



# OPERATING SYSTEMS

Module2\_Part9

Textbook : Operating Systems Concepts by Silberschatz



# Scheduling algorithms

CPU scheduling deals with the problem of deciding which of the processes in the ready queue

is to be allocated the CPU. There are many different CPU-scheduling algorithms. Some of them are

**First-Come, First-Served Scheduling**

**Shortest-Job-First Scheduling**

**shortest-remaining-time-first**

**Priority Scheduling**

**Round-Robin Scheduling**



# Round robin scheduling

- Round Robin(RR) scheduling algorithm is mainly designed for time-sharing systems. This algorithm is similar to FCFS scheduling, but in Round Robin(RR) scheduling, preemption is added which enables the system to switch between processes.
- A small unit of time, called a **time quantum** or time slice, is defined. Each process is assigned a time slice or time quantum for execution.
- Once a process is executed for the given time period, that process is preempted and another process executes for the given time period.

# Round robin

To implement RR scheduling,

we keep the ready queue as a FIFO queue of processes.

New processes are added to the tail of the ready queue.

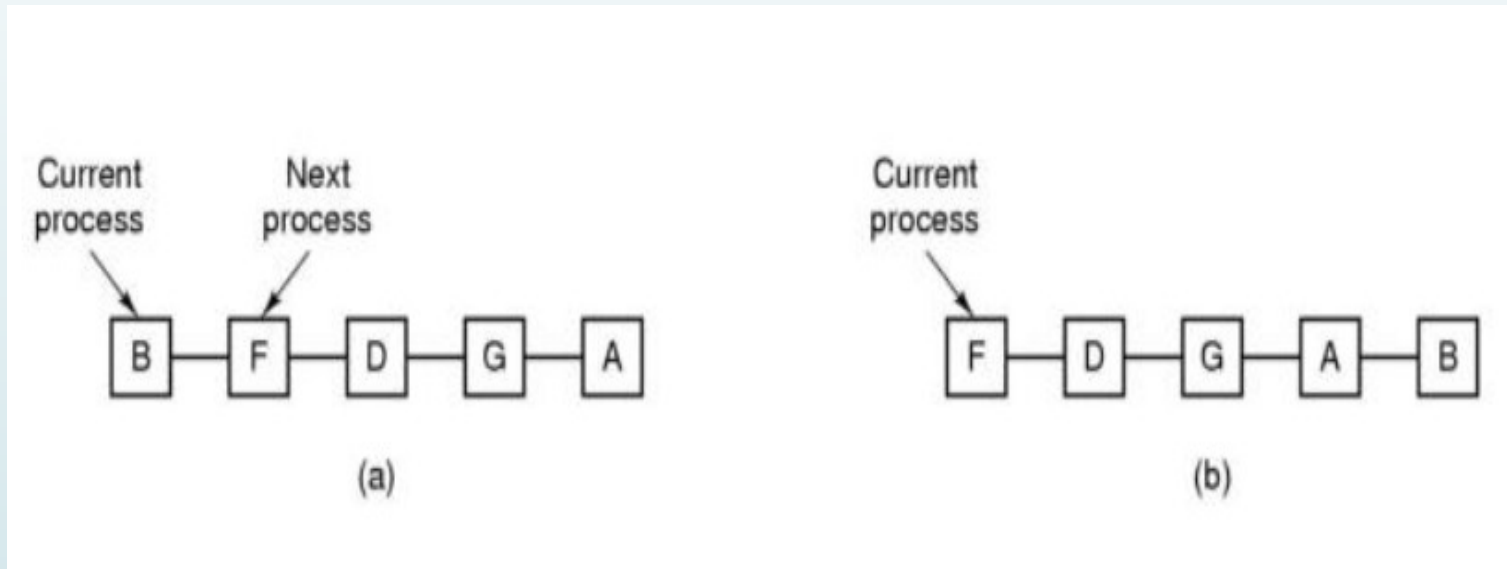
The CPU scheduler picks the first process from the ready queue, sets a timer to interrupt after 1 time quantum, and dispatches the process.

One of two things will then happen.

