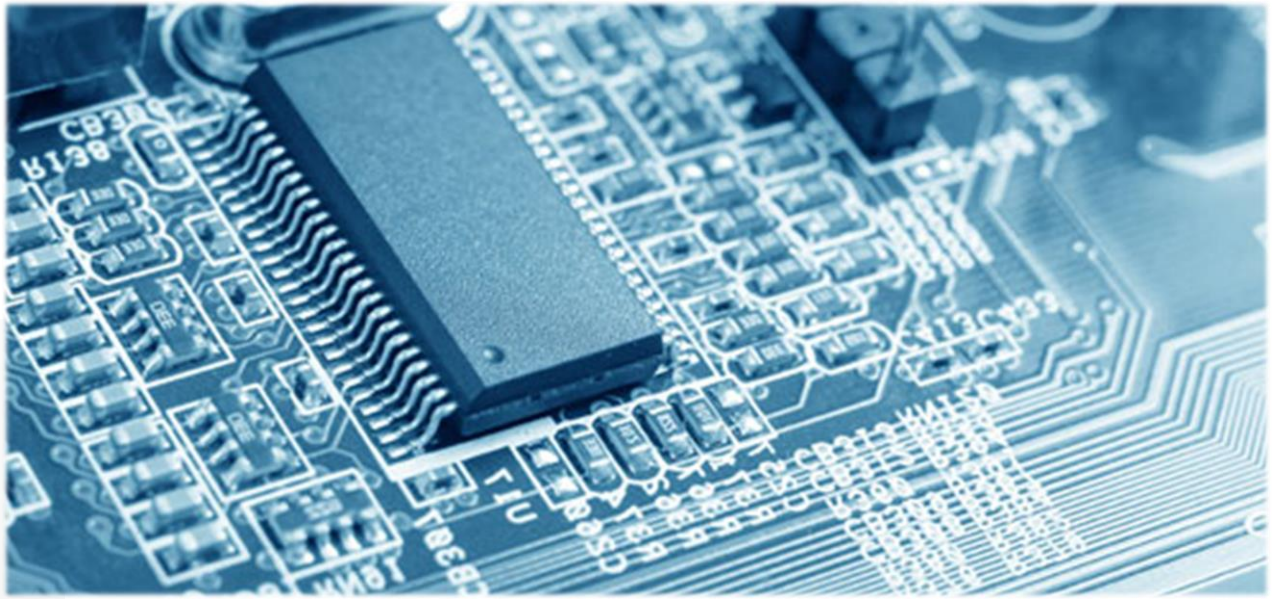
NAZARBAYEV UNIVERSITY
SCHOOL OF SCIENCE AND TECHNOLOGY



ROBT305 - Embedded Systems

Lecture 8 – Preemptive Process Scheduling. Round-Robin Scheduling

15 September, 2015

Course Logistics

Reading Assignment:
Chapter 5 of the Operating Systems Concept textbook **Chapter 3** of the Real-Time Systems Design and Analysis textbook
Homework Assignment #2 is out in Moodle and due to end of 26 September (Sunday)
Quiz #2 is on 17 September – Pthreads, Mutexes

Context Switching

- Context switching is the process of saving and restoring sufficient information for a task so that it can be resumed after being interrupted.
- It is performed by a dispatcher module
- The context is ordinarily saved to a stack data structure.
- Context switching time is a major contributor to response time and therefore must be minimized.
- The rule for saving context is simple: save the minimum amount of information necessary to safely restore any process after it has been interrupted.

Context Switching

- Context usually includes:
  - contents of general registers
  - contents of the program counter
  - contents of coprocessor registers (if present)
  - memory page register
  - images of memory-mapped I/O locations (mirror images)
- Interrupts are disabled during context-switching.

Context Switching

- The stack model for context switching is used mostly in embedded systems where the number of real-time or interrupt-driven tasks is fixed.
- In the stack model, each interrupt handler is associated with a hardware interrupt and is invoked by the CPU, which vectors to the instruction stored at the appropriate interrupt-handler location.
- The context is then saved to a specially designated memory area that can be static, in the case of a single-interrupt system, or a stack, in the case of a multiple-interrupt system.

