

# ***Sistemas de Tempo-Real***

## **Aula 5**

### **Escalonamento usando prioridades fixas**

- **Escalonamento on-line com prioridades fixas**
- **O critério Rate-Monotonic**
  - limite de utilização de CPU
- **Os critérios Deadline-Monotonic e prioridades fixas arbitrárias**
  - análise do tempo de resposta de pior caso

Adaptado dos slides desenvolvidos pelo Prof. Doutor Luís Almeida  
para a disciplina “Sistemas de Tempo-Real”  
Revisto em Out/2013 por Paulo Pedreiras

# ***Aula anterior (4)***

## **Introdução ao escalonamento de tarefas**

- O conceito de **complexidade temporal**
- Definição de **escalonamento** e de **algoritmo de escalonamento**
- Algumas **técnicas preliminares** de escalonamento (EDD, EDF, BB)
- Escalonamento **estático cíclico**



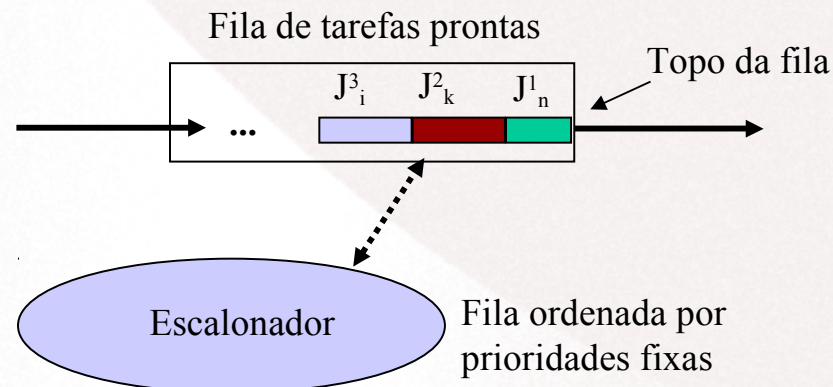
# Escalonamento on-line com prioridades fixas

O escalonamento é **construído** com o sistema **em funcionamento** (*on-line*) e baseia-se num **parâmetro estático** (prioridade).

A fila de tarefa prontas a executar é ordenada por prioridades decrescentes. Executa **primeiro** a que tem **maior prioridade**.

(**com preempção**) Sempre que no topo da fila de tarefas prontas está uma tarefa com maior prioridade do que a que está a executar, esta última é suspensa (regressa à fila) e a nova tarefa é executada.

Complexidade  **$O(n)$**





# Escalonamento on-line com prioridades fixas

## A favor

- Escalável
  - alterações nas tarefas podem ser imediatamente tidas em conta pelo *scheduler*
- Acomoda facilmente tarefas esporádicas
- Comportamento determinístico em sobrecargas
  - apenas as tarefas menos prioritárias são afetadas

## Contra

- Implementação mais complexa
  - requer um kernel de prioridades fixas
- *Overhead* de execução mais elevado (*scheduler* + *dispatcher*)
- Sobrecargas (e.g. devido a erros de projeto ou programação) ao nível de prioridades elevadas podem bloquear o sistema

# Escalonamento on-line com prioridades fixas

## Atribuição de prioridades

- Inversamente proporcional ao período (**RM – Rate Monotonic**)  
**Ótimo em relação aos critérios de prioridades fixas**
- Inversamente proporcional à *deadline* (**DM – Deadline Monotonic**)  
**Ótimo quando  $D \leq T$**
- Proporcional à **importância** da tarefa  
Pode causar uma perda de eficiência – **não ótimo**



# *Escalonamento on-line com prioridades fixas*

## Verificação de escalonabilidade

Como o escalonamento só é construído *on-line* pode ser importante saber *a priori* se um dado conjunto de tarefas cumpre ou não os seus requisitos temporais.

