrival Time	Processing Time	Delay	Turnaround Time
⬚	1	0	3		
▨	2	2	6		
🟥	3	4	4		
🟦	4	6	5		
🟩	5	8	2		



Shortest Remaining Time

- Preemptive version of SJF

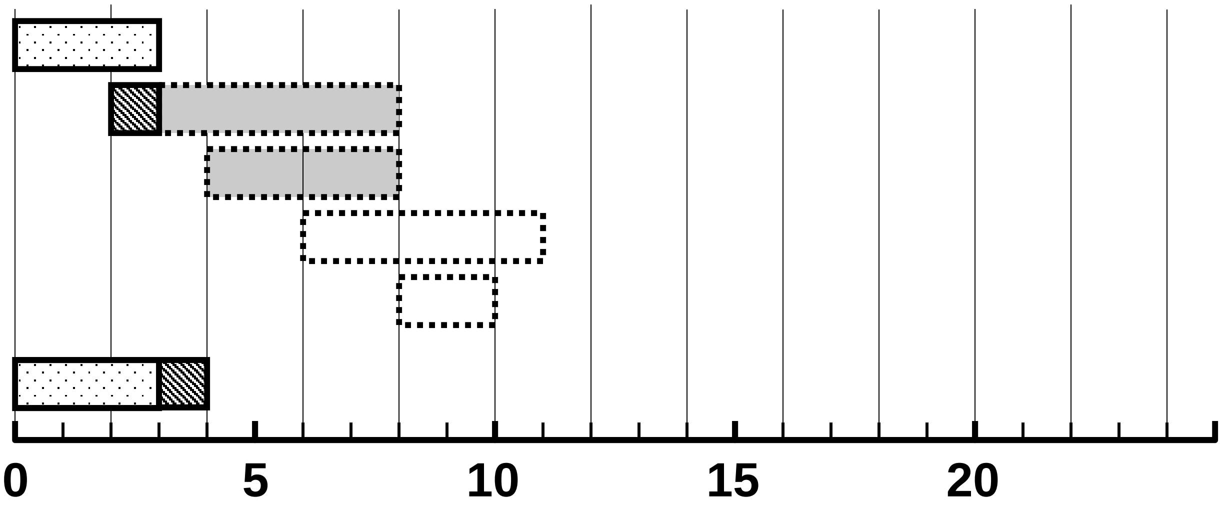
Process		Arrival Time	Processing Time	Delay	Turnaround Time
░	1	0	3		
▧	2	2	6		
🟥	3	4	4		
🟦	4	6	5		
🟩	5	8	2		



Shortest Remaining Time

- Preemptive version of SJF

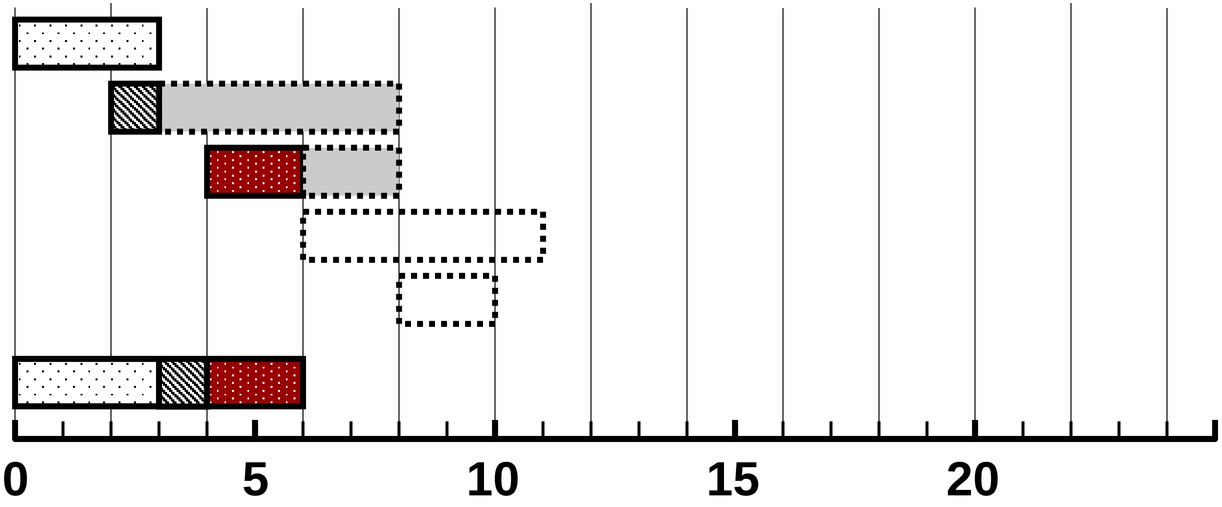
Process	Arrival Time	Processing Time	Delay	Turnaround Time
░ 1	0	3		
▧ 2	2	6		
🟥 3	4	4		
▥ 4	6	5		
🟩 5	8	2		



Shortest Remaining Time

- Preemptive version of SJF

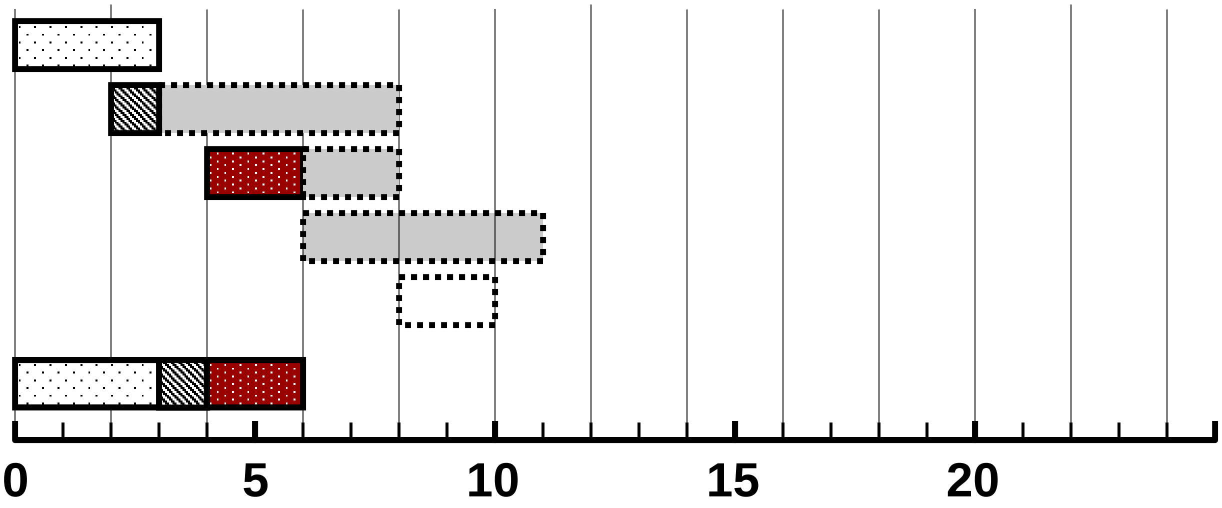
Process		Arrival	Processing		Turnaround
		Time	Time	Delay	Time
░	1	0	3		
▨	2	2	6		
🟥	3	4	4		
🟦	4	6	5		
🟩	5	8	2		



Shortest Remaining Time

- Preemptive version of SJF

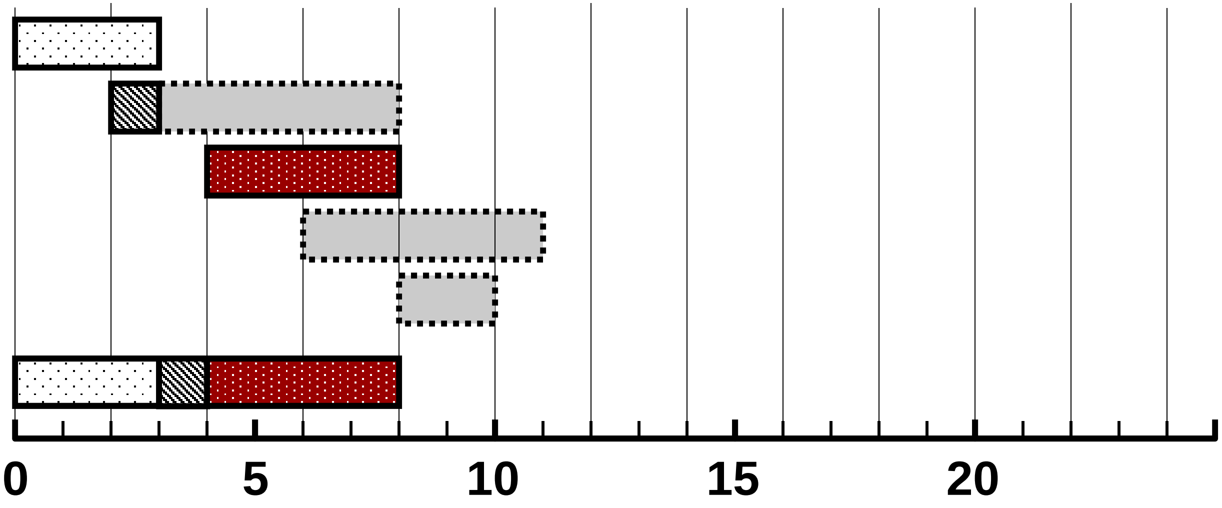
Process	Arrival Time	Processing Time	Delay	Turnaround Time
⬚ 1	0	3		
▨ 2	2	6		
🟥 3	4	4		
🟦 4	6	5		
🟩 5	8	2		



Shortest Remaining Time

- Preemptive version of SJF

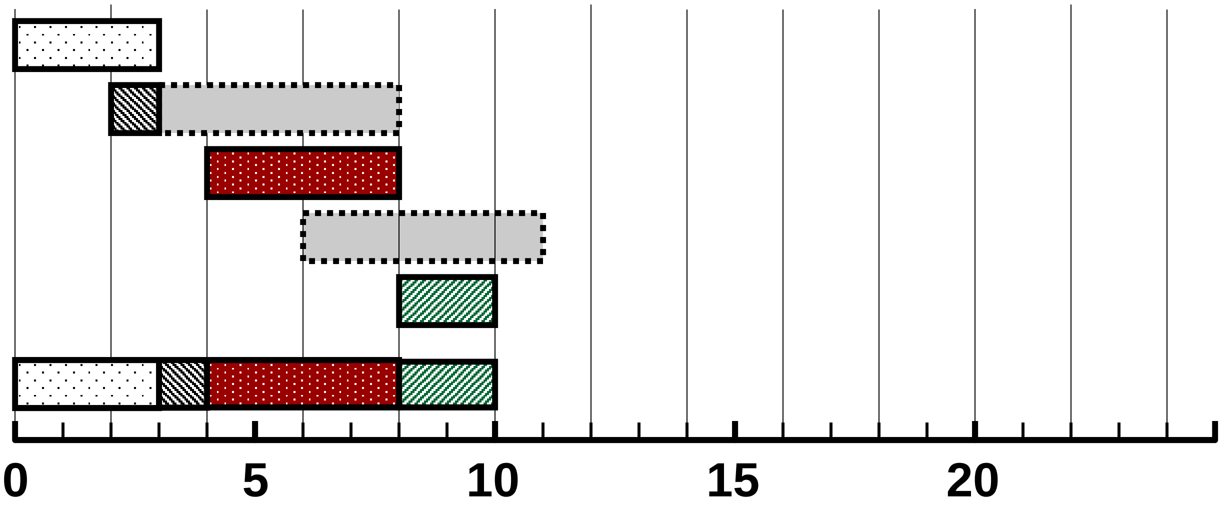
Process	Arrival Time	Processing Time	Delay	Turnaround Time
⬚ 1	0	3		
▨ 2	2	6		
🟥 3	4	4		
▥ 4	6	5		
▧ 5	8	2		



Shortest Remaining Time

- Preemptive version of SJF

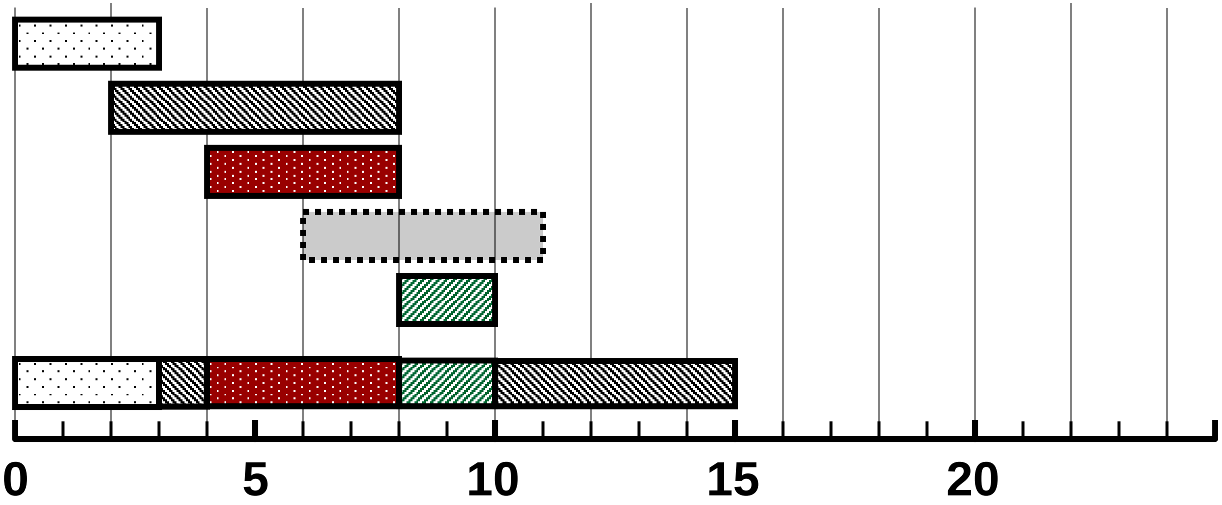
Process	Arrival Time	Processing Time	Delay	Turnaround Time
░ 1	0	3		
▨ 2	2	6		
🟥 3	4	4		
🟦 4	6	5		
🟩 5	8	2		



Shortest Remaining Time

- Preemptive version of SJF

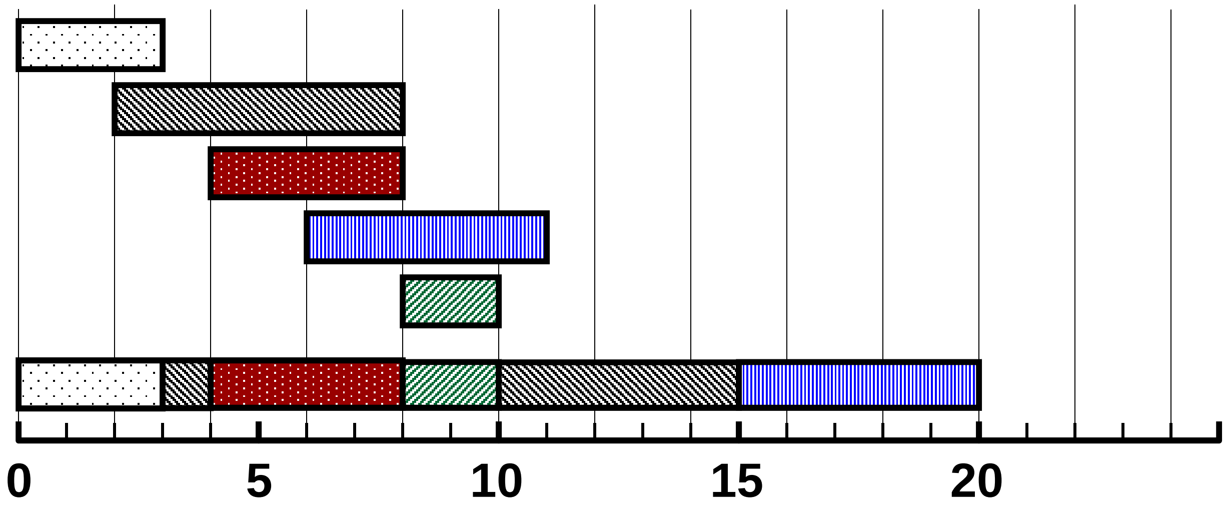
Process		Arrival Time	Processing Time	Delay	Turnaround Time
░	1	0	3		
▨	2	2	6		
🟥	3	4	4		
🟦	4	6	5		
🟩	5	8	2		