Types of Scheduling

- CPU scheduling decisions may take place when a process:
  1. Switches from running to waiting state (for example, termination of child process, I/O request)
  2. Switches from running to ready state (interrupt occurs)
  3. Switches from waiting to ready (completion of I/O)
  4. Terminates
- Scheduling under 1 and 4 is **nonpreemptive or cooperative**
- All other scheduling is **preemptive**

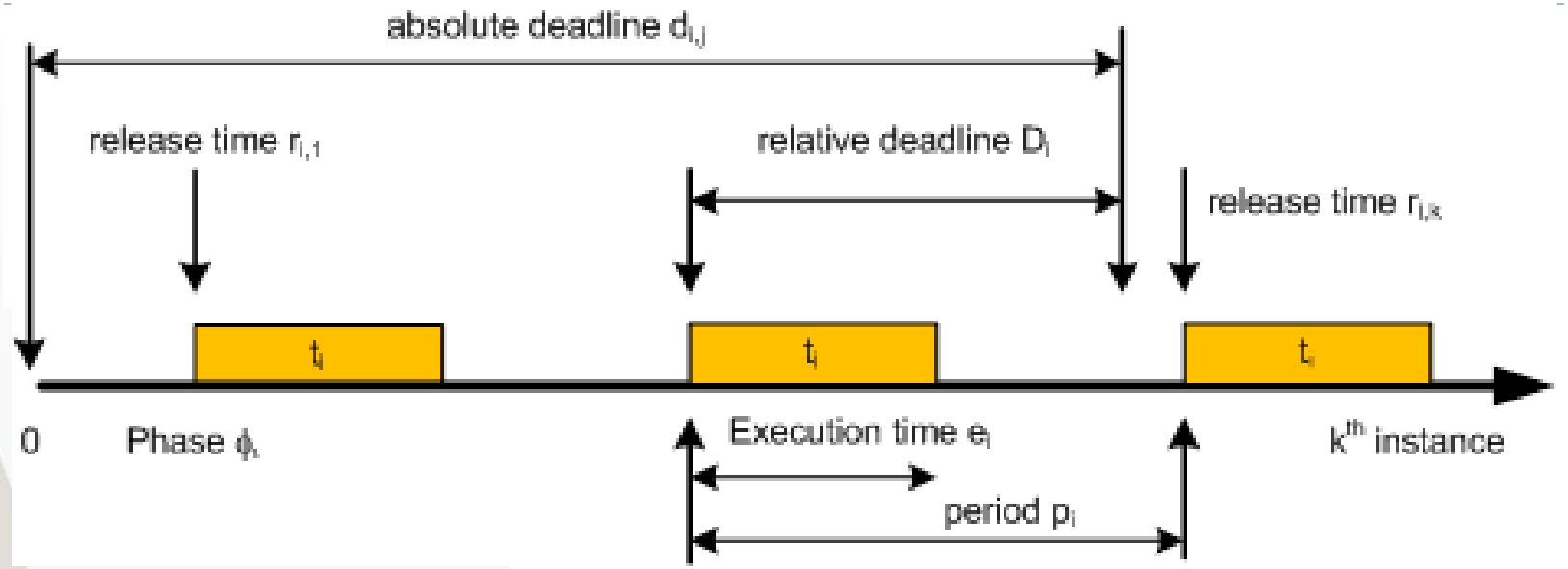
Preemptive Priority Systems

- Preemptive priority systems use preemption (prioritized interrupts). The priorities assigned to each interrupt are based on the urgency of the task associated with the interrupt.
- Prioritized interrupts can be either fixed priority or dynamic priority.
  - Fixed-priority systems are less flexible since the task priorities cannot be changed.– rate monotonic scheduling
  - Dynamic-priority systems can allow the priority of tasks to be adjusted at run-time to meet changing process demands – earliest deadline first scheduling
- Preemptive priority schemes can suffer from resource hogging by higher-priority tasks leading to a lack of available resources for lower-priority tasks. This is called starvation.

Task Characteristics of a Real Workload

- **Precedence constraints**: specifies if any task(s) needs to precede other tasks.
- **Release or arrival time** $r_{i,k}$: the release time of the k^{th} instance of task τ_i.
- **Phase**: Φ_i the release time of the first instant of task τ_i.
- **Response time**: Time span between the task activation and its completion.
- **Absolute deadline** d_i: is the instant of time by which the task τ_i must complete.
- **Relative deadline** D_i: is the maximum allowable response time of the task.
- **Laxity type**: Notion of urgency or leeway in a task's execution.
- **Period** p_i: is the minimum length of intervals between the release times of consecutive tasks.
- **Execution time** e_i: is the (maximum) amount of time required to complete the execution of a task *i* when it executes alone and has all the resources it requires.