Existem dois tipos principais de testes de escalonabilidade:

- Baseados na **taxa de utilização** do CPU
- Baseados no **tempo de resposta**

# Escalonamento RM

## Testes para RM baseados na taxa de utilização

(com preempção,  $n$  tarefas independentes e  $D=T$ )

- **Menor majorante** de Liu&Layland (1973)

$$U(n) = \sum_{i=1}^n \frac{C_i}{T_i} \leq n(2^{\frac{1}{n}} - 1) \Rightarrow \text{Uma ativação por período garantida}$$

- **Majorante hiperbólico** de Bini&Buttazzo&Buttazzo (2001)

$$\prod_{i=1}^n \left( \frac{C_i}{T_i} + 1 \right) \leq 2 \Rightarrow \text{Uma ativação por período garantida}$$



# Escalonamento RM

## Significado do teste de utilização de Liu&Layland

$U(n) > 1 \Rightarrow$  conjunto não escalonável (*overload*) – condição necessária

$U(n) \leq \text{Majorante} \Rightarrow$  conjunto escalonável – condição suficiente

$1 \geq U(n) > \text{Majorante} \Rightarrow$  situação indeterminada

Liu&Layland

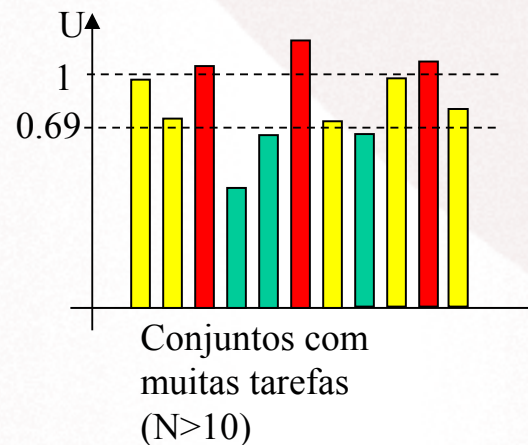
$$U(1) \leq 1$$

$$U(2) \leq 0.83$$

$$U(3) \leq 0.78$$

...

