

Sistemas de Tempo-Real

Aula 7

Acesso exclusivo a recursos partilhados

O acesso exclusivo a recursos partilhados
A inversão de prioridades como consequência do bloqueio
Técnicas básicas para acesso exclusivo a recursos partilhados
Herança de prioridades (*Priority Inheritance Protocol – PIP*)
Protocolo de tecto de prioridades (*Priority Ceiling Protocol – PCP*)
Protocolo de pilha de recursos (*Stack Resource Protocol- SRP*)

Adaptado dos slides desenvolvidos pelo Prof. Doutor Luís Almeida
para a disciplina “Sistemas de Tempo-Real”
Revisto em Out/2009 por Paulo Pedreiras

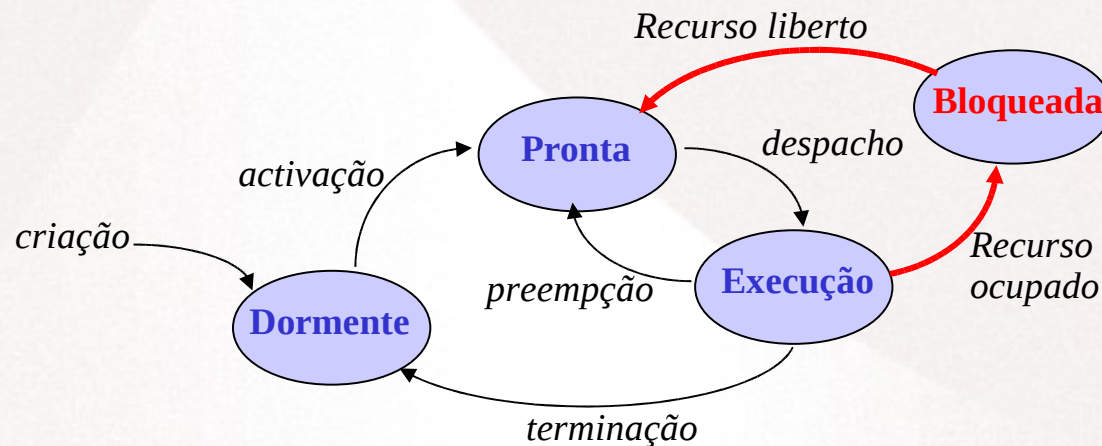
Aula anterior (6)

- Escalonamento *on-line* com prioridades dinâmicas
- O critério *EDF - Earliest Deadline First*:
 - limite de utilização de CPU
- **Optimalidade** de EDF e **comparação com RM**:
 - nível de escalonabilidade, número de preempções, *jitter* de disparo e tempo de resposta
- Outros critérios de prioridades dinâmicas:
 - *LLF (LST) - Least Laxity (Slack) First*
 - *FCFS - First Come First Served*

Recursos partilhados com acesso exclusivo

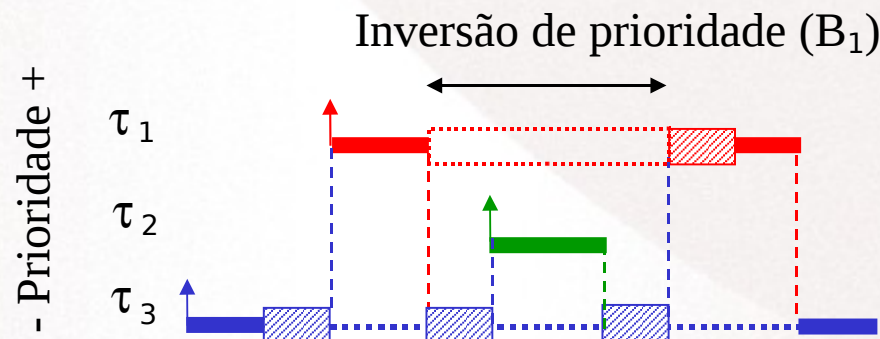
Tarefas: o estado de *Bloqueio*

Quando uma tarefa em execução tenta aceder a um recurso partilhado (e.g. uma porta de comunicação, um buffer em memória) que está ocupado em modo exclusivo por uma tarefa pronta, a primeira diz-se que fica **bloqueada**. Quando o recurso é libertado, a tarefa bloqueada fica novamente pronta para execução.



