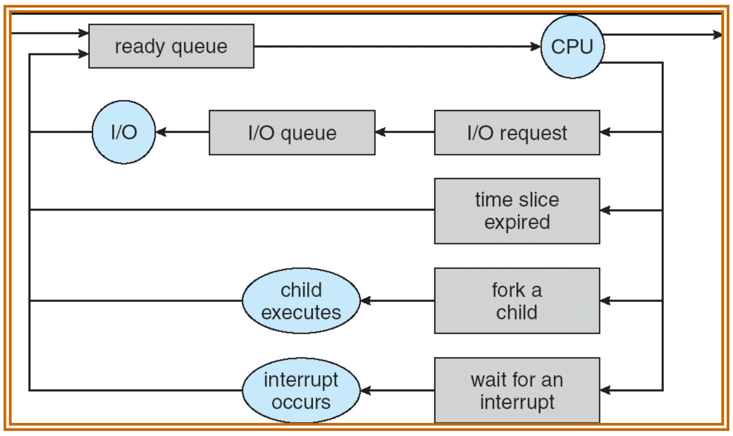
Round Robin Scheduling

CPU scheduling algorithm: Determining which task (thread) to be scheduled first and given access for execution.

This algorithm is implemented based on the assumptions –

- Each task is independent from other tasks.
- One thread is allocated per task.

Scheduling is all about selecting a task from the Ready queue and allocating the CPU to it.



- Active threads work their way from Ready queue to Running queue to various Waiting queues.
- The Ready queue has more priority.
- Each scheduling decision is about which task to give the CPU for use by its next CPU burst.

When multilevel queues are used, tasks are assigned to a priority classes, where –

- Each class has its own ready queue
- Scheduler picks the highest priority queue (class) which has at least one ready task.
- Selection of a task within the class could have its own scheme.

The following terminology is used in this context –

- Burst time: Actual time required to complete a task.
- Waiting time: Time a task waits in the ready queue.
- Service time: Time a task takes to run. This is equal to Burst time.
- Response time: Time from request to response.

Response time = Waiting time + Service time

- Throughput: Number of tasks completed per unit time.

The goal is to –

- Minimize Response time.
- Maximize Throughput.
- Fairness (Each task should get a fair share of the CPU).

<u>Round Robin Scheme:</u>

- Each task gets a small unit of CPU time (time quantum).
- After quantum expires, the process is preempted and added to the end of the ready queue.
- If there are *n* tasks in ready queue and time quantum is *q*, then,
    - Each task gets 1/*n* of the CPU time, with at most *q* time units in one iteration.
    - No task waits more than (*n* – 1)**q* time units in one iteration.
- In practice, need to balance short-task performance and long-task throughput –
    - Typical time-slice is 10-100 milliseconds.
    - Typical context-switching overhead is 0.1-1 milliseconds.
    - *q* must be larger with respect to context-switching cost, but not too large.

Advantages –

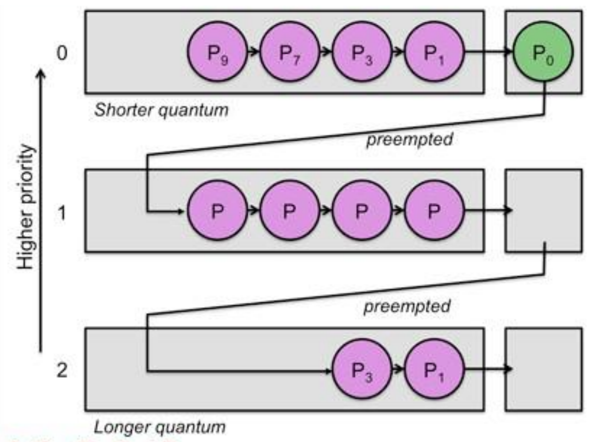
- It is fair (each task gets a fair chance to run on the CPU).
- Shorter tasks gets completed earlier than longer tasks.
- No starvation.

Disadvantages –

- Increased context-switching. Longer tasks require more time to complete.
- High Average Response time when all the tasks have same burst time.
- If *q* is very large, results in overheading.

Multilevel feedback queues –

- Multiple queues are maintained for tasks, even if they have common characteristics.
- Task dynamically moves between priority classes based on its CPU or I/O activity.
- CPU bound tasks are likely to complete its entire time-slice.
- I/O bound tasks may not complete the entire time-slice.



Multilevel Feedback Queue

- All tasks start in the highest priority class.
- If it finishes its time-slice:
  Move to next lower priority.
- If it doesn't finish its time-slice:
  Keep it in the same priority class.

Explanation:

Assume there are n tasks (P_1, P_2, P_3, …, P_n). The ith task P_i has Burst time b_i . Let q be the time quantum. Initially, all the tasks are queued in the Waiting queue. At time $t_0 = 0$, task P_1 is queued in Ready queue. If its Burst time is less than the time quantum, the task is moved to completion; time $t_1 = b_1$. If its Burst time is more than the time quantum, it is executed up to time quantum units and the Burst time is updated to its remaining time. Then, the task is sent back to waiting queue; time $t_1 = q$.

Now, task P_2 is queued in Ready queue. If its Burst time is less than the time quantum, the task is moved to completion; time $t = t_1 + b_2$. If its Burst time is more than the time quantum, it is executed up to time quantum units and the Burst time is updated to its remaining time. Then, the task is sent back to waiting queue; time $t = t_1 + q$.

Now, task P_3 is queued in Ready queue.

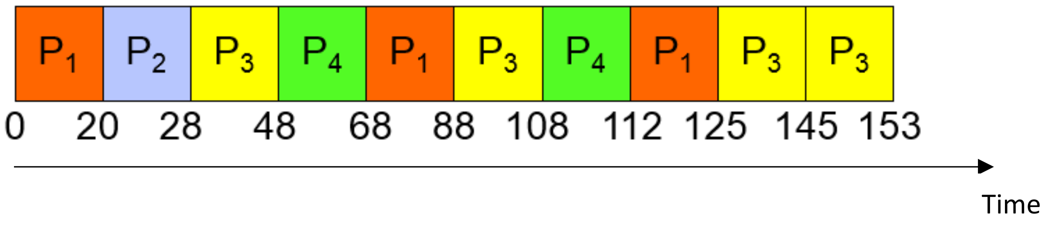
Similarly, the process is continued until all the tasks are executed completely.

Implementing Round Robin Scheme using an example:

Number of tasks = n = 4.   Considering time quantum = q = 20.

Task	Burst time
P_1	**53**
P_2	**8**
P_3	**68**
P_4	**24**

– The Gantt chart is:



— Total Waiting time for P_1 = (68-20)+(112-88) = 72

P_2 = (20-0) = 20

P_3 = (28-0)+(88-48)+(125-108) = 85

P_4 = (48-0)+(108-68) = 88

Average Waiting time = (72+20+85+88)/4 = 66.25

— Response time for P_1 = 125

P_2 = 28

P_3 = 153

P_4 = 112

Average Response time = (125+28+153+112)/4 = 104.25