$$U(\infty) \leq \ln(2) \approx 0.69$$

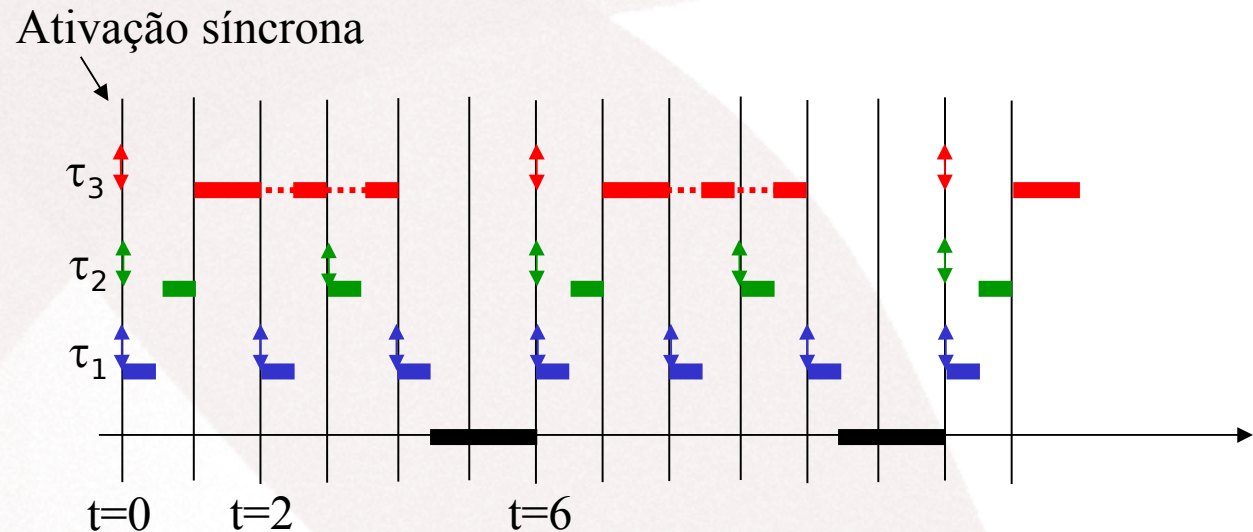




## Escalonamento RM – exemplo 1

## Tabela de propriedades das tarefas

$\tau_i$	$T_i$	$C_i$
1	2	0.5
2	3	0.5
3	6	2



$$U = 0.5/2 + 0.5/3 + 2/6$$

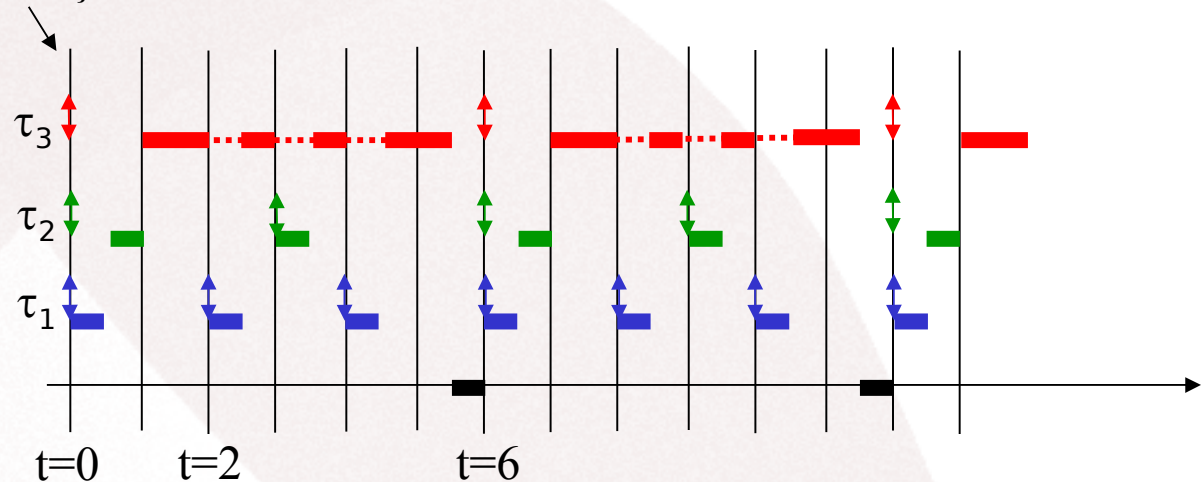
**$U = 0.75 < 0.78 \Rightarrow 1$  ativação por período garantida**

# Escalonamento RM – exemplo 2

Tabela de propriedades das tarefas

$\tau_i$	$T_i$	$C_i$
1	2	0.5
2	3	0.5
3	6	3

Ativação síncrona



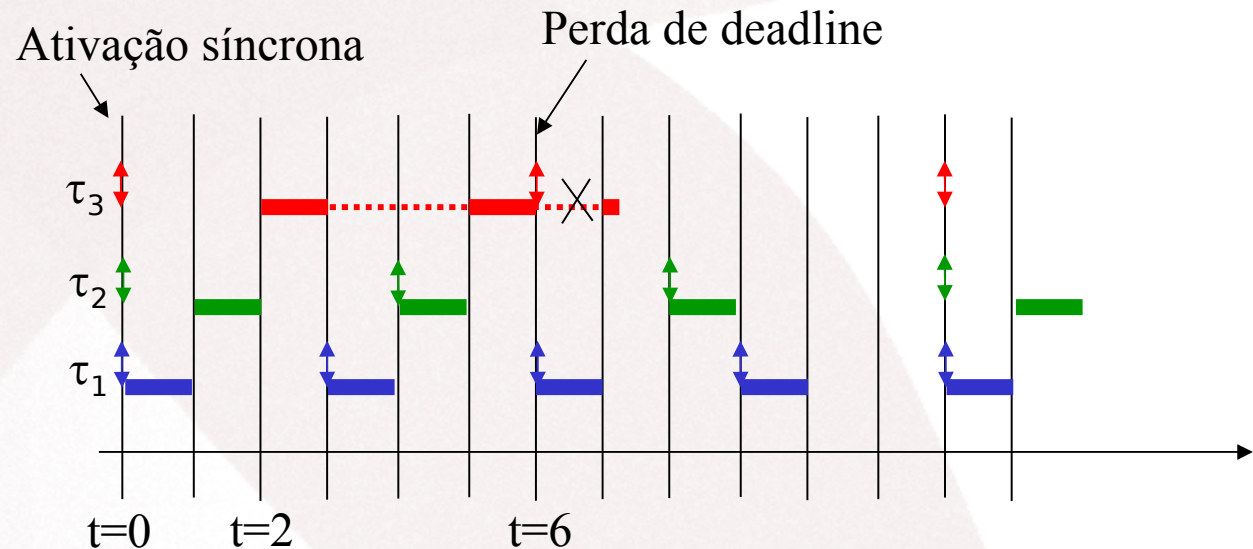
$U = 0.5/2 + 0.5/3 + 3/6 = 0.92 > 0.78 \Rightarrow 1$  ativação por período NÃO garantida mas praticável