O fenómeno da inversão de prioridades

- Num sistema de tempo real com preempção e com tarefas **independentes**, quando uma **tarefa executa** é porque detém a **maior prioridade** nesse instante.
- Contudo, quando as tarefas podem aceder em modo exclusivo a recursos partilhados, a situação pode ser diferente. Quando uma tarefa em execução fica bloqueada, a **tarefa bloqueante tem menor prioridade**. Quando a tarefa bloqueante executa (e quaisquer outras de prioridade intermédia) existe uma **inversão de prioridades**.



O fenómeno da inversão de prioridades

- A **inversão de prioridades** é um fenómeno inevitável na presença de recursos partilhados de acesso exclusivo (intrínseco ao bloqueio)
- Contudo, é fundamental **limitar e quantificar o seu impacto** no pior caso, para que se possa raciocinar sobre a **escalonabilidade** do conjunto de tarefas.
- Assim, as técnicas usadas para garantir o acesso exclusivo a cada recurso (**primitivas de sincronização**) deverão permitir limitar a zona de inversão de prioridades e ser analisáveis, i.e. permitir **quantificar o máximo bloqueio** que cada tarefa poderá sofrer.

Técnicas para acesso exclusivo a recursos

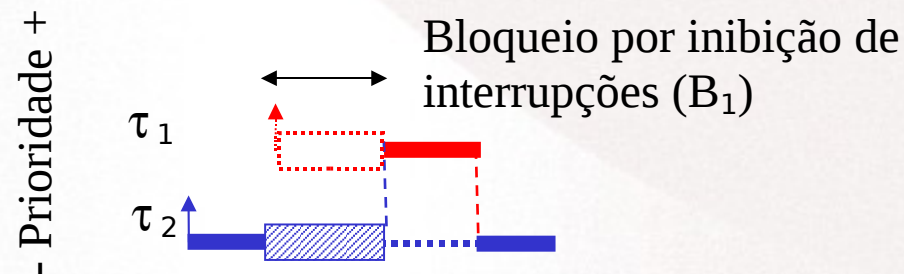
Primitivas de sincronização

- Inibição de **interrupções**
 - *disable / enable* ou *cli / sti*
- Inibição de **preempção**
 - *no_preemp / preempt*
- Utilização de **locks** ou **flags atómicas** (*mutexes* – se bem que este termo por vezes também é usado para designar semáforos – *lock / unlock*)
- Utilização de **semáforos**
 - contador+lista de processos – *P / V* ou *wait / signal*

Técnicas para acesso exclusivo a recursos

Inibição de interrupções

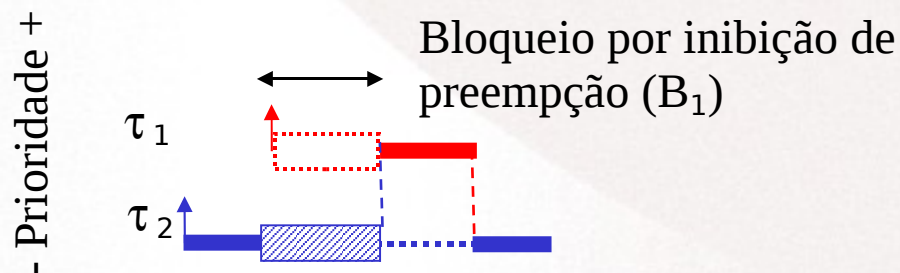
- Todas as restantes actividades do sistema ficam bloqueadas! (não apenas outras tarefas mas também o **atendimento a interrupções** externas e, principalmente, o atendimento do **tick**).
- Esta técnica é muito fácil de implementar mas só deve ser usada com regiões críticas muito curtas (e.g. acesso a uma variável)
- Cada tarefa só pode ser **bloqueada uma vez** e pela duração da **maior região crítica** das tarefas de menor prioridade (ou menor *deadline* relativa para EDF) mesmo quando não usa recursos !!