Shortest Remaining Time

- Preemptive version of SJF

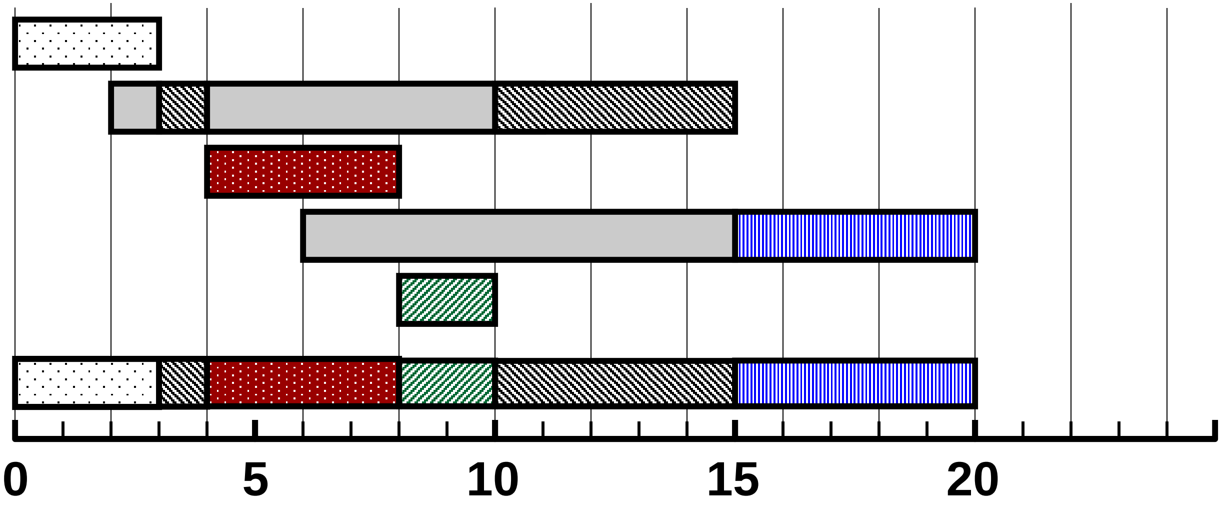
Process		Arrival Time	Processing Time	Delay	Turnaround Time
⬚	1	0	3		
▧	2	2	6		
🟥	3	4	4		
🟦	4	6	5		
🟩	5	8	2		



Shortest Remaining Time

- Preemptive version of SJF

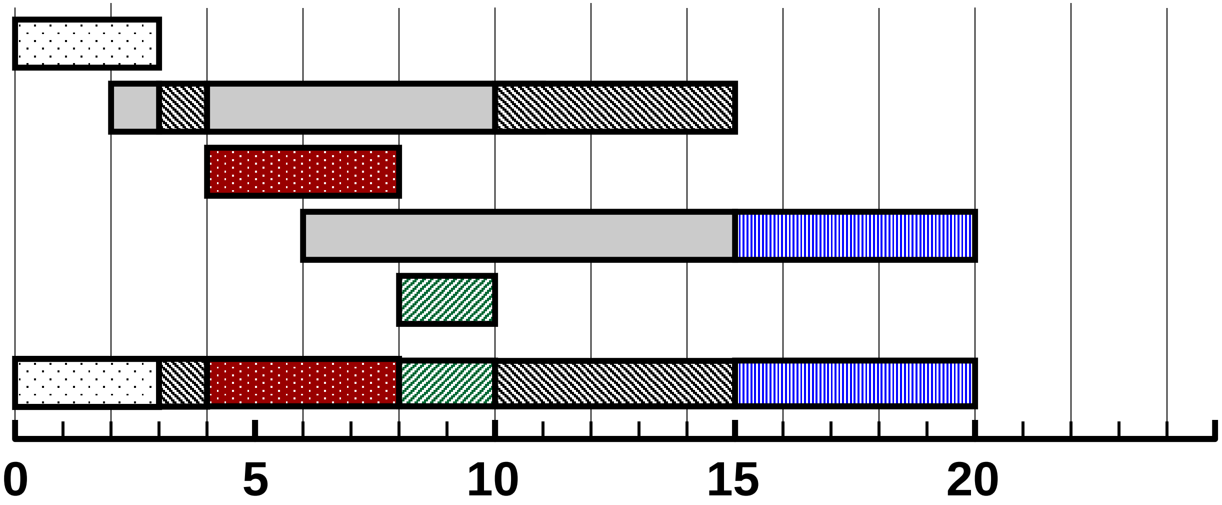
		Arrival	Processing		Turnaround
Process		Time	Time	Delay	Time
░	1	0	3		
▧	2	2	6		
🟥	3	4	4		
🟦	4	6	5		
🟩	5	8	2		



Shortest Remaining Time

- Preemptive version of SJF

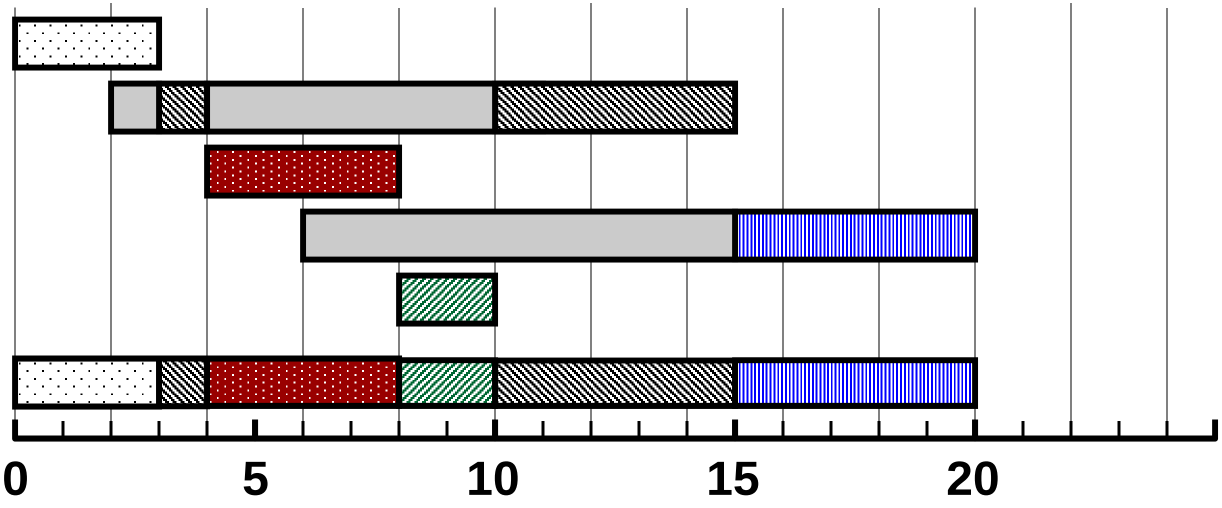
Process		Arrival Time	Processing Time	Delay	Turnaround Time
⬚	1	0	3	0	
▨	2	2	6	7	
🟥	3	4	4	0	
🟦	4	6	5	9	
🟩	5	8	2	0	



Shortest Remaining Time

- Preemptive version of SJF

Process	Arrival Time	Processing Time	Delay	Turnaround Time
⬚ 1	0	3	0	3
▨ 2	2	6	7	13
🟥 3	4	4	0	4
🟦 4	6	5	9	14
🟩 5	8	2	0	2



Round-Robin Scheduling

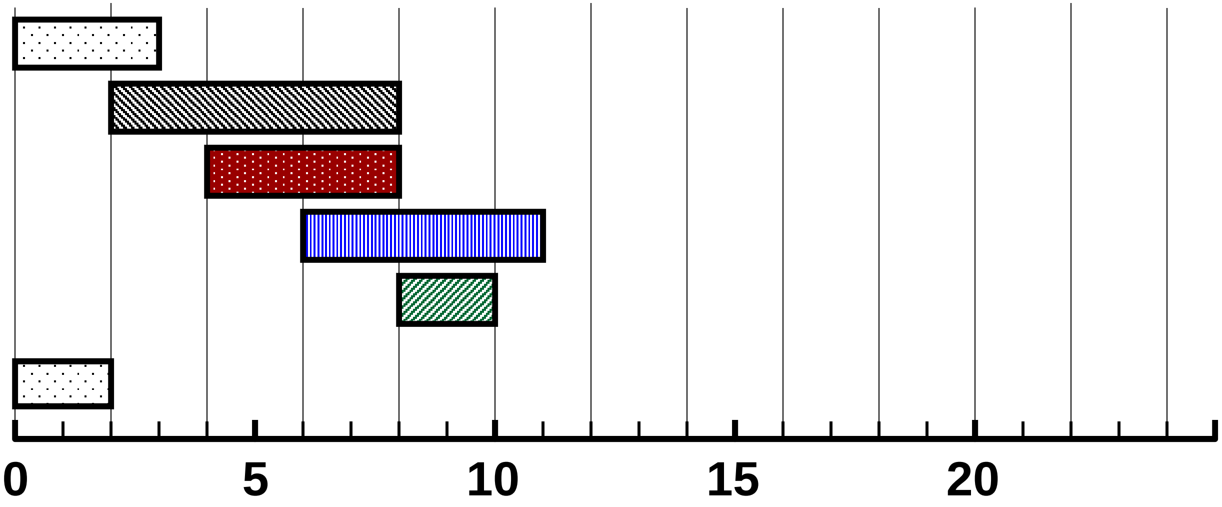
- **Goal:** **Enable interactivity**
  - Limit the amount of CPU that a process can have at one time.

- ***Time quantum***
  - Amount of time the OS gives a process before intervention
  - The "time slice"
  - Typically: 1 to 100ms

Round-Robin Scheduling

•

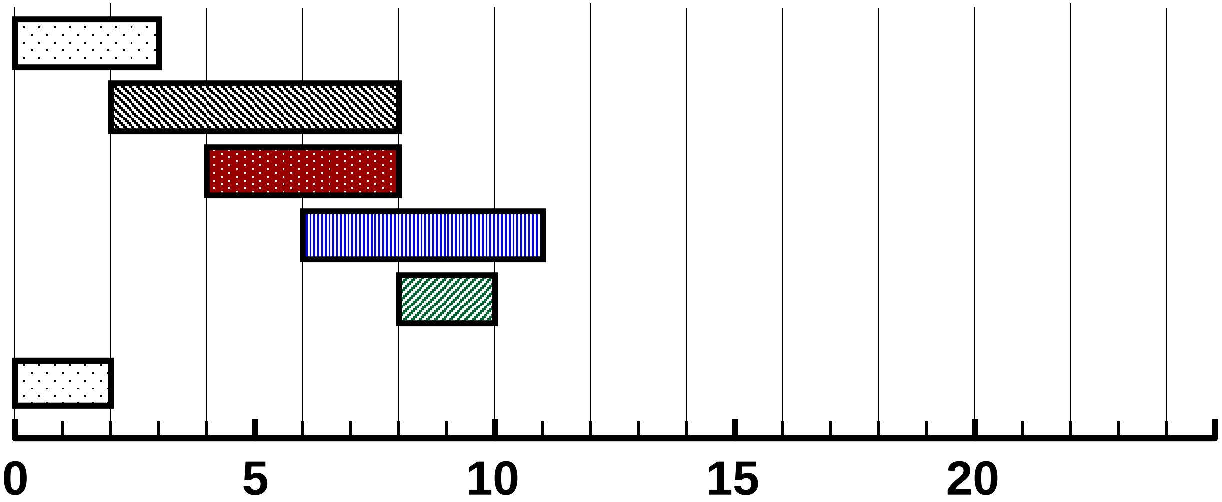
	Process	Arrival Time	Processing Time
░	1	0	3
▨	2	2	6
🟥	3	4	4
🟦	4	6	5
🟩	5	8	2

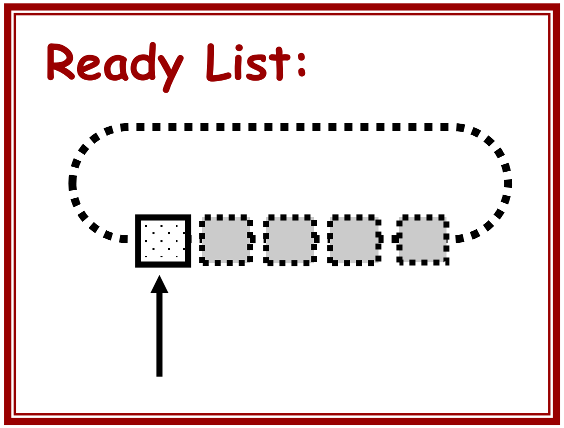


Round-Robin Scheduling

-

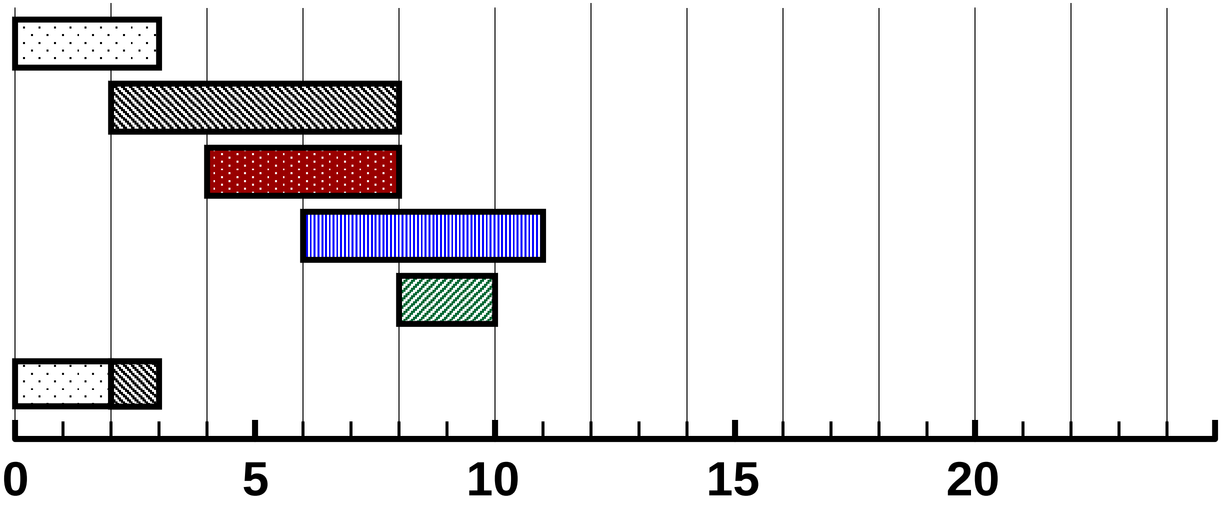
	Process	Arrival Time	Processing Time
░	1	0	3
▩	2	2	6
🟥	3	4	4
🟦	4	6	5
🟩	5	8	2