# Escalonamento RM – exemplo 3

Tabela de propriedades das tarefas

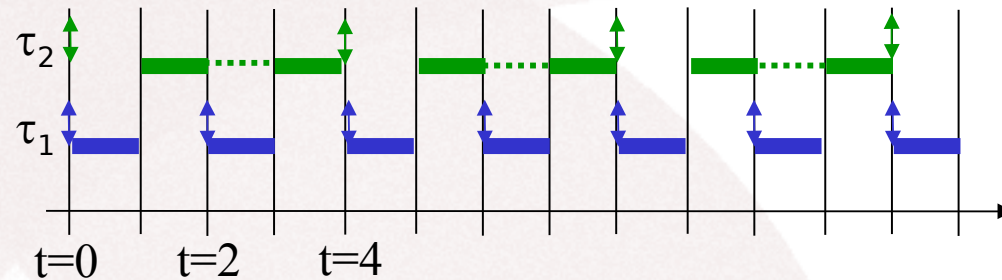
$\tau_i$	$T_i$	$C_i$
1	3	1
2	4	1
3	6	2.1



$U = 1/3 + 1/4 + 2.1/6 = 0.93 > 0.78 \Rightarrow 1$  ativação por período NÃO garantida, com perda de *deadline* em  $\tau_3$

# Escalonamento RM – caso particular

$$U = 1/2 + 2/4 = 1$$



Períodos harmónicos

**Escalonável sse  $U(n) \leq 1$**



# Escalonamento DM

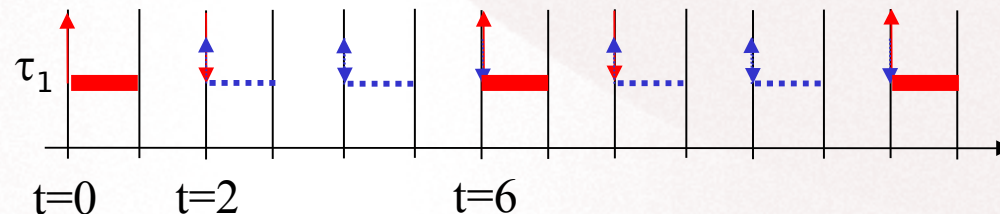
## Testes de escalonabilidade para DM

Quando uma tarefa tem um período de ativação relativamente lento mas necessita de ser atendida dentro de um prazo mais curto pode definir-se uma *deadline* menor que o período e escalonar as tarefas pela *deadline*.