Técnicas para acesso exclusivo a recursos

Inibição de preempção

- Todas as restantes tarefas do sistema ficam bloqueadas! Todavia o atendimento a interrupções, incluindo o do *tick*, não é afectado
- Fácil de implementar mas pouco eficiente (causa bloqueios desnecessários)
- Cada tarefa só pode ser **bloqueada uma vez** e pela duração da **maior região crítica** das tarefas de menor prioridade (ou menor *deadline* relativa para EDF) mesmo quando não usa recursos !!



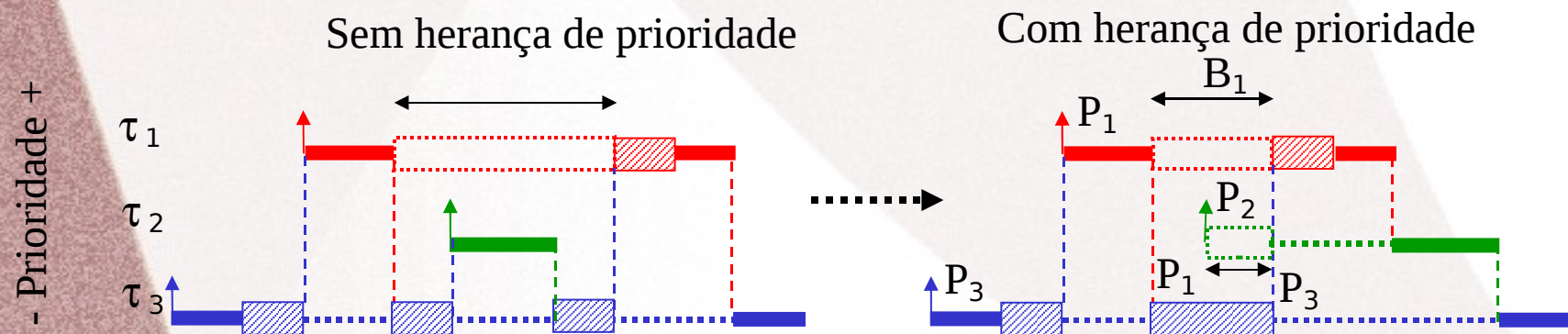
Técnicas para acesso exclusivo a recursos

Utilização de *locks* ou semáforos

- Apenas afectam as tarefas envolvidas na partilha de recursos.
- Implementação mais custosa mas mais eficiente (causa bloqueios apenas às tarefas que acedem a recursos)
- Contudo, a duração dos bloqueios é bastante dependente do **protocolo específico** utilizado para **gerir os semáforos**
- Neste caso é particularmente importante que o referido protocolo permita evitar:
 - **Bloqueios indeterminados**
 - **Bloqueios em cadeia**
 - **Deadlocks**

Protocolo de Herança de Prioridades (PIP – Priority Inheritance Protocol)

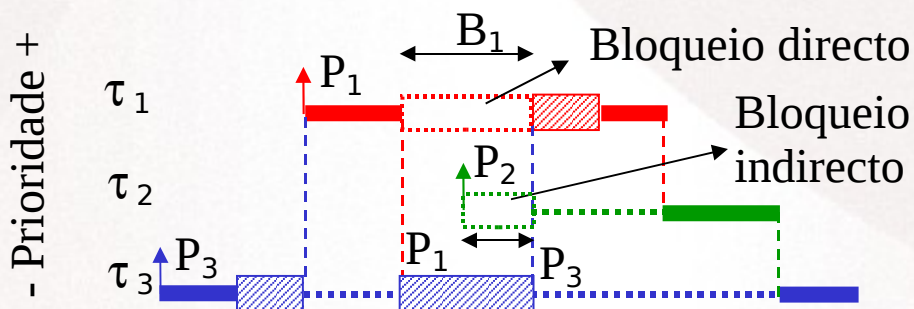
- A tarefa bloqueante (menos prioritária) **herda a prioridade** da tarefa bloqueada (mais prioritária).
- Limita a duração dos bloqueios evitando que tarefas de prioridade intermédia executem enquanto a tarefa de menor prioridade, a bloqueante, não terminar a respectiva região crítica.



