

# RR

## RR (Round-Robin; De roda; Carrusel)

Este algoritmo respecta a quenda de chegada, mais soamente deixa un tanto (q) de tempo o uso da CPU para cada proceso.

Imos ver un exemplo para explicar como traballa o algoritmo **RR**:

- Supoñemos a situación seguinte:

◇ **Tempo de chegada:** P1-->0, P2-->5, P3-->4, P4-->2

◇ **Cola:** P1, P4, P3, P2

◇ **Duración Proceso:** P1-->4 ciclos de CPU, P2-->7 ciclos de CPU, P3-->4 ciclos de CPU, P4-->1 ciclo de CPU.

◇ **Quanto:** q=2

sendo,

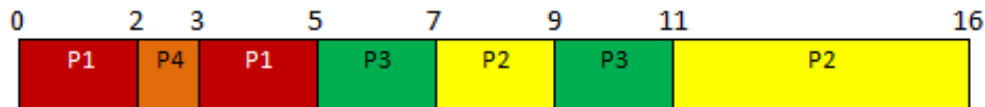
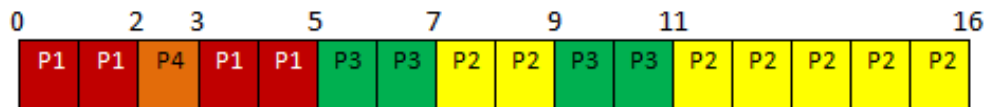
$te_{|P_i}$  O tempo de espera do Proceso  $P_i$

$tr_{|P_i}$  O tempo de retorno do Proceso  $P_i$

Imos calcular o tempo de espera(te), o tempo medio de espera, o tempo de espera máximo e o tempo de retorno(tr), así como o Diagrama de Gantt correspondente,

|                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| P1               | P1 | P1 | E  | P1 | P1 |    |    |    |    |    |    |    |    |    |    |    |
| P2               |    |    |    |    |    | E  | E  | P2 | P2 | E  | E  | P2 | P2 | P2 | P2 |    |
| P3               |    |    |    |    | E  | P3 | P3 | E  | E  | P3 | P3 |    |    |    |    |    |
| P4               |    |    | P4 |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Ciclos CPU       | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| Tempo de chegada | 0  | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | 10 | 11 | 12 | 13 | 14 | 15 |
|                  | ↑  |    | ↑  |    | ↑  | ↑  |    |    |    |    |    |    |    |    |    |    |
|                  | P1 |    | P4 |    | P3 | P2 |    |    |    |    |    |    |    |    |    |    |

|               |  |                |
|---------------|--|----------------|
| $te _{p_1}=1$ |  | $tr _{p_1}=5$  |
| $te _{p_2}=4$ | $\bar{te} = [(te _{p_1} + te _{p_2} + te _{p_3} + te _{p_4})/4] = [(1+4+3+0)/4]=2$ | $tr _{p_2}=11$ |
| $te _{p_3}=3$ | $te _{max}=2$ (Procesos P2 e P3)   | $tr _{p_3}=7$  |
| $te _{p_4}=0$ |  | $tr _{p_4}=1$  |



Como podemos ver na imaxe o primeiro en entrar na CPU é o proceso P1 pois na orde de chegada é o primeiro da cola de procesos. O algoritmo RR determina que ao entrar un proceso está ocupará a CPU durante un quanto de tempo  $q$ , no exemplo  $q=2$ , deixando liberada a CPU para o seguinte proceso que a acapará segundo o algoritmo RR, así:

1. **Ciclo 1 da CPU-Tempo de Chegada 0:** Entra o proceso P1 na CPU e acapara 2 ciclos da mesma, quedando para o remate do mesmo outros 2 ciclos -pois a duración deste proceso son 4 ciclos de CPU-.
2. **Ciclo de CPU 3-Tempo de Chegada 2:** A continuación entra o proceso P4 que acapará 2 ciclos de CPU, xa que  $q=2$ , mais soamente acapara 1 ciclo de CPU -pois a duración deste proceso é 1 ciclo de CPU-.
3. **Ciclo de CPU 4-Tempo de Chegada 3:** Nesta situación aínda non temos ningún outro proceso en cola agás o P1 co cal entrará o P1 outros dous ciclos de CPU, xa que  $q=2$ , rematando así o proceso -pois a duración deste proceso son 4 ciclos de CPU-.
4. **Ciclo de CPU 6-Tempo de Chegada 5:** Agora entra na CPU o proceso P3, xa que os procesos P3 e P2 están en cola mais o proceso P3 leva máis tempo na mesma -cando debe entrar un proceso na CPU e varios dos que están en cola teñen as mesmas posibilidades ou probabilidades resolvemos este conflito mediante o algoritmo FCFS-. Así entra o proceso P3 durante 2 ciclos de CPU -quedándolle outros 2 ciclos para o seu remate-.
5. **Ciclo de CPU 8-Tempo de Chegada 7:** Logo entra o proceso P2 durante 2 ciclos de CPU -quedándolle 5 ciclos de duración-.
6. **Ciclo de CPU 10-Tempo de Chegada 9:** Logo entra de novo P3 2 ciclos de CPU e remata a súa execución.
7. **Ciclo de CPU 12-Tempo de Chegada 11:** Entra P2 2 ciclos de CPU -quedándolle 3 ciclos de duración-.
8. **Ciclo de CPU 14-Tempo de Chegada 13:** Como agora non existe ningún outro proceso en cola segue entrando na CPU o proceso P2 ata o remate da súa execución.