Task Characteristics of a Real Workload



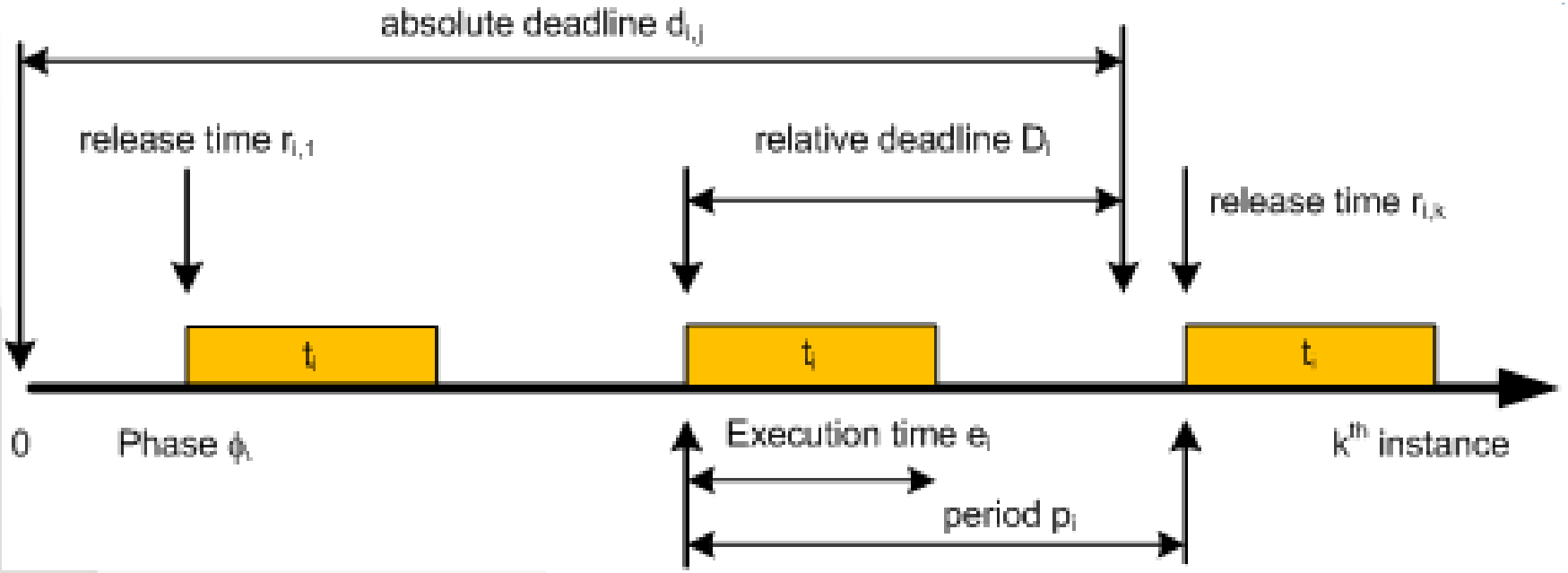
- Mathematically, some of the parameters listed above are related as follows:

$$\phi_i = r_{i,1} \text{ and } r_{i,k} = \phi_i + (k-1) * p_i$$

- the absolute deadline $d_{i,j}$ of the j^{th} instance of task τ_i is found as follows

$$d_{i,j} = \phi_i + (j-1) * p_i + D_i$$

Task Characteristics of a Real Workload



- If the relative deadline of a periodic task D_i is equal to its period p_i, then

$$d_{i,k} = r_{i,k} + p_i \Rightarrow \phi_i + k * p_i$$

where k is some positive integer greater than or equal to one, corresponding to the k^{th} instance of that task.