The process may have a CPU burst of less than 1 time quantum. In this case, the process itself will release the CPU voluntarily. The scheduler will then proceed to the next process in the ready queue.

if the CPU burst of the currently running process is longer than 1 time quantum, the timer will go off and will cause an interrupt to the operating system. A context switch will be executed, and the process will be put at the tail of the ready queue. The CPU scheduler will then select the next process in the ready queue

# Round robin scheduling



←  
Front of the  
queue

**When B completes its time slice, put back to the tail of the queue. F will run. When F completes F put back to the tail of the queue. Then D runs and so on**

# Round robin scheduling

The average waiting time under the RR policy is often long. Consider the following set of processes that arrive at time 0, with the length of the CPU burst given in milliseconds:

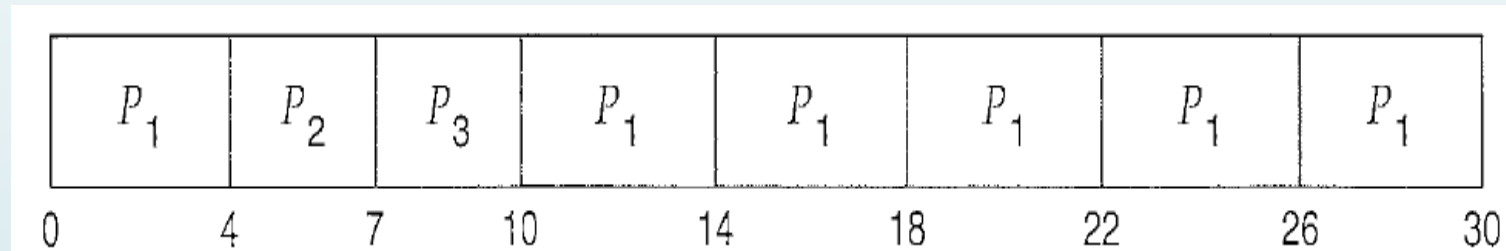
- Example with 3 processes

	<u>Process</u>	<u>Burst Time</u>
	$P_1$	24
	$P_2$	3
	$P_3$	3

- If we use a time quantum of 4 milliseconds, then process  $P_1$  gets the first 4 milliseconds. it requires another 20 milliseconds, it is preempted after the first time quantum
- CPU is given to the next process in the queue, process  $P_2$  . Process  $P_2$  does not need 4 milliseconds, so it quits before its time quantum expires.
- The CPU is then given to the next process, process  $P_3$ . Once each process has received 1 time quantum, the CPU is returned to process  $P_1$  for an additional time quantum.

# Round robin scheduling

□ The resulting RR schedule is as follows:



Let's calculate the average waiting time for the above schedule.  $P_1$  waits for 6 milliseconds (10- 4),  $P_2$  waits for 4 milliseconds, and  $P_3$  waits for 7 milliseconds. Thus, the average waiting time is  $17/3 = 5.66$  milliseconds.

# Round robin scheduling

- If there are  $n$  processes in the ready queue and the time quantum is  $q$ , then each process gets 1 *timeslice* of the CPU time in chunks of at most  $q$  time units. Each process must wait no longer than  $(n - 1) \times q$  time units until its next time quantum. For example, with five processes and a time quantum of 20 milliseconds, each process will get up to 20 milliseconds every 100 milliseconds.

The performance of the RR algorithm depends heavily on the size of the time quantum.

At one extreme, if the time quantum is extremely large, the RR policy is the same as the

FCFS policy.