Neste caso também se podem usar os testes baseados em utilização mas considerando a *deadline* em vez do período, i.e.

$$U'(n) = \sum_{i=1}^n \frac{C_i}{D_i} \leq n(2^{\frac{1}{n}} - 1) \quad \text{Teste fica muito pessimista!}$$

$\tau_1 (C_1=1, T_1=6, D_1=2)$



# Análise do tempo de resposta

Para **prioridades fixas arbitrárias**, incluindo RM, DM e outras, a análise do **tempo de resposta** permite realizar um teste de escalonabilidade que nas condições consideradas (preempção e ativação síncrona mais independência) é **necessário e suficiente** (i.e. exato!).

Tempo de resposta de pior caso = máximo intervalo de tempo desde a ativação até à terminação.  $Rwc_i = \max_k(f_{i,k} - a_{i,k})$

Teste de escalonabilidade com tempo de resposta

Calcular  $Rwc_i \quad \forall_i$

$\forall_i, Rwc_i \leq D_i \quad \Leftrightarrow \quad \text{conjunto escalonável}$



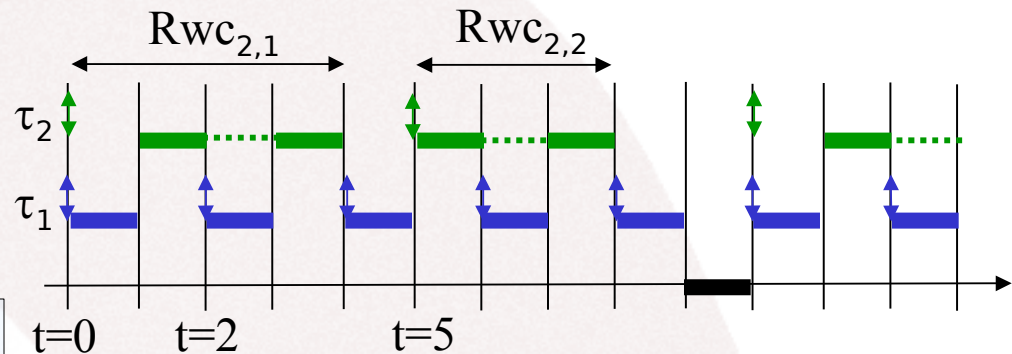
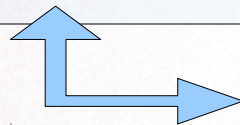
# Análise do tempo de resposta

O tempo de resposta de pior caso ocorre quando uma tarefa é ativada em simultâneo com todas as tarefas de maior prioridade (**instante crítico**)

Cálculo de  $R_{wc_i}$

$$\forall_i R_{wc_i} = I_i + C_i$$

$$I_i = \sum_{k \in hp(i)} \left\lceil \frac{R_{wc_i}}{T_k} \right\rceil * C_k$$



nº de vezes que a tarefa k de maior prioridade é ativada no intervalo  $R_{wc_i}$

$I_i$  – interferência causada pela execução de tarefas de maior prioridade

# Análise do tempo de resposta

A resolução da equação do tempo de resposta faz-se usando um processo iterativo que:

- **Diverge** ( $R_{wc_i} > D_i$ )
- **Converge num número finito de passos**
  - $R_{wc_i}(m+1) = R_{wc_i}(m)$

$$R_{wc_i}(0) = \sum_{k \in hp(i)} C_k + C_i$$

.....

$$R_{wc_i}(m+1) = \sum_{k \in hp(i)} \left\lceil \frac{R_{wc_i}(m)}{T_k} \right\rceil * C_k + C_i$$