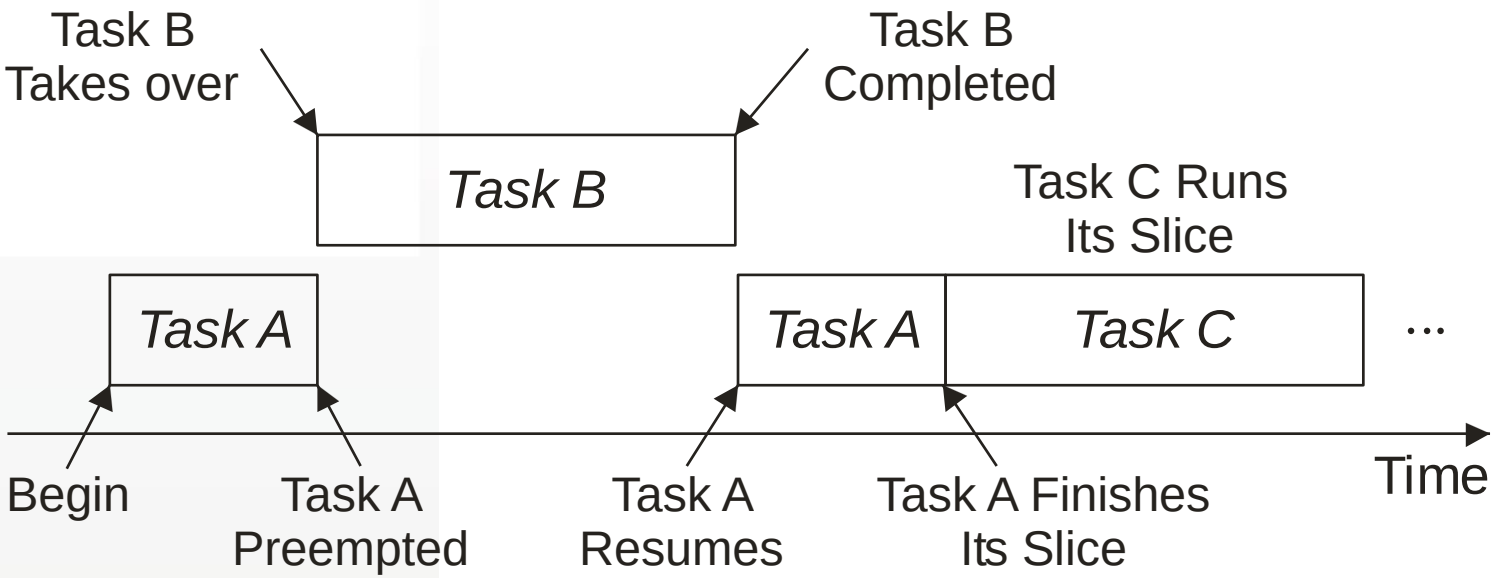
Task Characteristics of a Real Workload

- A simple task model has the following simplifying assumptions:
  - All tasks in the task set are strictly periodic.
  - The relative deadline of a task is equal to its period/frame.
  - All tasks are independent; there are no precedence constraints.
  - No task has any non-preemptible section, and the cost of preemption is negligible.
  - Only processing requirements are significant; memory, and I/O requirements are negligible.
- For real-time systems, it is of utmost importance that the scheduling algorithm produces a predictable schedule
- Many real-time operating systems use round-robin scheduling policy because it is simple and predictable.

Round-Robin Scheduling

- In a round-robin system several processes are executed sequentially to completion, often in conjunction with a cyclic executive.
- In round-robin systems with time slicing, each process gets a small unit of CPU time, (**time quantum** or a **time slice)** in which to execute. It is usually 10-100 milliseconds.
- After this time has elapsed, the process is preempted and added to the end of the ready queue. The context of the next executable task in the list is restored, and it resumes execution.
- Timer interrupts every quantum to schedule next process
- Round-robin systems can be combined with pre-emptive priority systems, yielding a kind of mixed system.

Round-Robin Scheduling

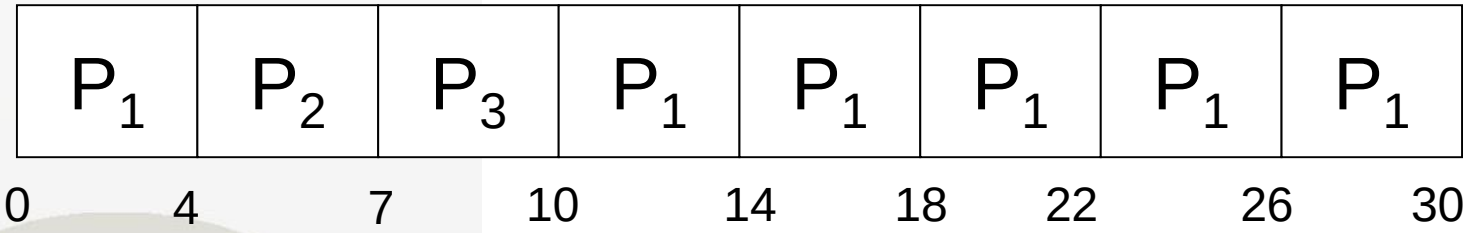


Mixed scheduling of three tasks. Here process A and C are of the same priority, whereas B is of higher priority. Process A is executing for some time when it is preempted by task B, which executes until completion (because B has higher priority than all other tasks). When process A resumes, it continues until its time slice expires, at which time context is switched to process C, which begins executing.