Protocolo de Herança de Prioridades (PIP)

Para majorar o **tempo de bloqueio** (B) note-se que uma tarefa pode ser bloqueada por qualquer tarefa de **menor prioridade**:

- com a qual partilhe um recurso (directo) ou
- que possa bloquear uma tarefa de maior prioridade (indirecto).
- Note-se ainda que, na ausência de acessos encadeados:
 - cada tarefa só pode bloquear outra uma vez
 - cada tarefa só pode ficar bloqueada uma vez em cada semáforo



e.g.	S_1	S_2	S_3
τ_1	1	2	0
τ_2	0	9	3
τ_3	8	7	0
τ_4	6	5	4

Protocolo de Herança de Prioridades (PIP)

Análise de escalonabilidade (RM)

$$\forall_{1 \leq i \leq n} \sum_{k=1}^i \frac{C_k}{T_k} + \frac{B_i}{T_i} \leq i(2^{\frac{1}{i}} - 1)$$

$$\sum_{i=1}^n \frac{C_i}{T_i} + \max_{i=1 \dots n} \frac{B_i}{T_i} \leq n(2^{\frac{1}{n}} - 1)$$

$$R_{wc_i} = C_i + B_i + \sum_{k=1}^{i-1} \left\lceil \frac{R_{wc_i}}{T_k} \right\rceil C_k$$

e.g.	C _i	T _i	B _i
τ ₁	5	30	17
τ ₂	15	60	13
τ ₃	20	80	6
τ ₄	20	100	0

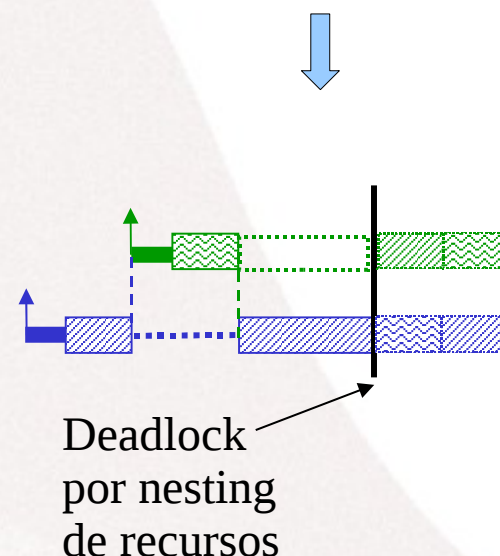
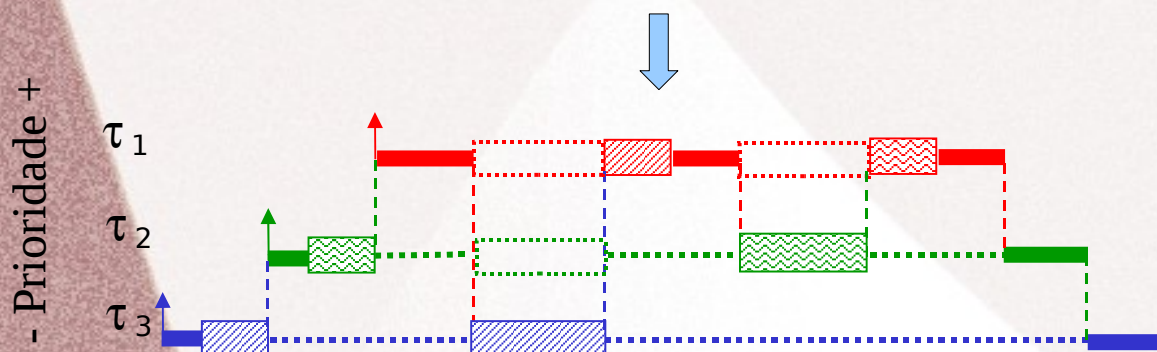


e.g.	S ₁	S ₂	S ₃
τ ₁	1	2	0
τ ₂	0	9	3
τ ₃	8	7	0
τ ₄	6	5	4

Protocolo de Herança de Prioridades (PIP)