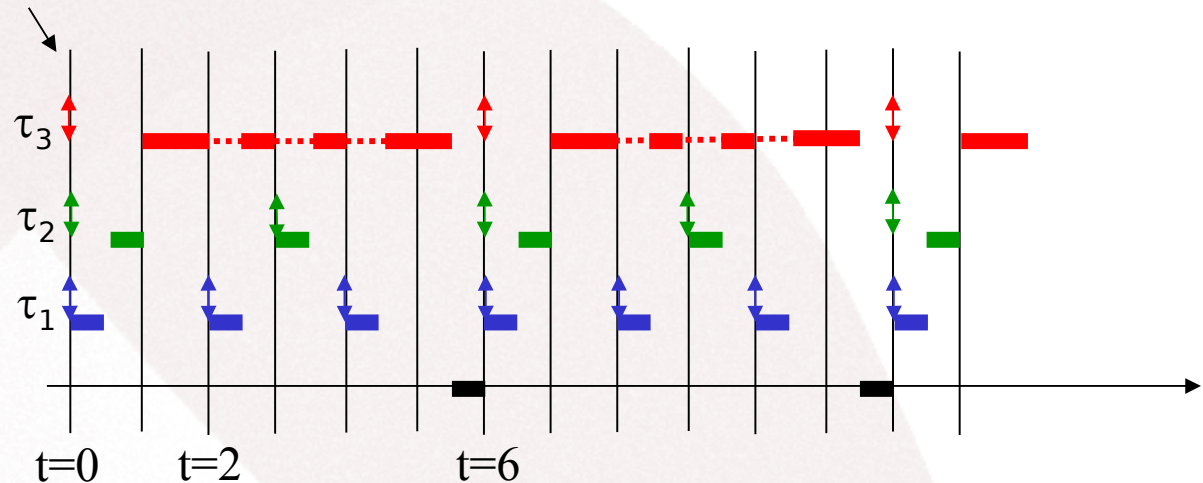


# Análise do tempo de resposta

Tabela de propriedades das tarefas

$\tau_i$	$T_i$	$C_i$
1	2	0.5
2	3	0.5
3	6	3

Instante crítico



$Rwc_1$ : ?

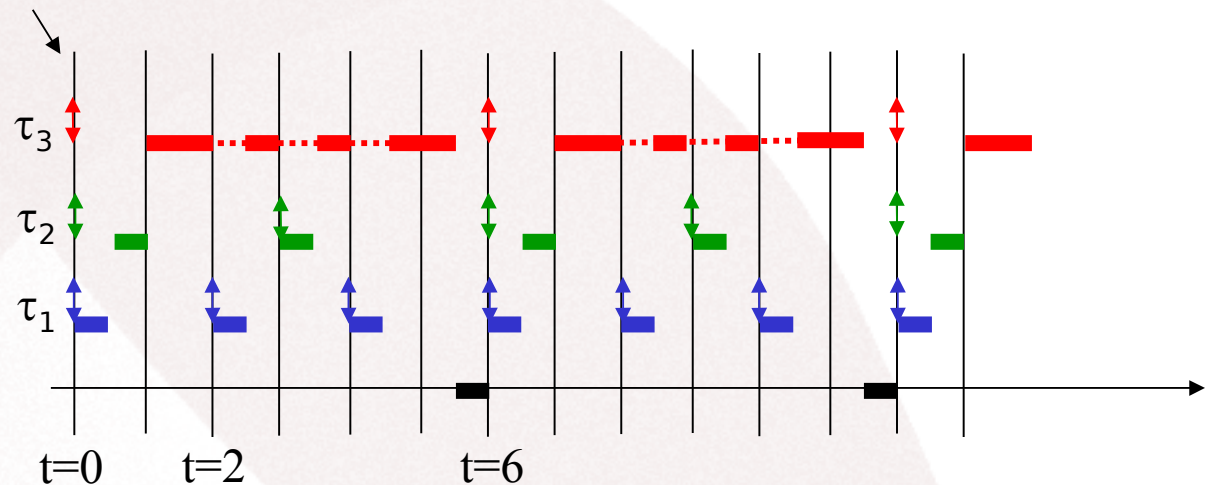
$Rwc_2$ : ?