Example of Round Robin Scheduling with Time Quantum = 4

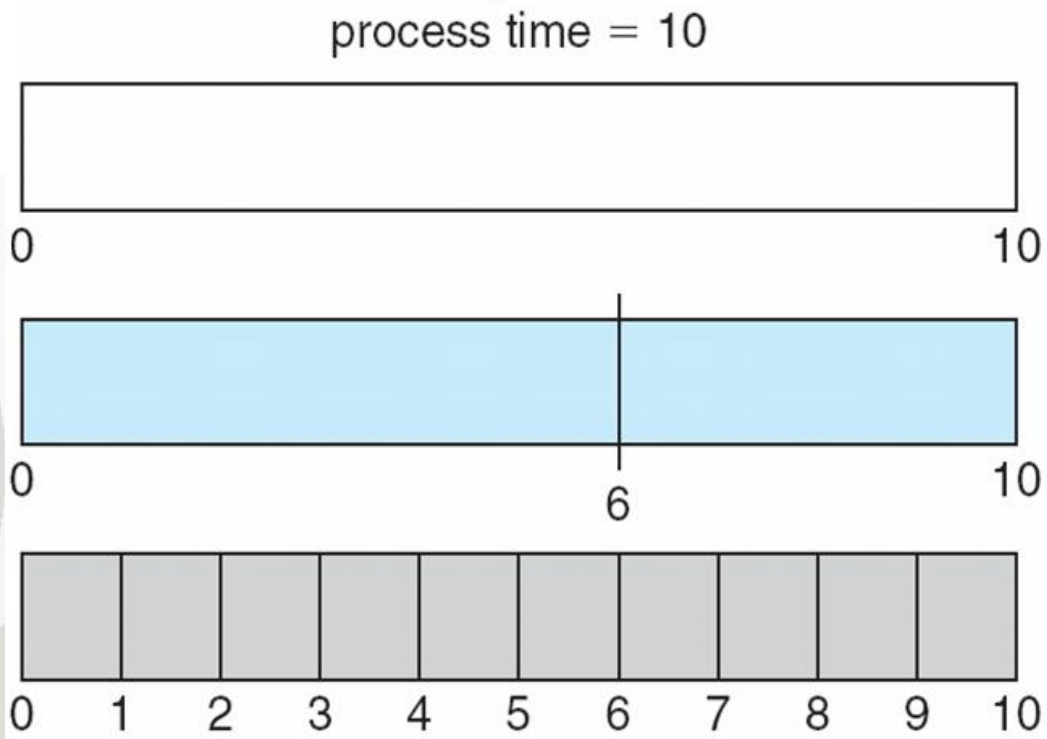
Process	Execution Time
P_1	24
P_2	3
P_3	3

- The Gantt chart is:



- q should be large compared to context switch time
- q usually 10ms to 100ms, context switch < 10 μsec

Time Quantum and Context Switch Time



quantum	context switches
12	0
6	1
1	9

Practice Round Robin Scheduling

A set of independent tasks A, B, C and D need to execute on a processor.

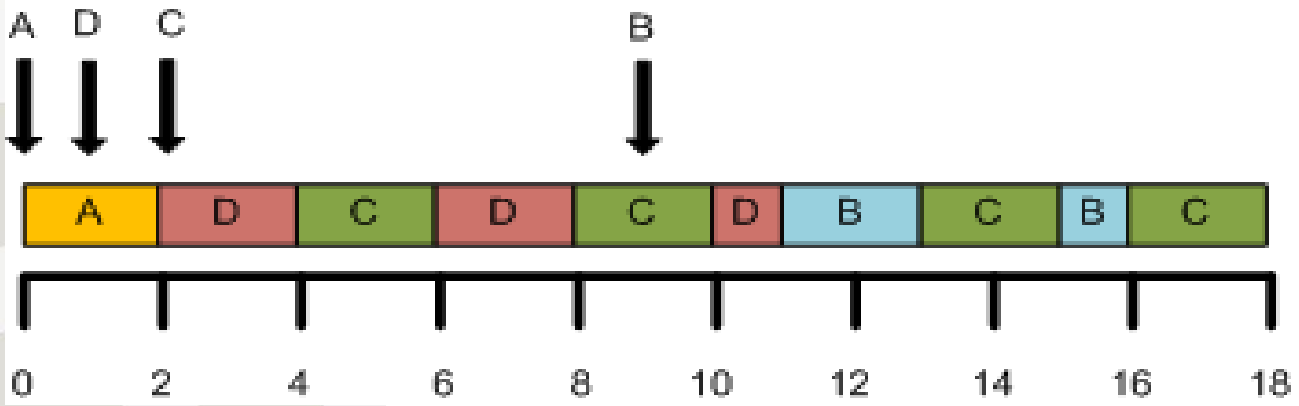
Task	Arrival Time	Execution time
A	0	2
B	9	3
C	2	8
D	1	5