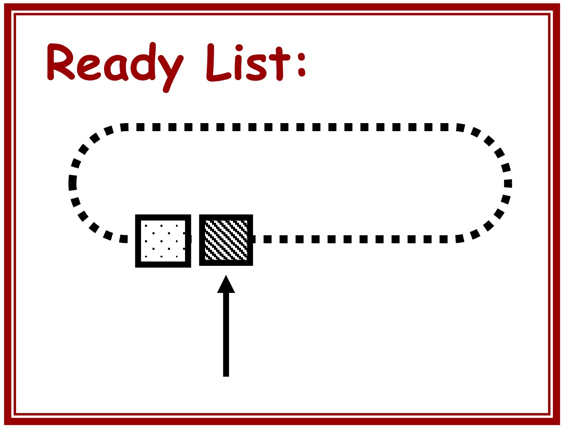



Ready List:

Round-Robin Scheduling

	Process	Arrival Time	Processing Time
⬚	1	0	3
▩	2	2	6
🟥	3	4	4
🟦	4	6	5
🟩	5	8	2

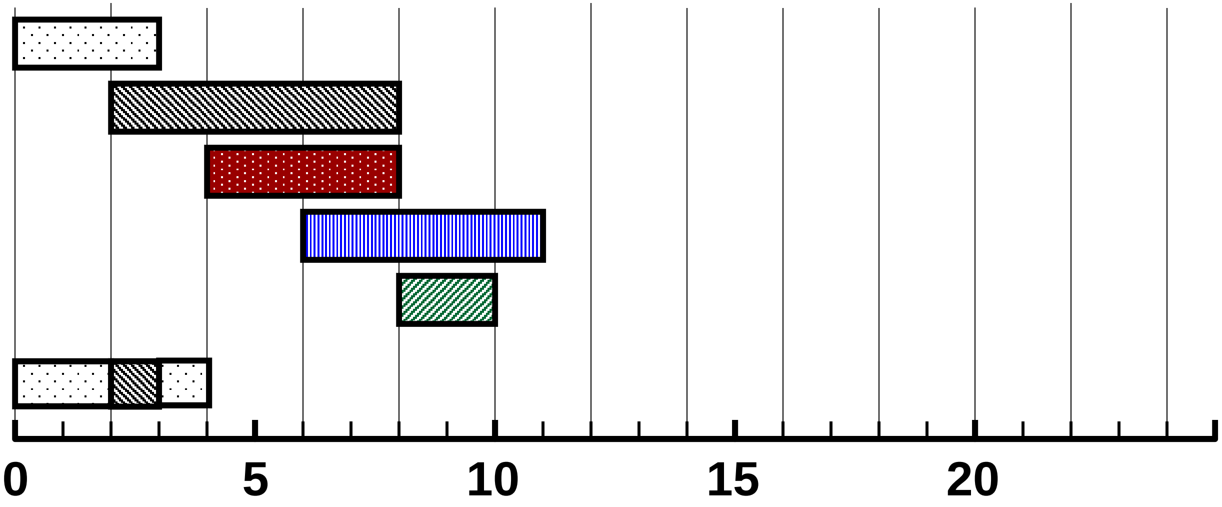


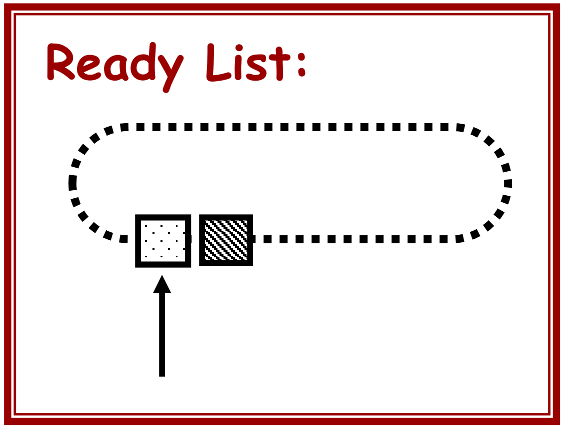

Ready List:

Round-Robin Scheduling

-

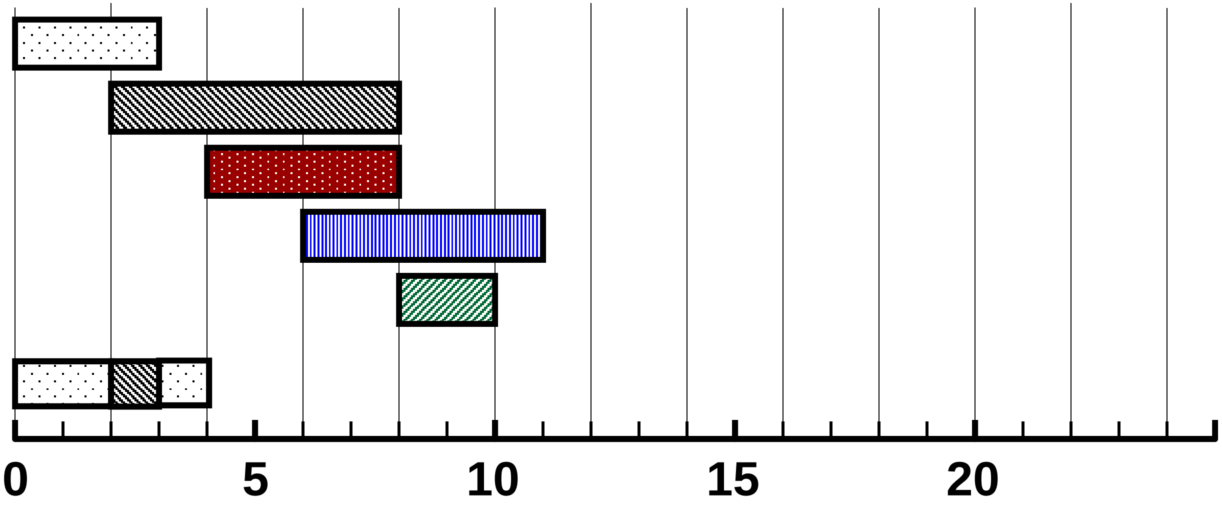
	Process	Arrival Time	Processing Time
░	1	0	3
▨	2	2	6
🟥	3	4	4
🟦	4	6	5
🟩	5	8	2

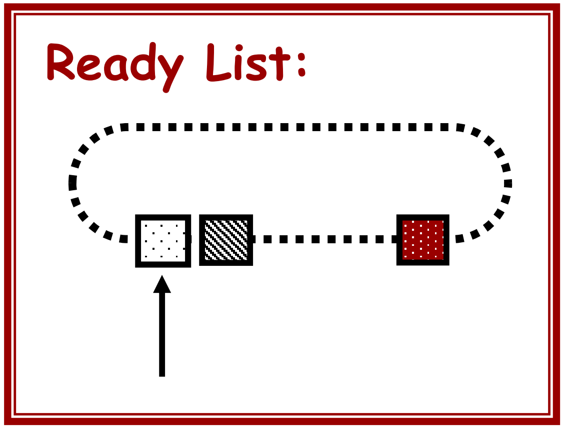



Ready List:

Round-Robin Scheduling

	Process	Arrival Time	Processing Time
⬚	1	0	3
▨	2	2	6
🟥	3	4	4
🟦	4	6	5
🟩	5	8	2

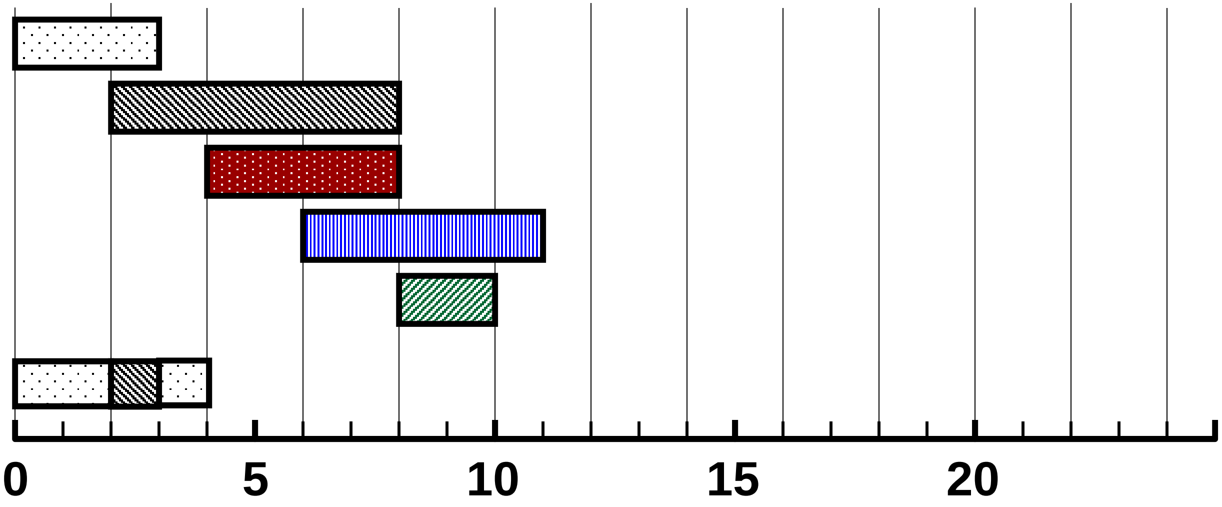


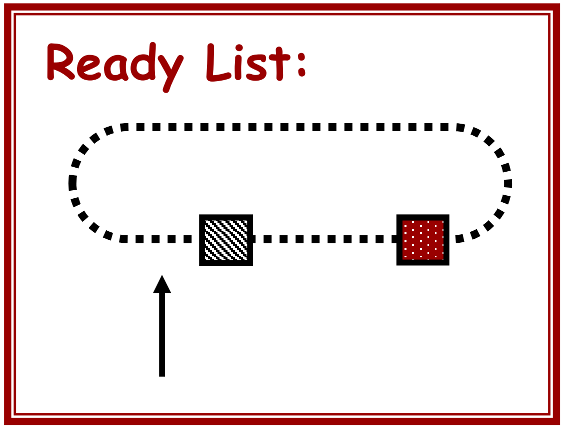


Round-Robin Scheduling

-

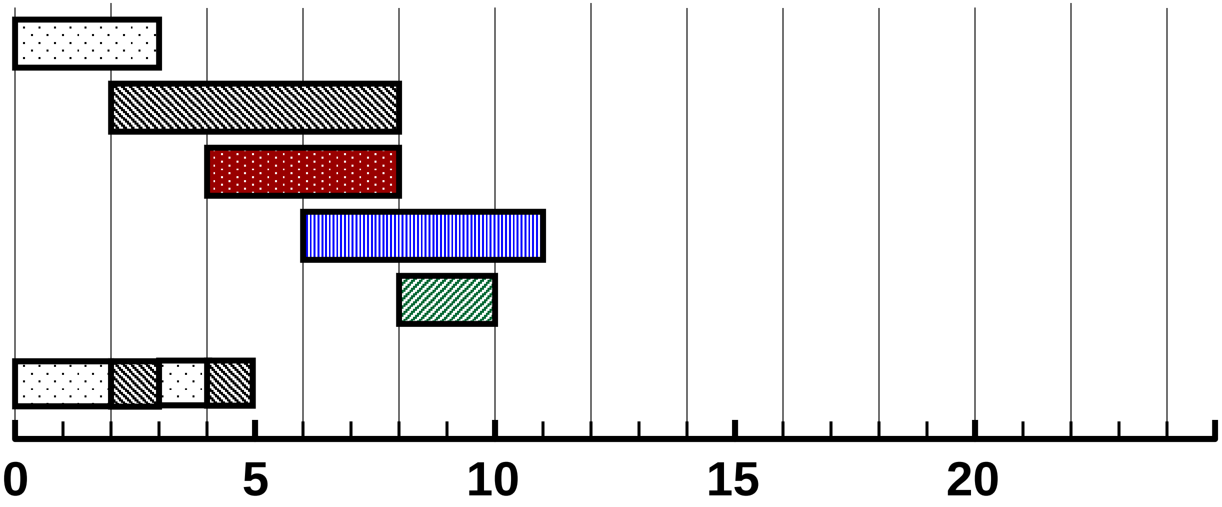
	Process	Arrival Time	Processing Time
░	1	0	3
▩	2	2	6
🟥	3	4	4
🟦	4	6	5
🟩	5	8	2

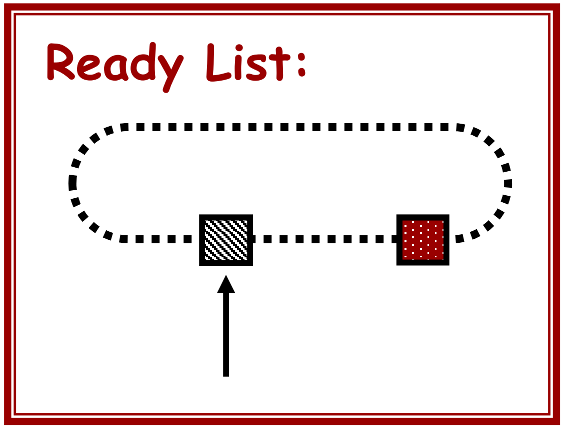



Ready List:

Round-Robin Scheduling

	Process	Arrival Time	Processing Time
░	1	0	3
▩	2	2	6
▦	3	4	4
▥	4	6	5
▨	5	8	2

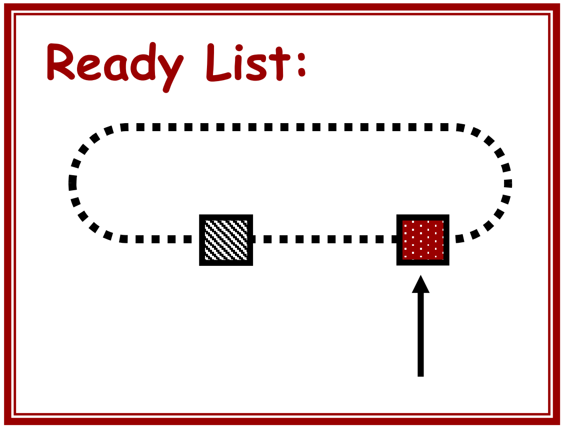




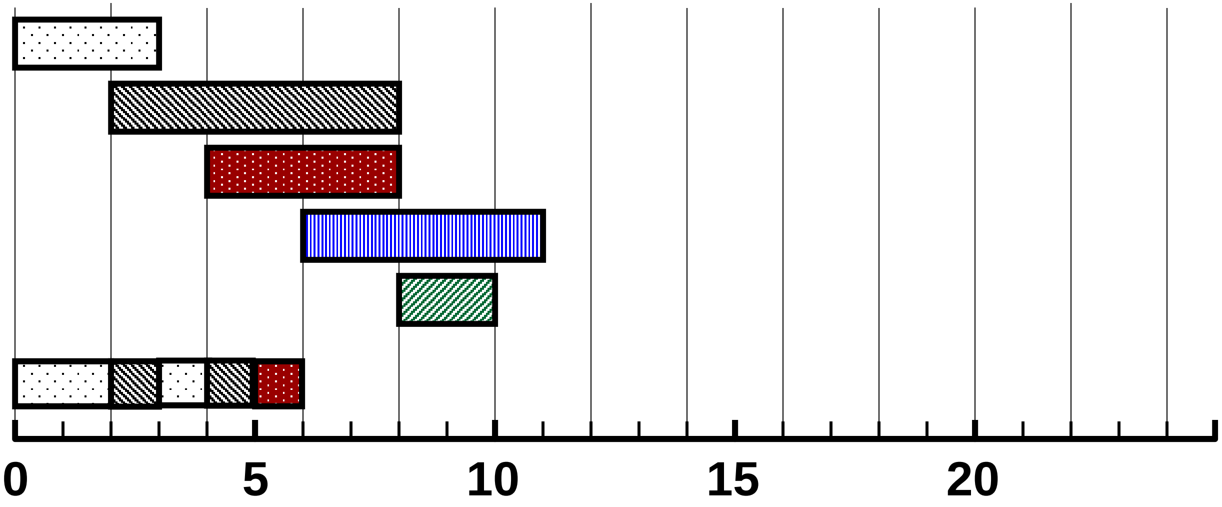
Round-Robin Scheduling

-

	Process	Arrival Time	Processing Time
⬚	1	0	3
▨	2	2	6
🟥	3	4	4
🟦	4	6	5
🟩	5	8	2

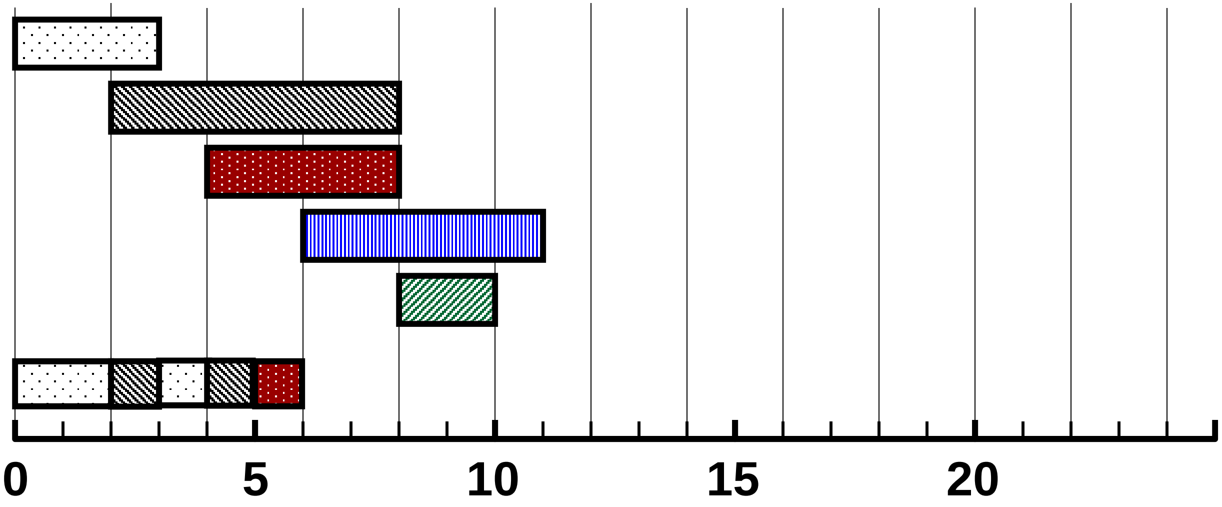


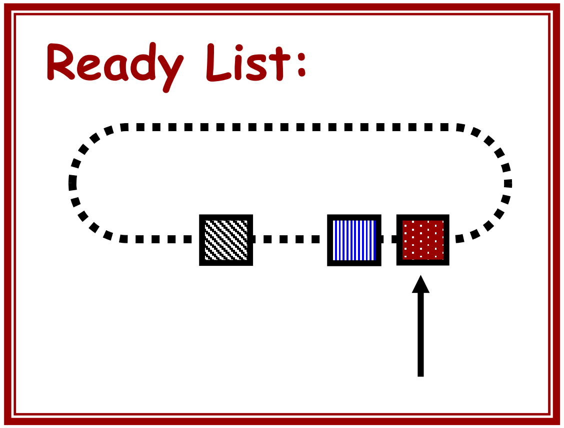
Ready List:



Round-Robin Scheduling

	Process	Arrival Time	Processing Time
⬚	1	0	3
▨	2	2	6
🟥	3	4	4
🟦	4	6	5
🟩	5	8	2

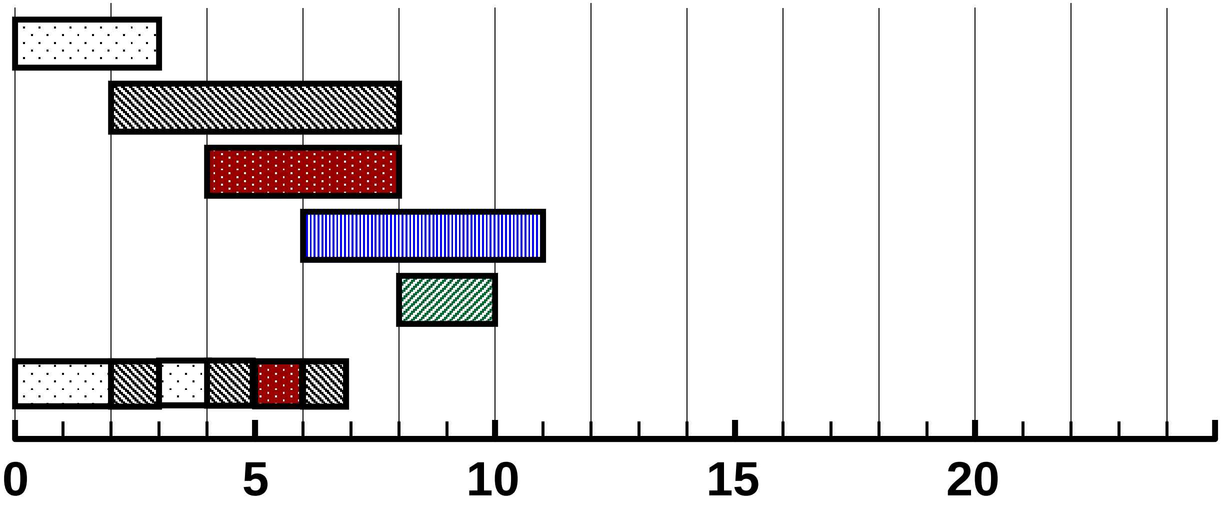


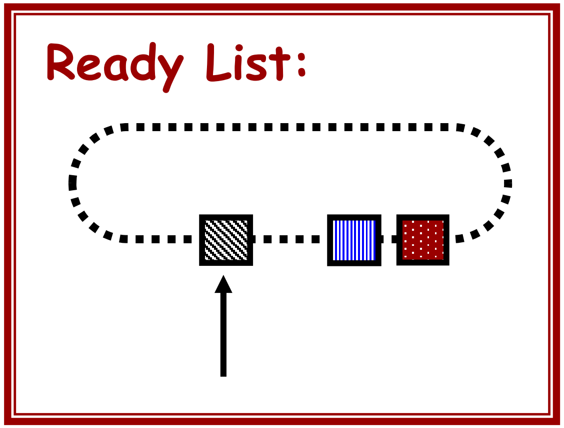

Ready List:

Round-Robin Scheduling

-

	Process	Arrival Time	Processing Time
░	1	0	3
▨	2	2	6
🟥	3	4	4
🟦	4	6	5
🟩	5	8	2

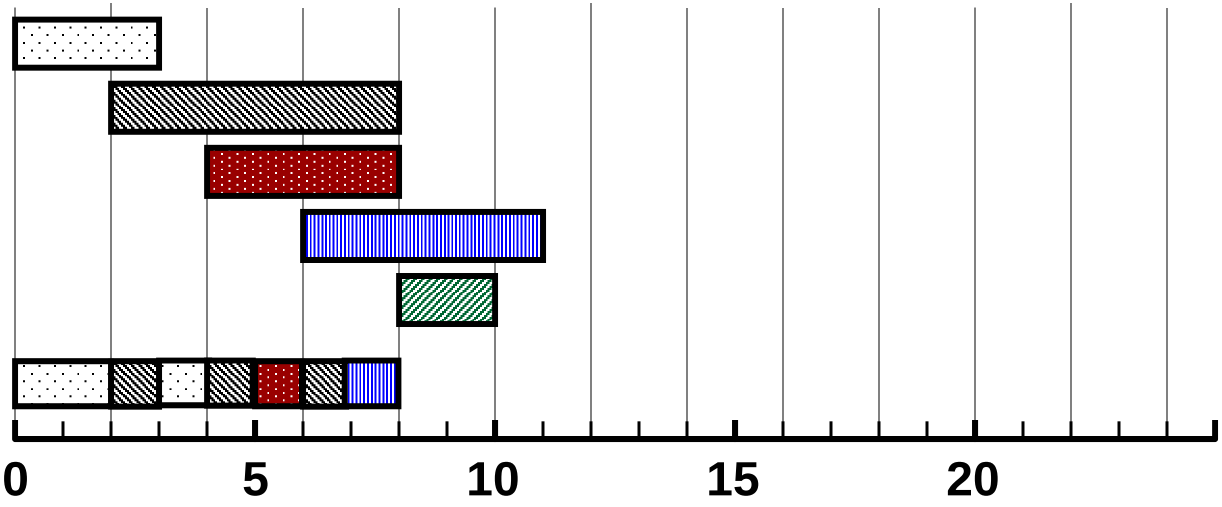



Ready List:

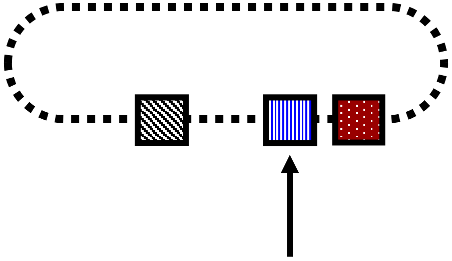
Round-Robin Scheduling

-

Process		Arrival Time	Processing Time
⬚	1	0	3
▨	2	2	6
🟥	3	4	4
🟦	4	6	5
🟩	5	8	2

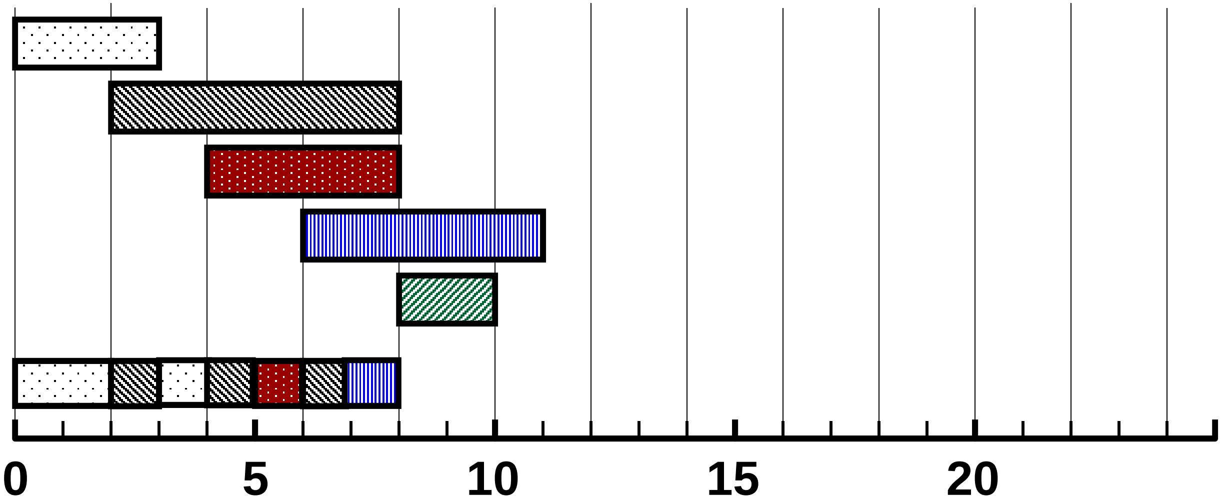


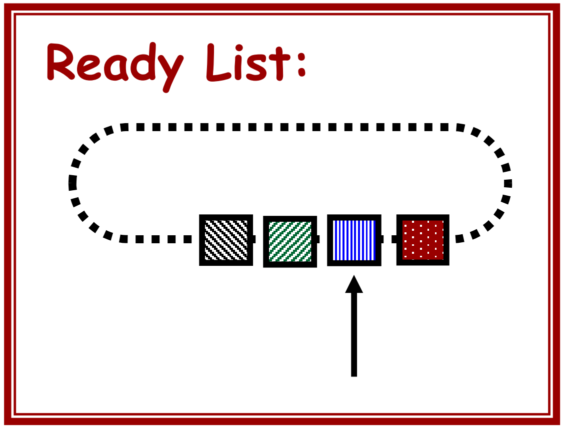
Ready List:



Round-Robin Scheduling

Process	Arrival Time	Processing Time
░ 1	0	3
▨ 2	2	6
🟥 3	4	4
🟦 4	6	5
🟩 5	8	2

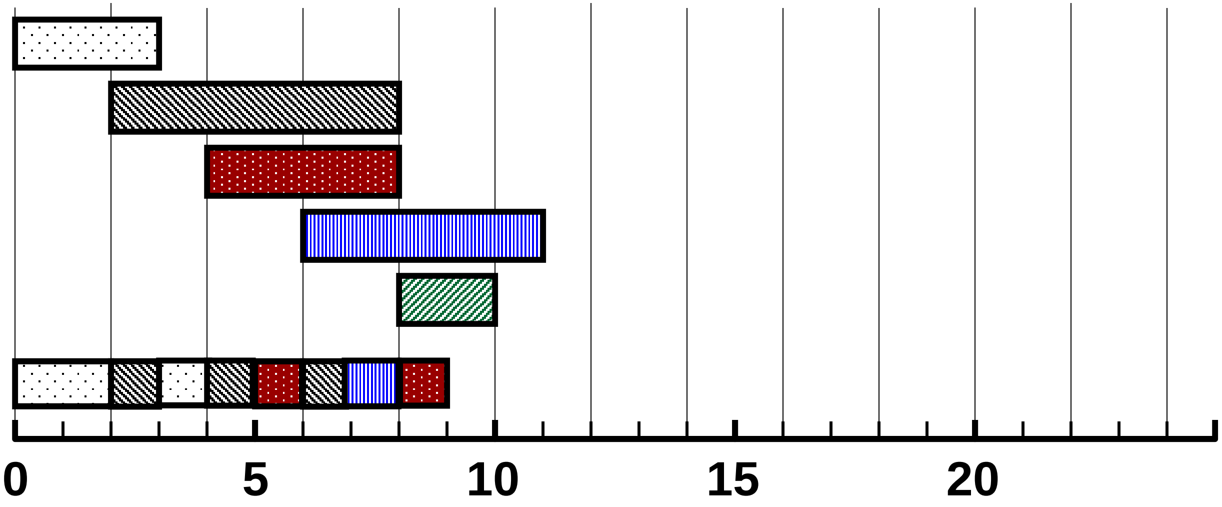


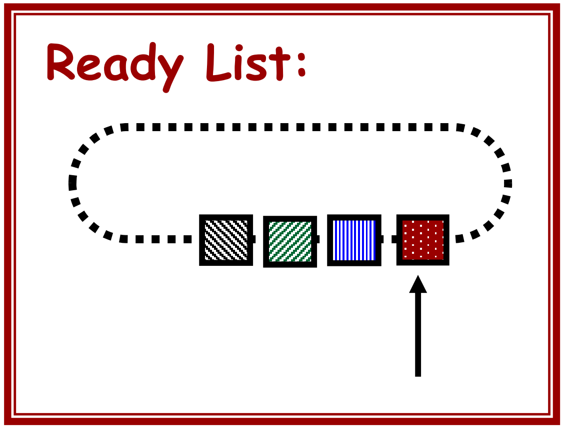

Ready List:

Round-Robin Scheduling

-

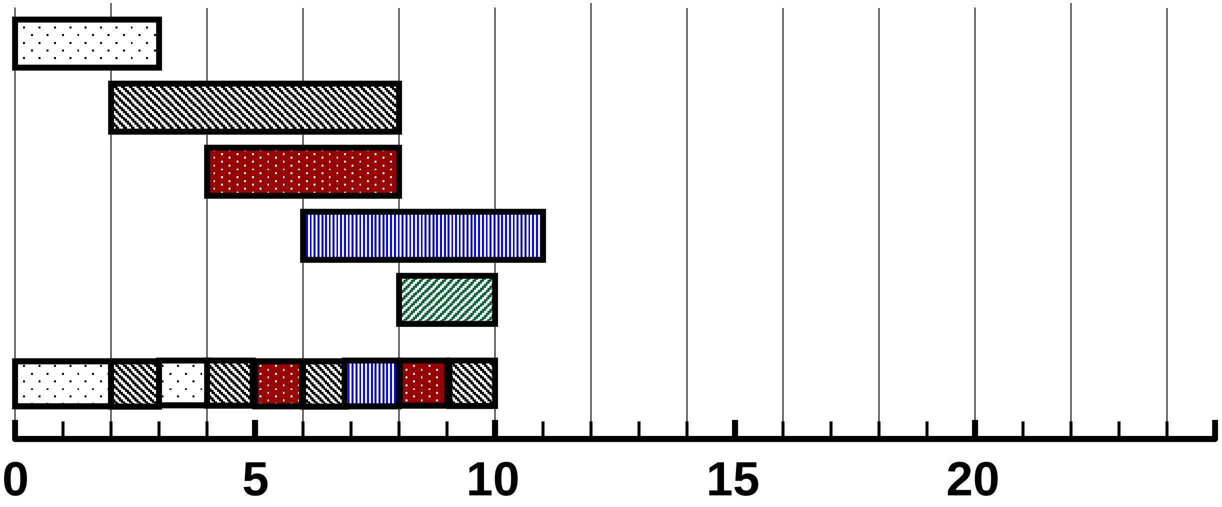
		Arrival	Processing
	Process	Time	Time
⬚	1	0	3
▨	2	2	6
🟥	3	4	4
🟦	4	6	5
🟩	5	8	2

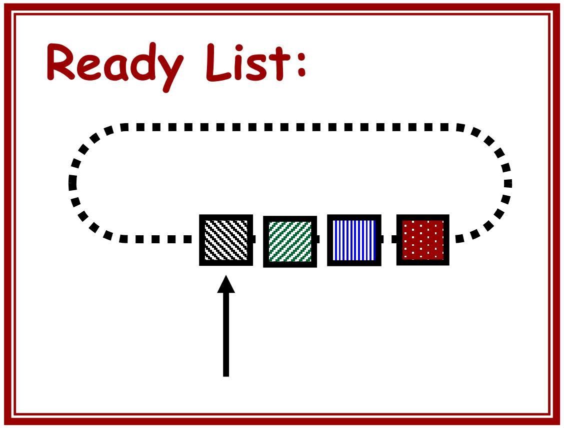




Round-Robin Scheduling

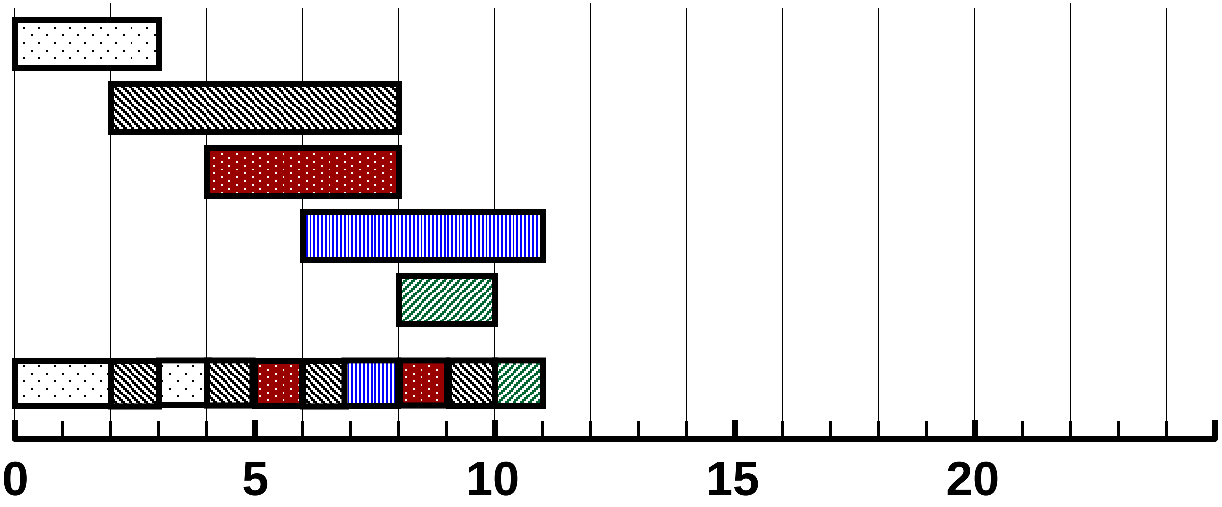
	Arrival	Processing
Process	Time	Time
⬚ 1	0	3
▨ 2	2	6
🟥 3	4	4
🟦 4	6	5
🟩 5	8	2

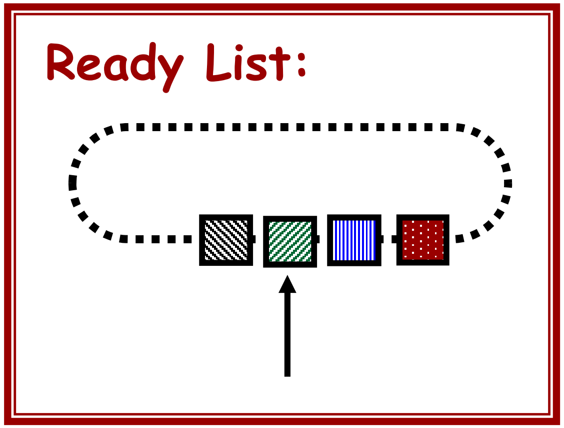




Round-Robin Scheduling

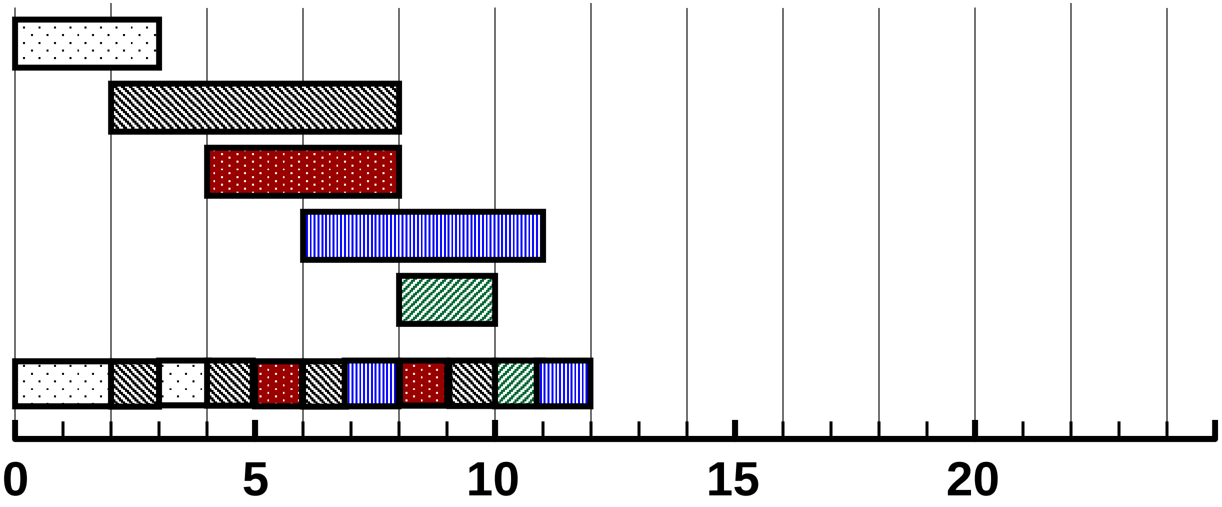
	Process	Arrival Time	Processing Time
░	1	0	3
▧	2	2	6
🟥	3	4	4
▥	4	6	5
🟩	5	8	2

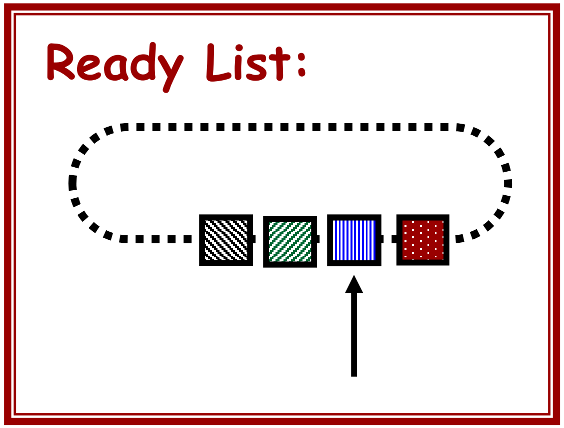




Round-Robin Scheduling

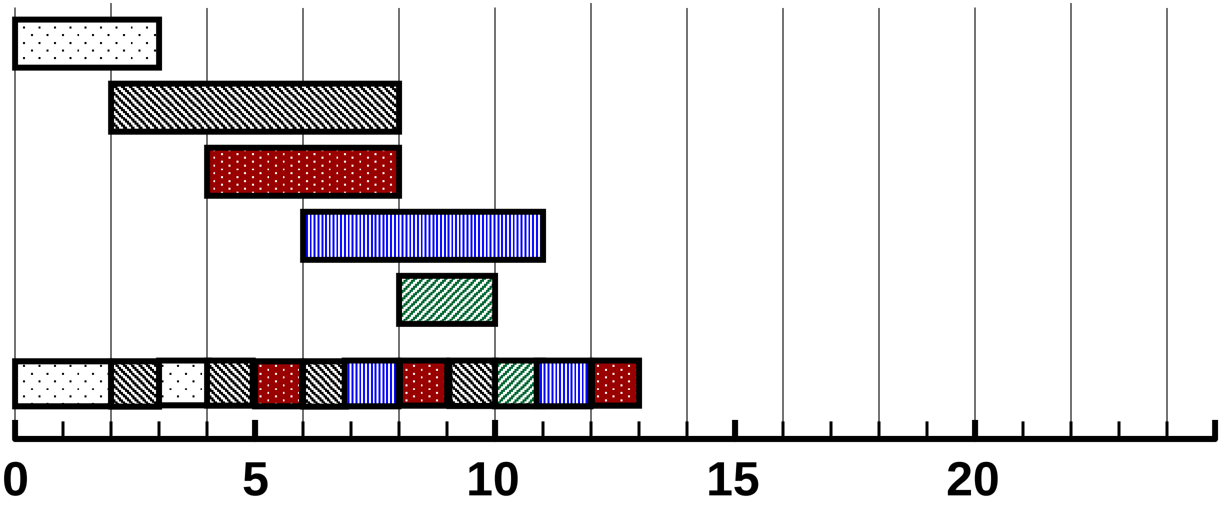
	Process	Arrival Time	Processing Time
⬚	1	0	3
▨	2	2	6
🟥	3	4	4
🟦	4	6	5
🟩	5	8	2

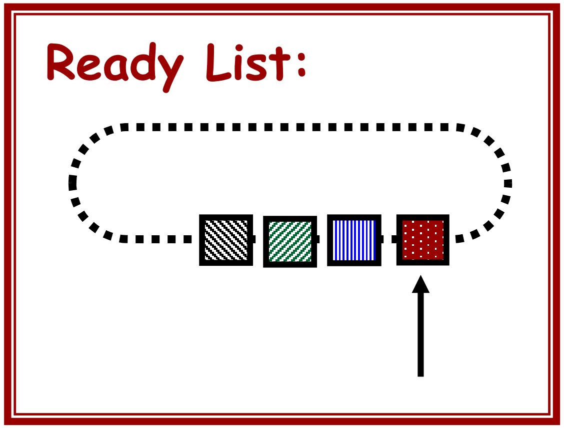




Round-Robin Scheduling

	Process	Arrival Time	Processing Time
⬚	1	0	3
▨	2	2	6
🟥	3	4	4
🟦	4	6	5
🟩	5	8	2

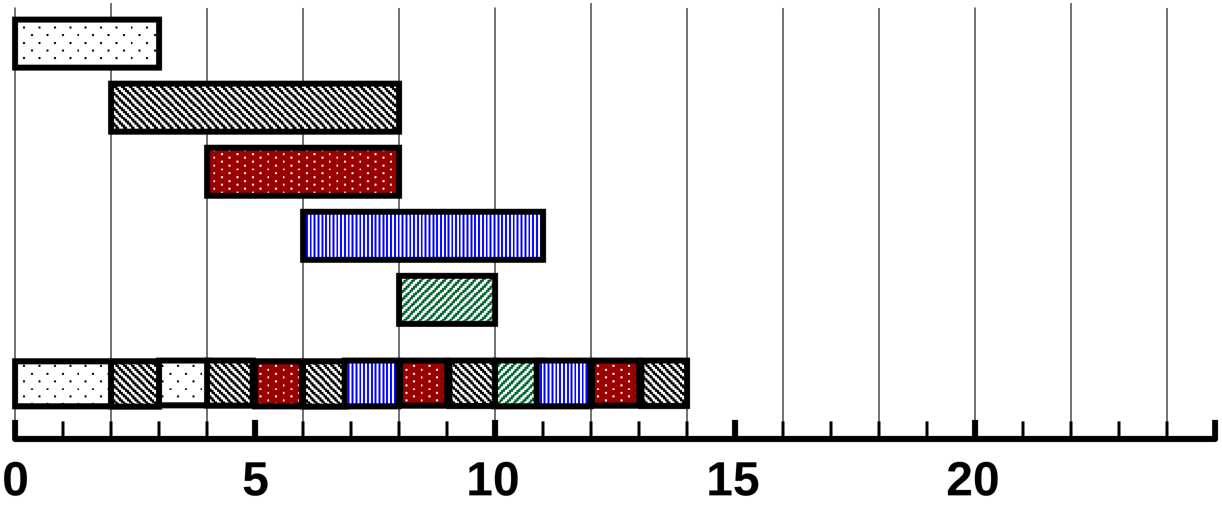


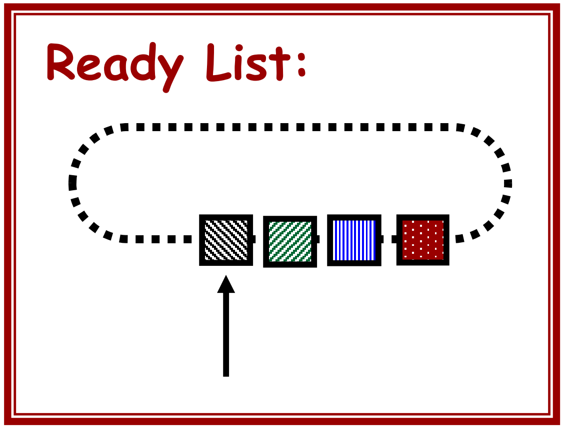

Ready List:

Round-Robin Scheduling

-

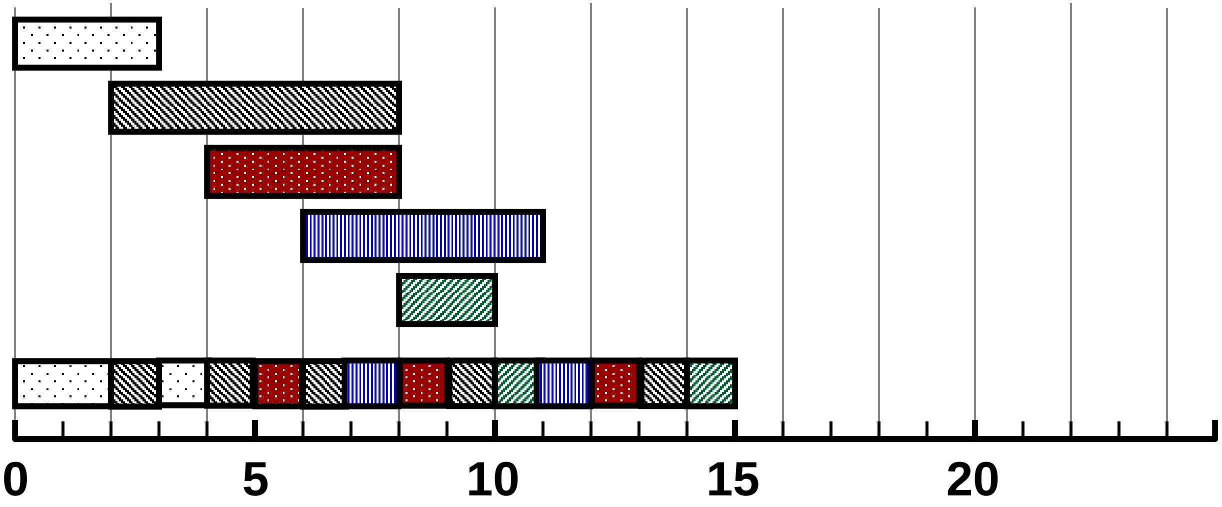
Process		Arrival Time	Processing Time
⬚	1	0	3
▨	2	2	6
🟥	3	4	4
▥	4	6	5
🟩	5	8	2

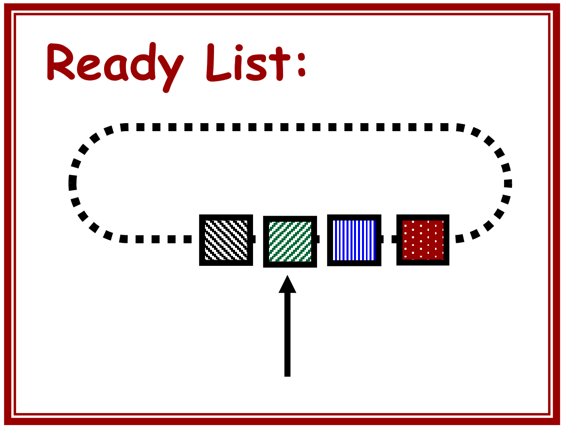



Ready List:

Round-Robin Scheduling

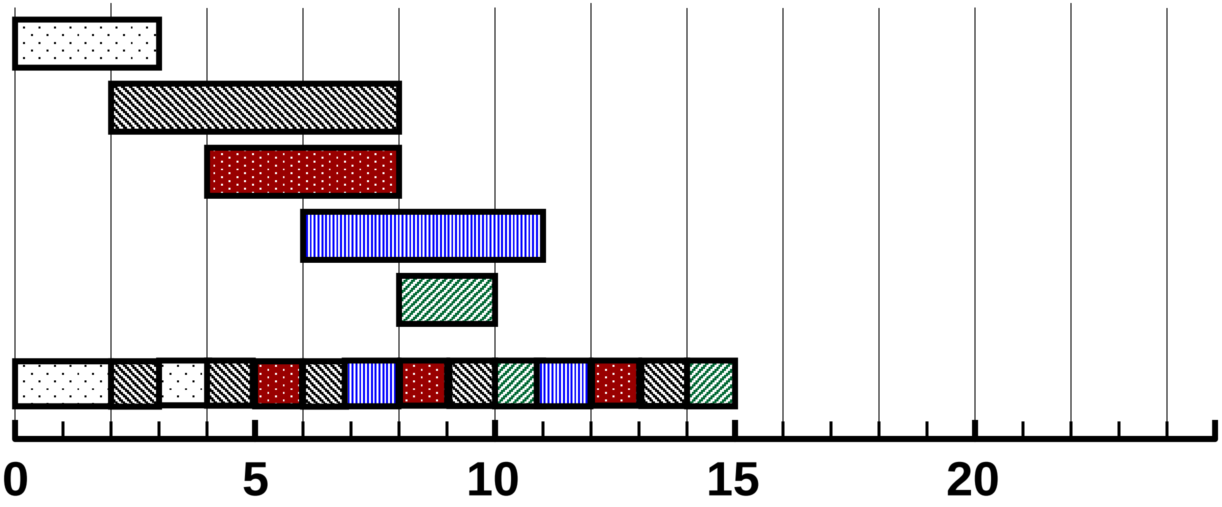
	Process	Arrival Time	Processing Time
░	1	0	3
▨	2	2	6
🟥	3	4	4
🟦	4	6	5
🟩	5	8	2

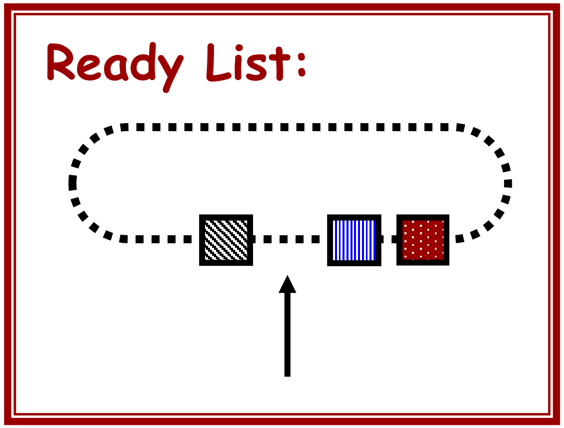



Ready List:

Round-Robin Scheduling

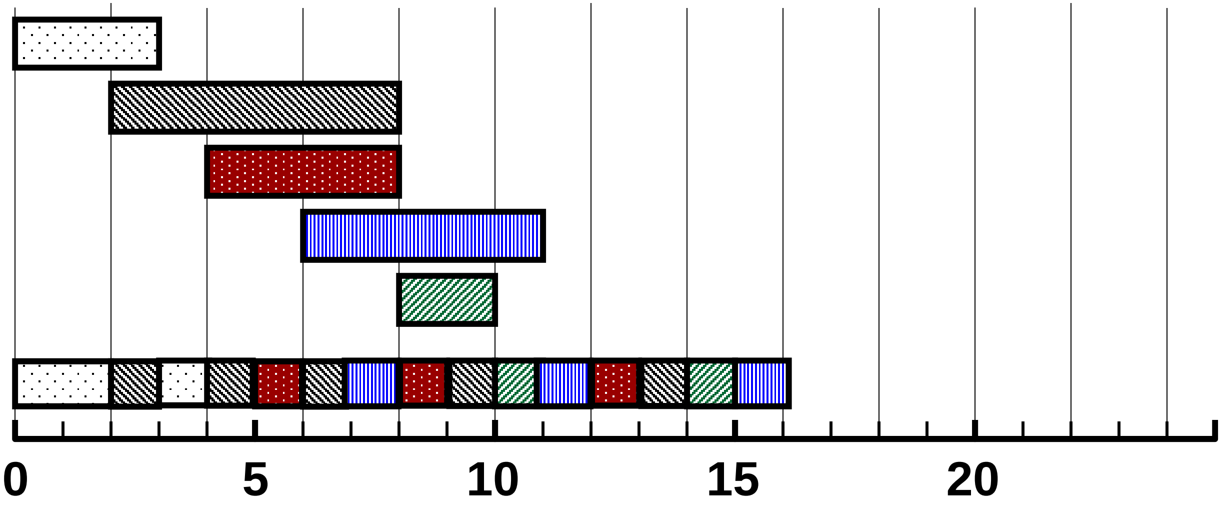
	Process	Arrival Time	Processing Time
⬚	1	0	3
▨	2	2	6
🟥	3	4	4
▥	4	6	5
🟩	5	8	2

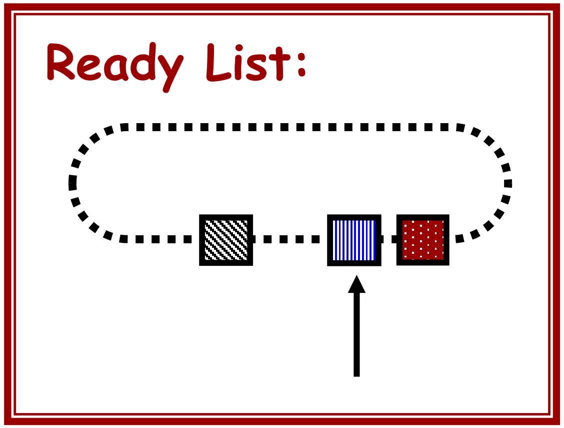



Ready List:

Round-Robin Scheduling

Process	Arrival Time	Processing Time
⬚ 1	0	3
▨ 2	2	6
🟥 3	4	4
🟦 4	6	5
🟩 5	8	2

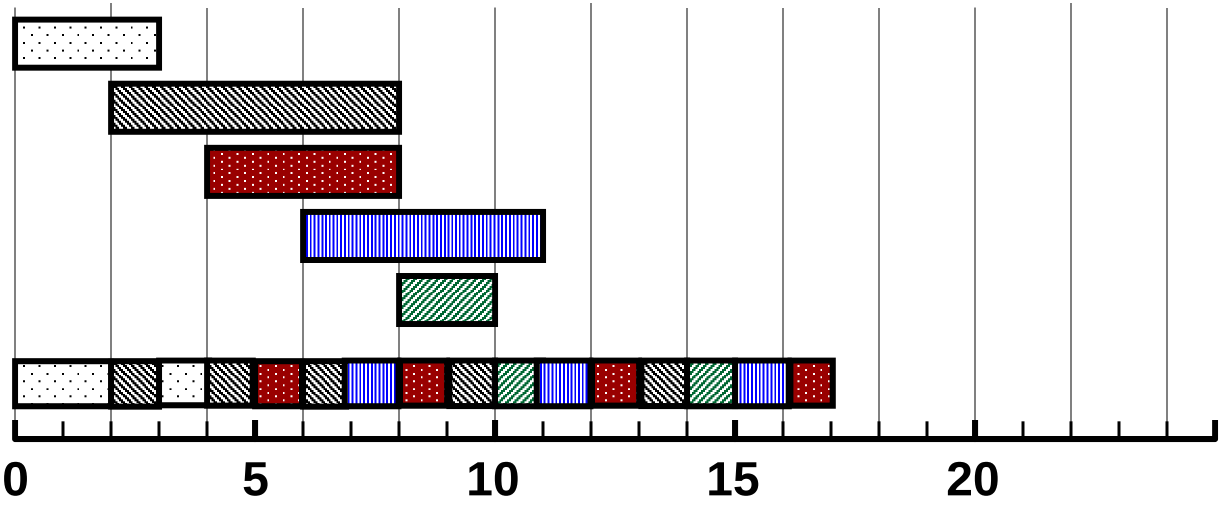


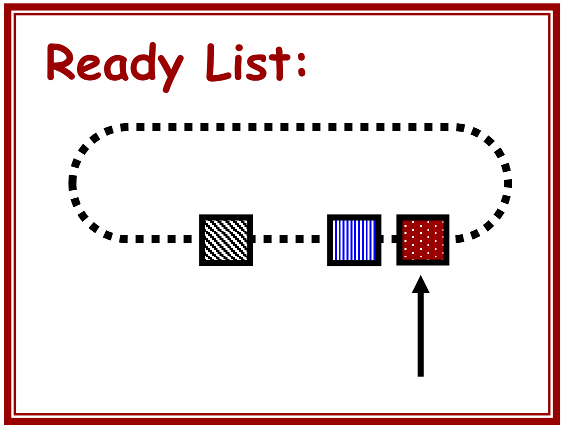

Ready List:

Round-Robin Scheduling

-

	Process	Arrival Time	Processing Time
░	1	0	3
▨	2	2	6
🟥	3	4	4
🟦	4	6	5
🟩	5	8	2

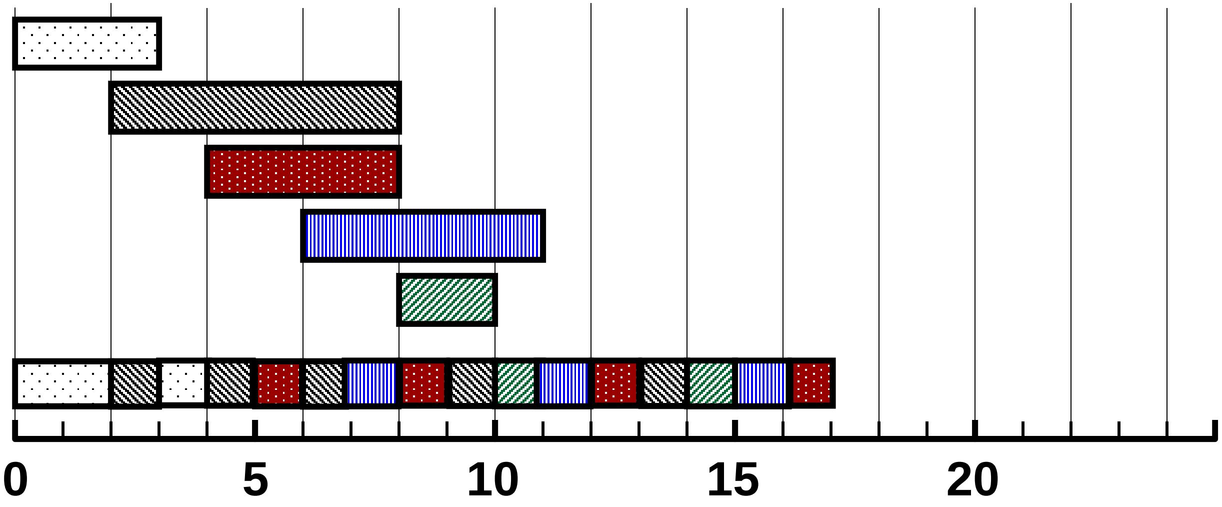


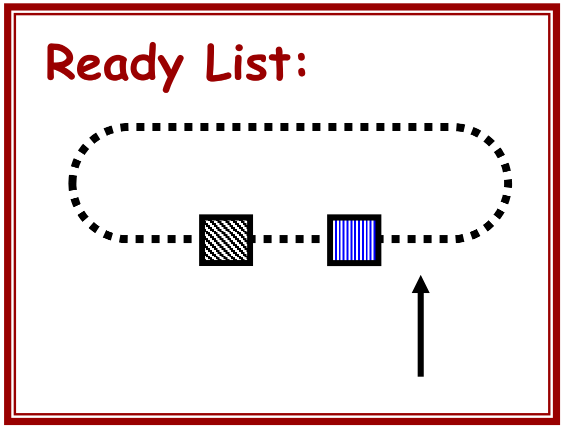


Round-Robin Scheduling

-

	Process	Arrival	Processing
		Time	Time
⬚	1	0	3
▨	2	2	6
🟥	3	4	4
🟦	4	6	5
🟩	5	8	2

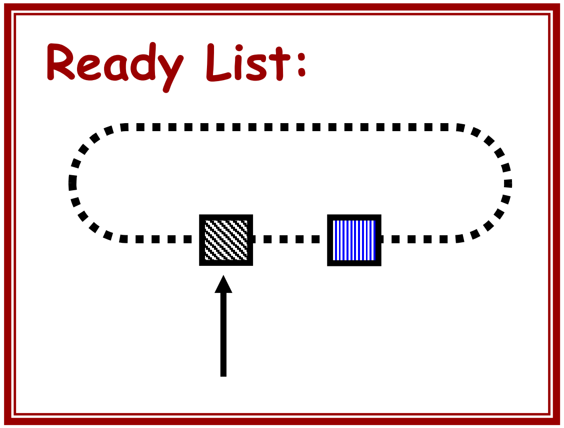


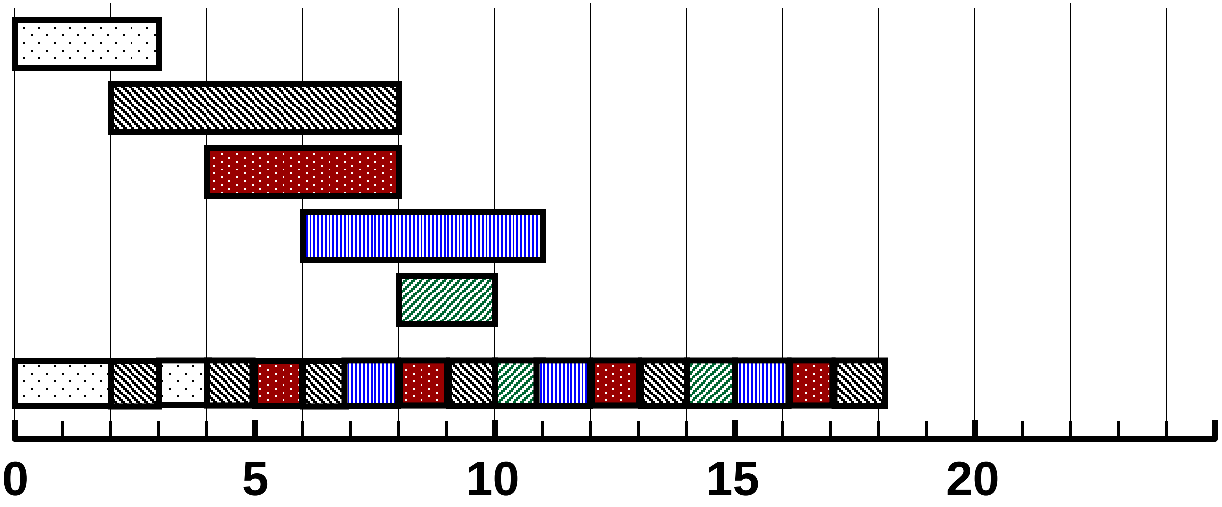

Ready List:

Round-Robin Scheduling

-

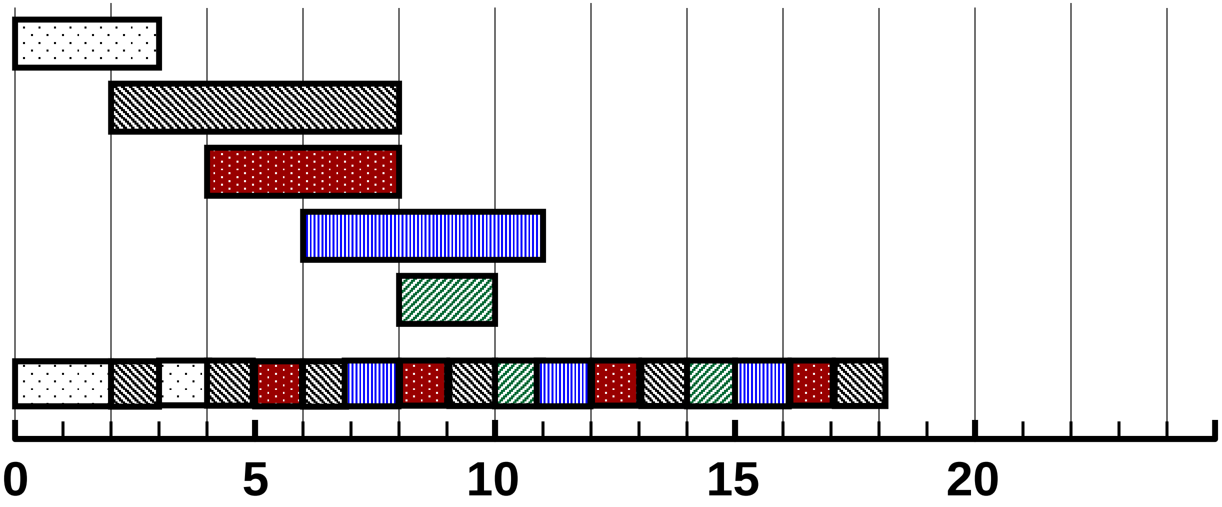
	Process	Arrival Time	Processing Time
⬚	1	0	3
▧	2	2	6
▦	3	4	4
▥	4	6	5
▨	5	8	2

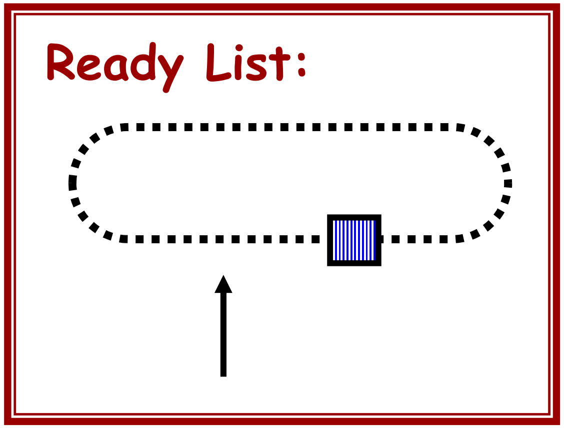




Round-Robin Scheduling

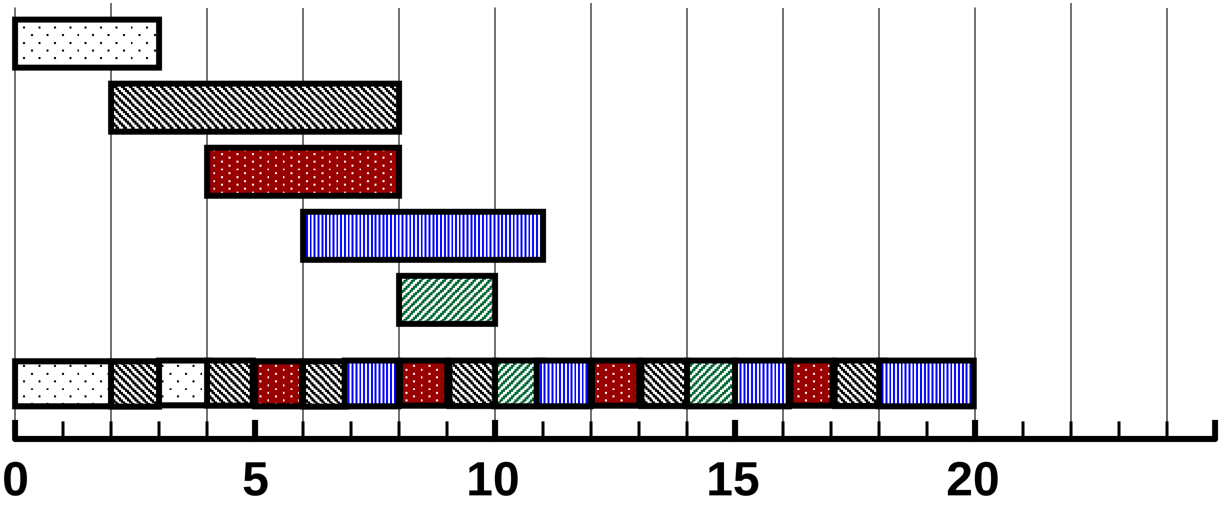
	Process	Arrival Time	Processing Time
░	1	0	3
▨	2	2	6
🟥	3	4	4
🟦	4	6	5
🟩	5	8	2

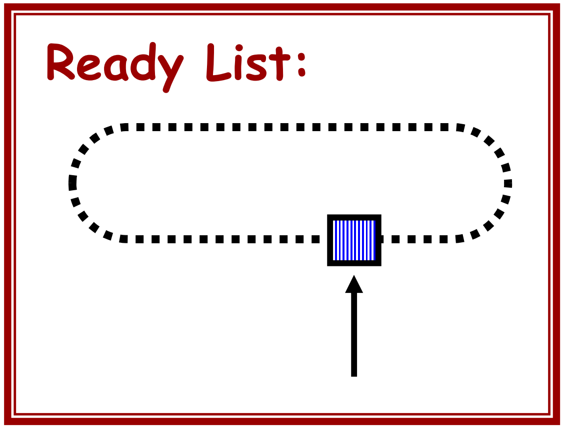



Ready List:

Round-Robin Scheduling

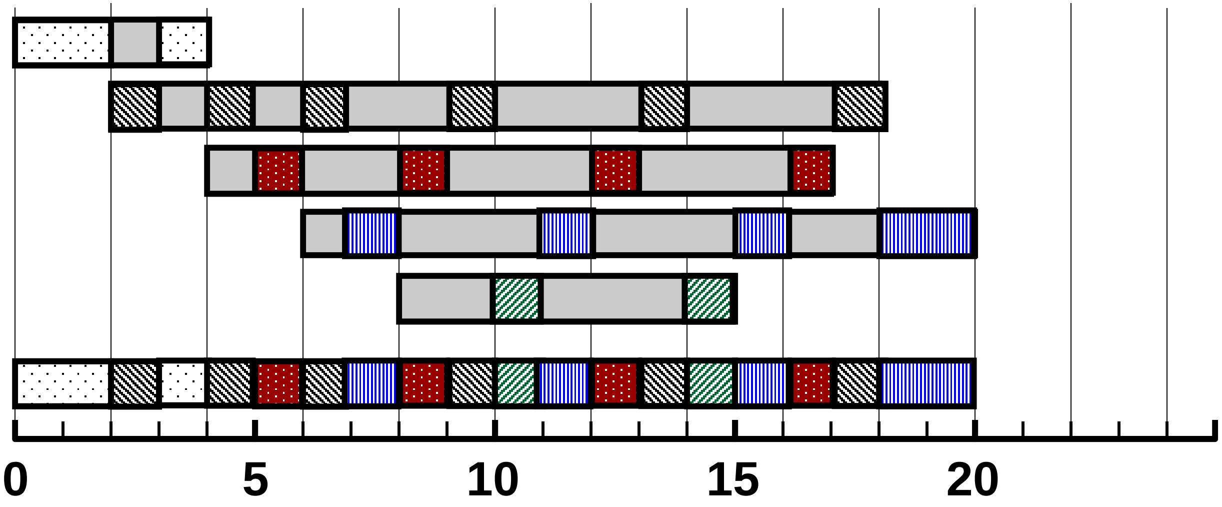
	Process	Arrival	Processing
		Time	Time
⬚	1	0	3
▨	2	2	6
🟥	3	4	4
🟦	4	6	5
🟩	5	8	2




Ready List:

Round-Robin Scheduling

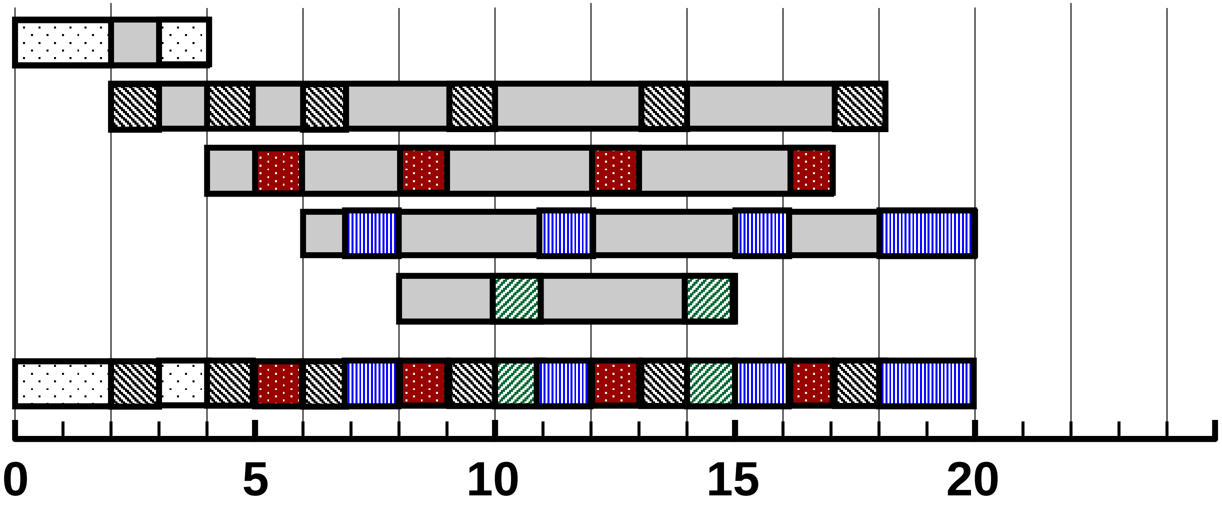
Process		Arrival Time	Processing Time
⬚	1	0	3
▨	2	2	6
🟥	3	4	4
🟦	4	6	5
🟩	5	8	2



Round-Robin Scheduling

-

	Process	Arrival	Processing		Turnaround
		Time	Time	Delay	Time
⬚	1	0	3	1	4
▨	2	2	6	10	16
🟥	3	4	4	9	13
🟦	4	6	5	9	14
🟩	5	8	2	5	7



Round-Robin Scheduling

- **Effectiveness of round-robin depends on**
  - The number of jobs, and
  - The size of the time quantum.
- **Large # of jobs means that the time between scheduling of a single job increases**
  - Slow responses
- **Larger time quantum means that the time between the scheduling of a single job also increases**
  - Slow responses
- **Smaller time quantum means higher processing rates but also more overhead!**

Scheduling in general purpose systems

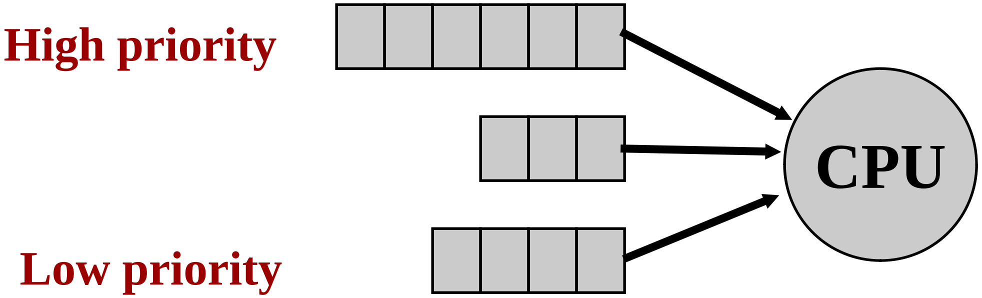
Priority scheduling

- **Assign a priority (number) to each process**
- **Schedule processes based on their priority**
- **Higher priority processes get more CPU time**

- **Managing priorities**
  - Can use “nice” to reduce your priority
  - Can periodically adjust a process’ priority
    - **Prevents starvation of a lower priority process**
    - **Can improve performance of I/O-bound processes by basing priority on fraction of last quantum used**

Multi-Level Queue Scheduling

- Multiple queues, each with its own priority.
  - Equivalently: each priority level has its own ready queue
- Within each queue...Round-robin scheduling.
- Simplist Approach:
  - A Process's priority is fixed & unchanging



Multi-Level Feedback Queue Scheduling

- **Problem: Fixed priorities are too restrictive**
  - Processes exhibit varying ratios of CPU to I/O times.
- **Dynamic Priorities**
  - Priorities are altered over time, as process behavior changes!