# Análise do tempo de resposta

Tabela de propriedades das tarefas

$\tau_i$	$T_i$	$C_i$
1	2	0.5
2	3	0.5
3	6	3

Instante crítico



$Rwc_1:$   $Rwc_1(0) = C_1 = 0.5$

$Rwc_2:$   $Rwc_2(0) = C_1 + C_2 = 1$

$Rwc_2(1) = \lceil Rwc_2(0)/T_1 \rceil * C_1 + C_2 = 1$

$Rwc_2 = 1$

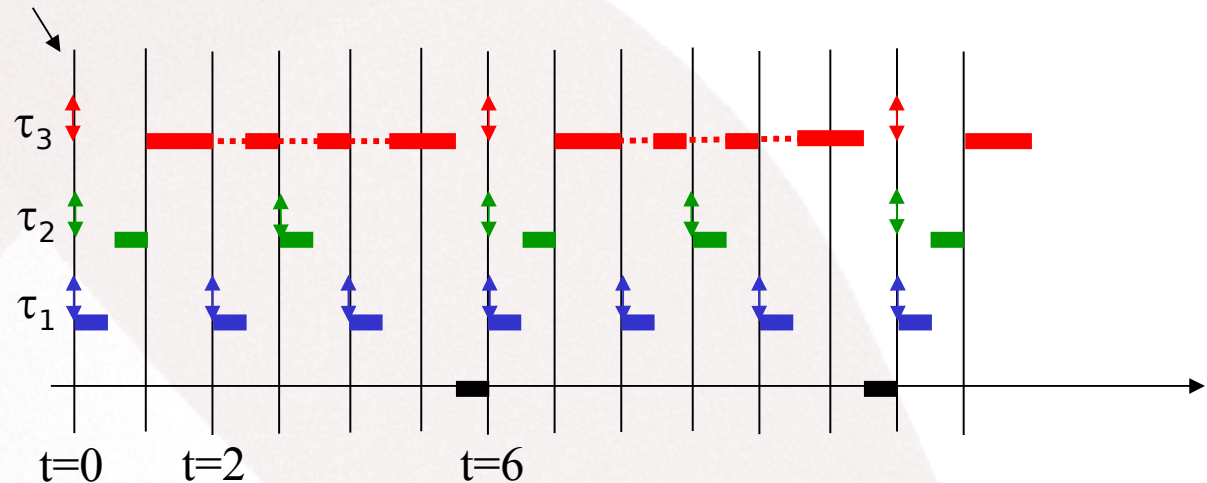


# Análise do tempo de resposta

Tabela de propriedades das tarefas

$\tau_i$	$T_i$	$C_i$
1	2	0.5
2	3	0.5
3	6	3

Instante crítico



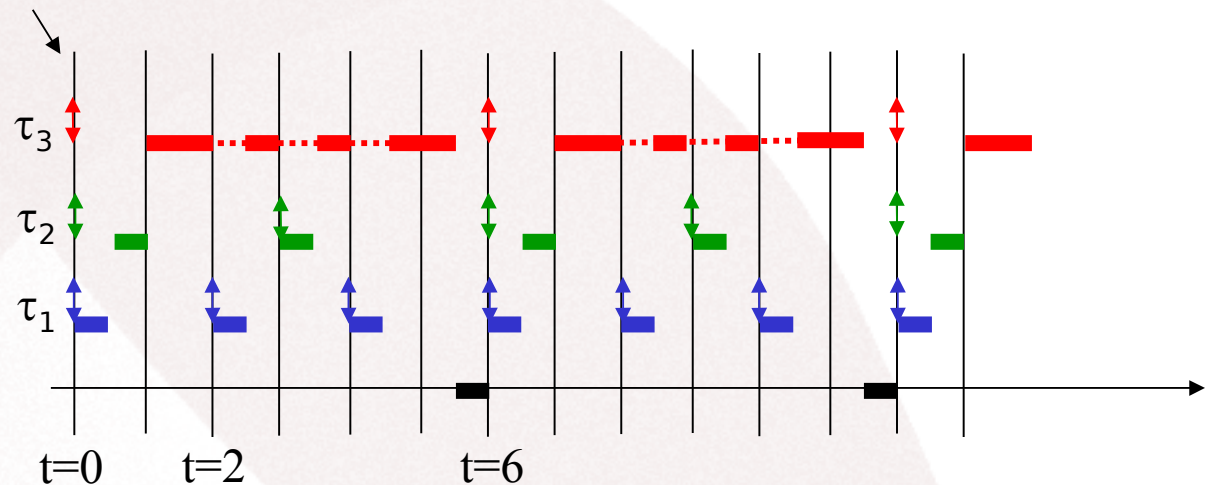
$Rwc_3$ : ?