Determine the schedule generated by the Round Robin (RR) scheduling algorithm. Assume a time quantum, q = 2 time units, and processor context switching time of 0 time units.

Practice Round Robin Scheduling

A set of independent tasks A, B, C and D need to execute on a processor.

Task	Arrival Time	Execution time
A	0	2
B	9	3
C	2	8
D	1	5



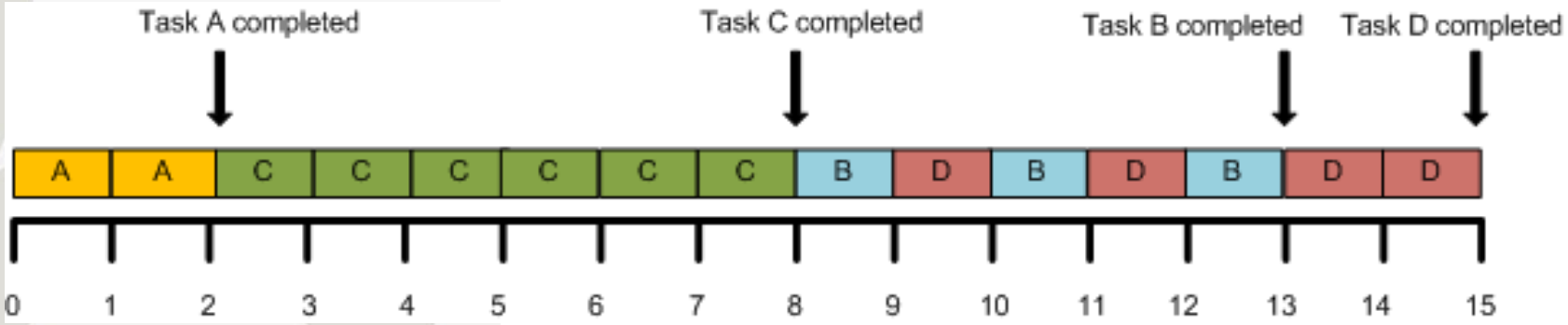
Round Robin Priority Scheduling

Task	Execution Time	Priority
A	2	1
B	3	3
C	6	2
D	4	3

Draw a Gantt chart for time quantum = 1 msec.
Low priority number defines higher priority

Round Robin Priority Scheduling

Task	Execution Time	Priority
A	2	1
B	3	3
C	6	2
D	4	3



Analysis of Round-Robin Systems

- Assume that a round - robin system has ***n*** tasks in the ready queue, no new ones arrive after the scheduling starts, and none terminates prematurely.
- Further assume that all the tasks have the maximum end- to-end execution time of ***c*** time units.
- With constant time quantum ***q,*** worst case time **T** from readiness to completion for any task (also known as turnaround time), is the waiting time plus undisturbed time to complete, **c**, or

$$T = (n-1)\left\lceil \frac{c}{q} \right\rceil q + c$$

- **Example: Turnaround Time Calculation without Context Switching Overhead**

Suppose there are 5 processes with a maximum execution time of 500 ms. The time quantum is 100ms. Then total completion time is

$$T = (5-1)\left\lceil \frac{500}{100} \right\rceil 100 + 500 = 2500ms$$

Analysis of Round-Robin Systems

- Now assume that there is a context switching overhead **o.** Then

$$T = \left[(n-1)q + n \cdot o\right]\left\lceil \frac{c}{q} \right\rceil + c$$

- **Example: Turnaround Time Calculation with Context Switching Overhead**
- Consider previous case with context switch overhead of 1 ms. Then time to complete task set is

$$T = \left[(5-1)\cdot 100 + 5 \cdot 1\right]\left\lceil \frac{500}{100} \right\rceil + 500 = 2525$$

- Context switch costs 1 ms each time for the 25 times it occurs

Any Questions?