Multi-Level Feedback Queue Scheduling

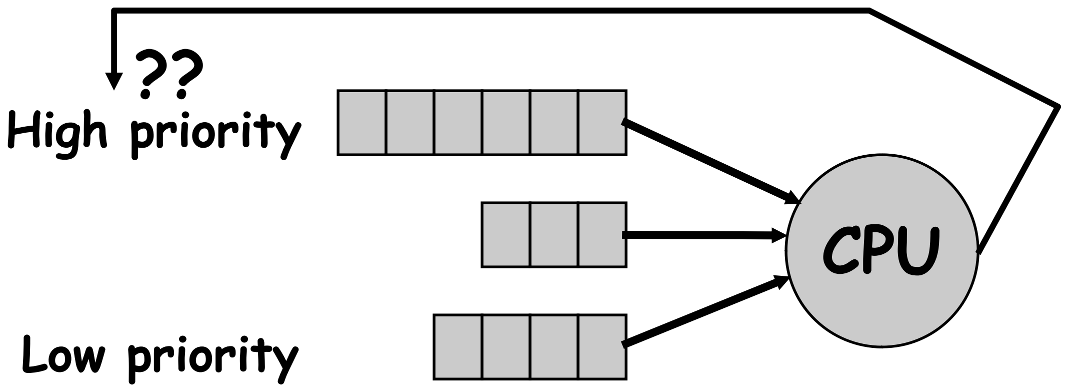
- **Problem: Fixed priorities are too restrictive**
  - Processes exhibit varying ratios of CPU to I/O times.
- **Dynamic Priorities**
  - Priorities are altered over time, as process behavior changes!
- **Issue: When do you change the priority of a process and how often?**

Multi-Level Feedback Queue Scheduling

- **Problem: Fixed priorities are too restrictive**
  - Processes exhibit varying ratios of CPU to I/O times.
- **Dynamic Priorities**
  - Priorities are altered over time, as process behavior changes!
- **Issue: When do you change the priority of a process and how often?**
- **Solution: Let the amount of CPU used be an indication of how a process is to be handled**
  - Expired time quantum → more processing needed
  - Unexpired time quantum → less processing needed
- **Adjusting quantum and frequency vs. adjusting priority?**

Multi-Level Feedback Queue Scheduling

- *n* priority levels, round-robin scheduling within a level
- *Quanta* increase as priority decreases
- Jobs are demoted to lower priorities if they do not complete within the current quantum



Multi-Level Feedback Queue Scheduling

- ***Details, details, details...***
  - Starting priority?
  - Frequency of moving between priorities?
  - How long should the time quantum be?

Lottery Scheduling

- **Scheduler gives each thread some lottery tickets**
- **To select the next process to run...**
  - The scheduler randomly selects a lottery number
  - The winning process gets to run

- ***Example*** **Thread A gets 50 tickets**
  Thread B gets 15 tickets
  Thread C gets 35 tickets
  There are 100 tickets outstanding.

Lottery Scheduling

- Scheduler gives each thread some lottery tickets.
- To select the next process to run...
  - The scheduler randomly selects a lottery number
  - The winning process gets to run
- *Example* Thread A gets 50 tickets → 50% of CPU

  Thread B gets 15 tickets → 15% of CPU

  Thread C gets 35 tickets → 35% of CPU

  There are 100 tickets outstanding.

Lottery Scheduling

- Scheduler gives each thread some lottery tickets.
- To select the next process to run...
- The scheduler randomly selects a lottery number
- The winning process gets to run
- *Example* Thread A gets 50 tickets → 50% of CPU
  Thread B gets 15 tickets → 15% of CPU
  Thread C gets 35 tickets → 35% of CPU
  There are 100 tickets outstanding.

  - Flexible
  - Fair
  - Responsive

A Brief Look at Real-Time Systems

- **Assume processes are relatively periodic**
  - Fixed amount of work per period (e.g. sensor systems or multimedia data)

A Brief Look at Real-Time Systems

- **Assume processes are relatively periodic**
  - Fixed amount of work per period (e.g. sensor systems or multimedia data)
- **Two "main" types of schedulers...**
- **Rate-Monotonic Schedulers**
- **Earliest-Deadline-First Schedulers**

A Brief Look at Real-Time Systems

- **Assume processes are relatively periodic**
  - Fixed amount of work per period (e.g. sensor systems or multimedia data)
- **Two “main” types of schedulers...**
- **Rate-Monotonic Schedulers**
  - Assign a *fixed, unchanging* priority to each process
  - No dynamic adjustment of priorities
    - **Less aggressive allocation of processor**
- **Earliest-Deadline-First Schedulers**
  - Assign dynamic priorities based upon deadlines

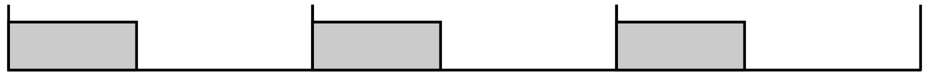
A Brief Look at Real-Time Systems

- **Typically real-time systems involve several steps (that aren't in traditional systems)**
- **Admission control**
  - All processes must ask for resources ahead of time.
  - If sufficient resources exist, the job is "admitted" into the system.
- **Resource allocation**
  - Upon admission...
  - the appropriate resources need to be reserved for the task.
- **Resource enforcement**
  - Carry out the resource allocations properly

Rate Monotonic Schedulers

- **For preemptable, periodic processes (tasks)**
- **Assigns a fixed priority to each task**
  - T = The period of the task
  - C = The amount of processing per task period

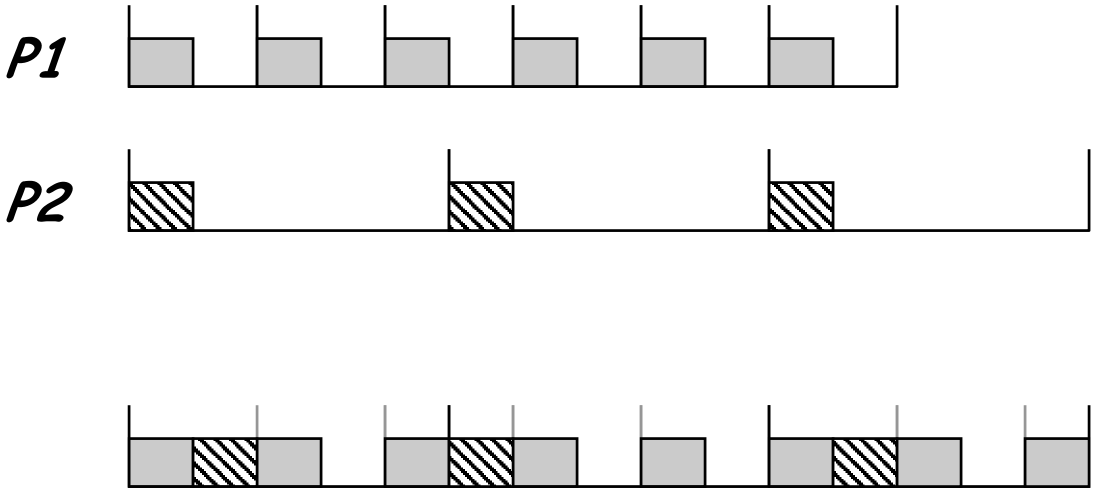
Process P1



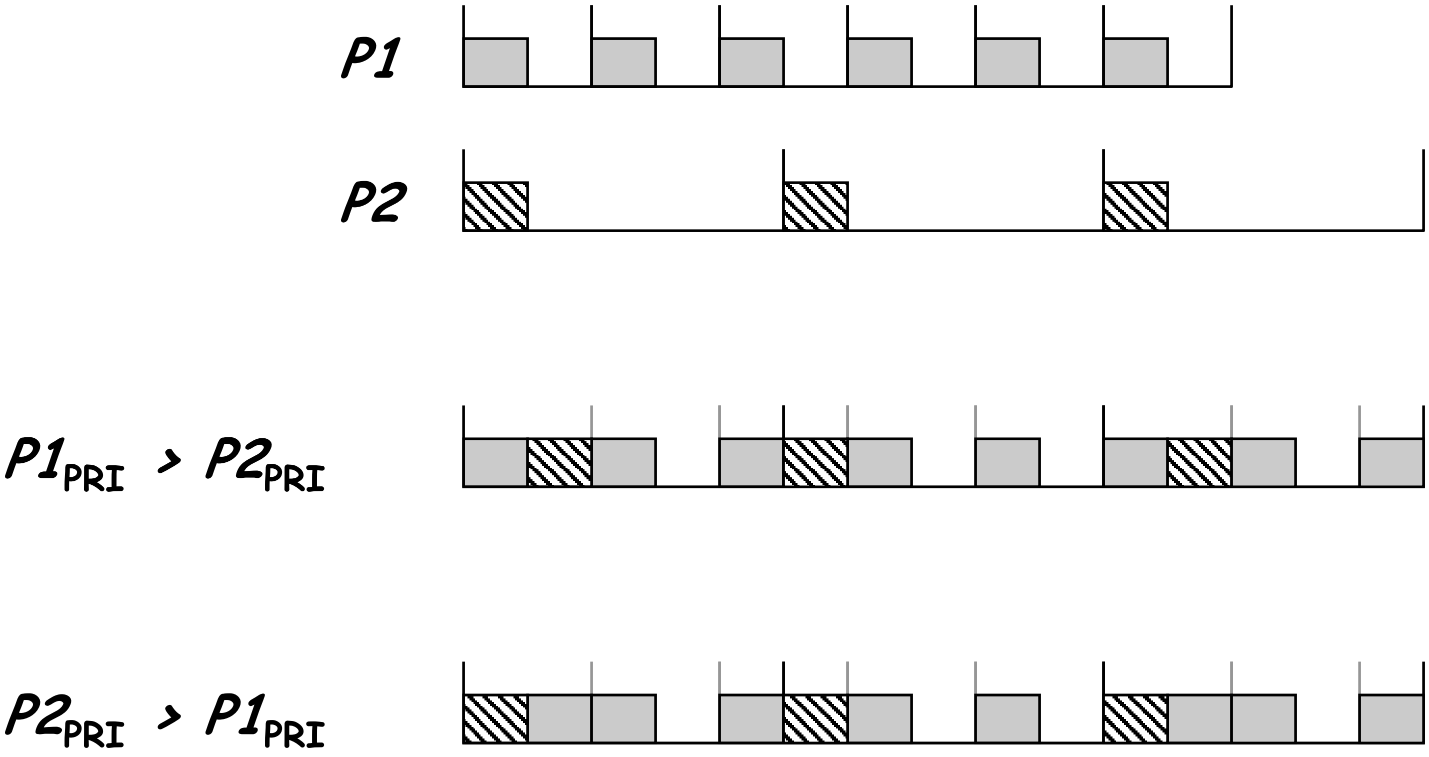
T = 1 second
C = 1/2 second / period

- **In RMS scheduling, the question to answer is...**
  - *What priority should be assigned to a given task?*

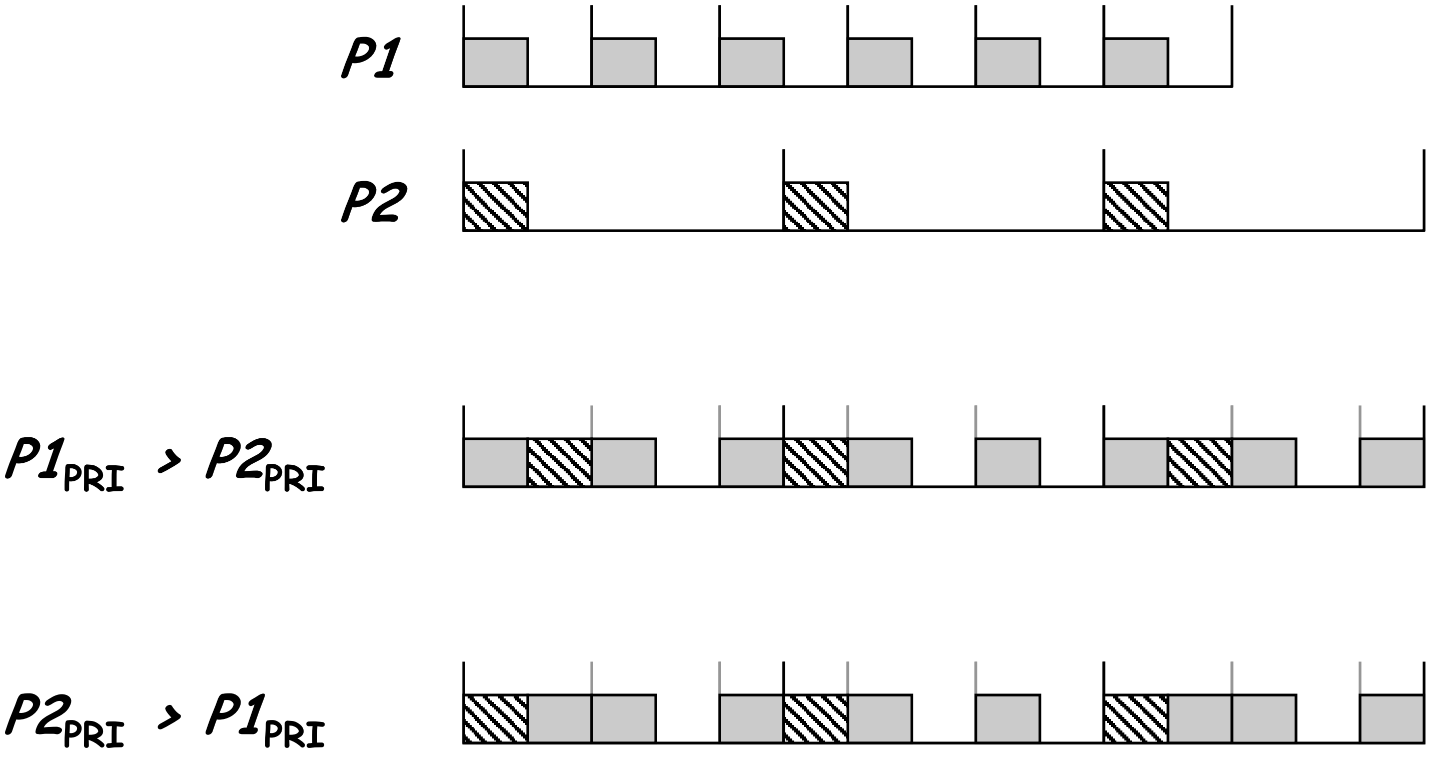
Rate Monotonic Schedulers



Rate Monotonic Schedulers



Rate Monotonic Schedulers



Which is best?

Rate Monotonic Schedulers

- **Assign shortest period tasks to the highest priorities**
- **Admission control is difficult...**
  - < 69% There is a standard formula
  - > 69% It may be possible, but need to do more complex analysis

Rate Monotonic Schedulers

- ***Assumptions:***
  - Processes complete (yield) within their period
  - Independent processes
  - Same CPU requirements per burst
  - Other non-periodic processes have no deadlines
  - Instantaneous preemption with no overhead

Earliest Deadline First

- ***When processes do not have periodic execution or constant CPU requirements...***
- ***When processes have deadline specifications...***

- **Unlike RMS, EDF uses dynamic priorities (based upon earliest deadline first)**

  (+) 100% processor utilization ...?

  (-) Need to keep track of deadlines

- **Admission Control**
  - Just check to see if 100% processor utilization.
  - Sum the C_i/T_i's and see if less than or equal to 1
  - What about overhead?

Quiz

- What are the main tasks of the scheduler?
- What is the difference between preemptive and non-preemptive scheduling?
- What is the advantage of a shorter scheduling quantum?
- What is the advantage of a longer scheduling quantum?
- Why is feedback scheduling useful for interactive jobs?
- Are these scheduling policies subject to starvation?
  - Shortest Job First scheduling?
  - Round Robin scheduling?
  - First Come First Served scheduling?
  - Priority scheduling?
  - Earliest Deadline First scheduling?