# Análise do tempo de resposta

Tabela de propriedades das tarefas

$\tau_i$	$T_i$	$C_i$
1	2	0.5
2	3	0.5
3	6	3

Instante crítico



**Rwc<sub>3</sub>:**  $Rwc_3(0) = C_1 + C_2 + C_3 = 4$

$$Rwc_3(1) = \lceil Rwc_3(0)/T_1 \rceil * C_1 + \lceil Rwc_3(0)/T_2 \rceil * C_2 + C_3 = 5$$

$$Rwc_3(2) = \lceil Rwc_3(1)/T_1 \rceil * C_1 + \lceil Rwc_3(1)/T_2 \rceil * C_2 + C_3 = 5.5$$

$$Rwc_3(3) = \lceil Rwc_3(2)/T_1 \rceil * C_1 + \lceil Rwc_3(2)/T_2 \rceil * C_2 + C_3 = 5.5$$

$$Rwc_3 = 5.5$$



# Restrições às análises apresentadas

Os resultados anteriores têm de ser modificados no caso de:

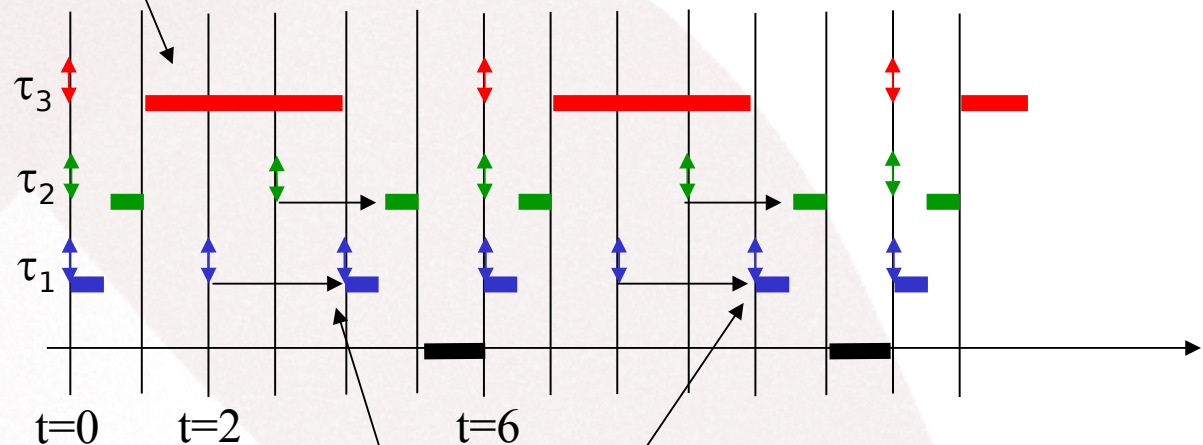
- Não-preempção
- Não independência:
  - Recursos partilhados
  - Precedências
- E é necessário ter em conta o **overhead** do SO ou do Executivo (e.g., nas mudanças de contexto, no atendimento da interrupção de contagem do tempo)
- Bem como o tempo gasto no atendimento de **interrupções assíncronas**

# Impacto da não-preempção

Tabela de propriedades das tarefas

$\tau_i$	$T_i$	$C_i$
1	2	0.5
2	3	0.5
3	6	3

Executa sem preempção



Bloqueio e perda de *deadline*



# Resumo da Aula 5

- Escalonamento **on-line** usando **prioridades fixas**
- O critério de escalonamento **Rate Monotonic** – análise de escalonabilidade baseada na **utilização**
- O critério **Deadline Monotonic** e de **prioridade fixa arbitrária**
- Análise usando **tempo de resposta** de pior caso.