Setting too short causes too many process switched and lowers cpu efficiency

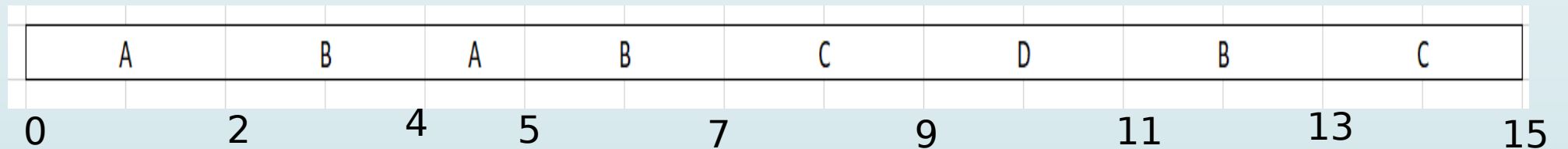
We have to take time quantum in between



# Round robin scheduling

For the processes listed draw gantt chart illustrating their execution(quantum=2)

process	Arrival time	Processing time
A	0.000	3
B	1.001	6
C	4.001	4
D	6.001	2

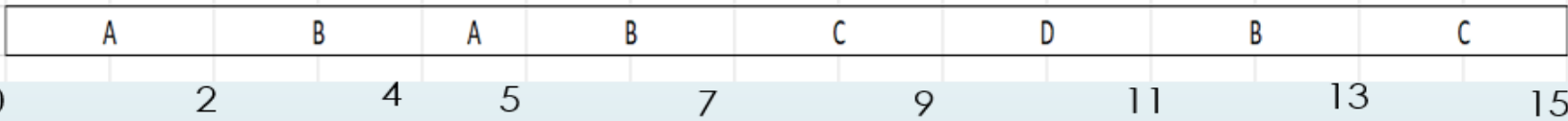


When Process A's first time quantum expires process B runs. At time 4 process A restarts process B returns to the ready queue. At time 4.001 process C enters the ready queue after B. At time 6 process D enters the ready queue after C. Starting at &process C D B and C runs in sequence.

# Round robin scheduling

- For the process listed what is the average turn around time?

Process	Arrival time	Processing time	Completion time	Turn around time
A	0.000	3	5	5
B	1.001	6	13	12
C	4.001	4	15	11
D	6.001	2	11	5



Turn around time=completion time -arrival time

Average turn around time= $((5-0)+(13-1)+(15-4)+(11-6))/4 = 8.25$

## SRTF

□ For the processes listed what is the waiting time for each process?

Process	Arrival time	Processing time	Completion time	Turn around time	Waiting time
A	0.001	3	5	5	0
B	1.001	6	13	12	8
C	4.001	4	15	11	0
D	6.001	2	11	5	2

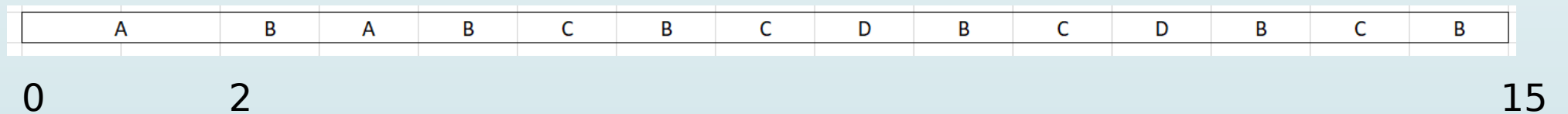
Waiting time = turn around time - execution time

A:  $(5-3)=2$     B:  $(12-6)=6$     C:  $(11-4)=7$     D:  $(5-2)=3$

# Round robin scheduling

For the processes listed draw gantt chart illustrating their execution(quantum=1)

process	Arrival time	Processing time
A	0.000	3
B	1.001	6
C	4.001	4
D	6.001	2



Remember here A runs two times, At 1 A runs again. B does not arrive until 1.001

Do as an exercise average turn around time and waiting time

# Time Quantum and Context Switch Time

## The effect of context switching on the performance of RR scheduling.

Assume, that we have only one process of 10 time units.

If the quantum is 12 time units, the process finishes in less than 1 time quantum, with no overhead.

If the quantum is 6 time units, however, the process requires 2 quanta, resulting in a context switch.

If the time quantum is 1 time unit, then nine context switches will occur, slowing the execution of the process accordingly

Although the time quantum should be large compared with the context switch time, it should not be too large. Time quantum  $q$  should be large compared to context switch time.  $q$  usually 10 milliseconds to 100 milliseconds, Context switch  $< 10$  microseconds