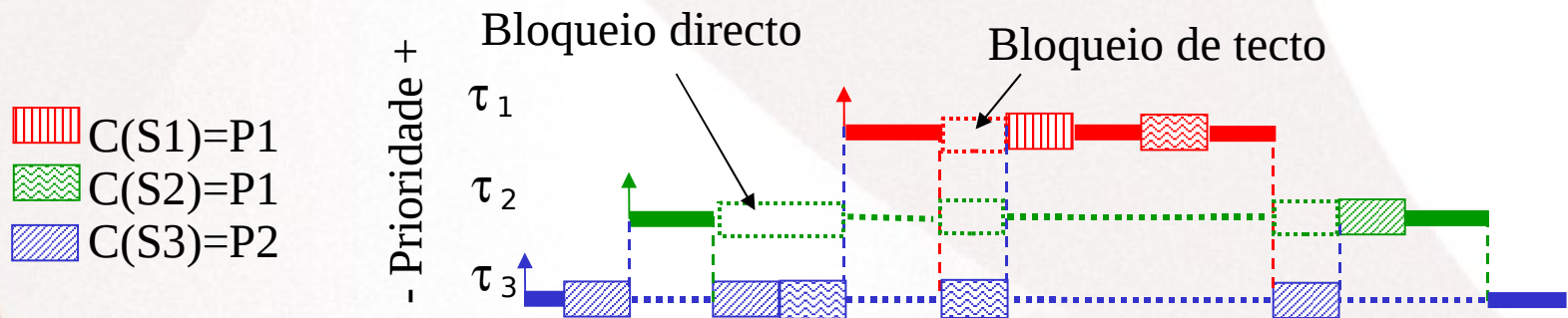
Propriedades:

- 😊 Relativamente fácil de concretizar
 - requer mais um campo no TCB, a prioridade herdada
- 😞 É transparente para o programador (cada tarefa só usa informação local)
- 😞 Sofre de **bloqueio em cadeia** e, principalmente, **não evita deadlocks**



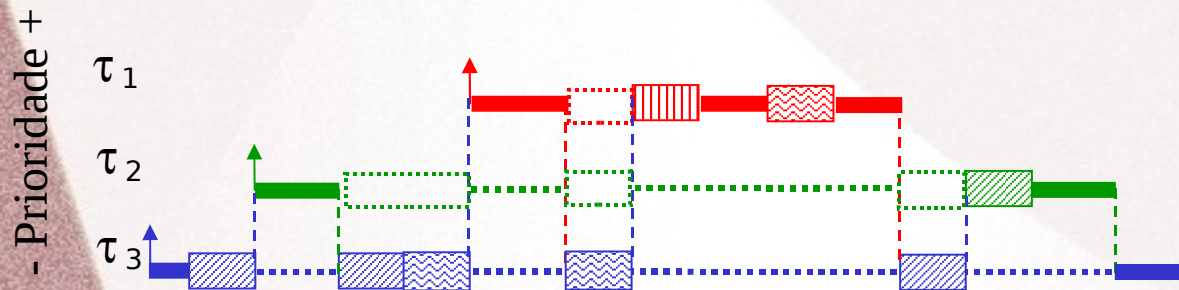
Protocolo de Tecto de Prioridades (PCP – Priority Ceiling Protocol)

- **Extensão ao PIP** mas com uma regra sobre o **acesso aos semáforos livres** (para garantir que todos os semáforos necessários estão livres)
- É definido um **tecto (ceiling) de prioridade** para cada semáforo igual à prioridade mais elevada de entre as tarefas que o usam.
- Uma tarefa só pode **passar um semáforo** se o semáforo **estiver livre** e se a sua **prioridade for maior do que os tectos** de todos os restantes semáforos correntemente fechados.



Protocolo de Tecto de Prioridades (PCP)

- O protocolo PCP apenas permite o acesso a um semáforo quando todos os restantes semáforos de que uma tarefa necessita estão livres
- Para majorar o **tempo de bloqueio** (B) note-se que uma tarefa pode ser bloqueada por qualquer tarefa de **menor prioridade** que use um semáforo cujo **tecto seja pelo menos igual à sua prioridade**
- Note-se ainda que **cada tarefa só pode ser bloqueada uma vez**



e.g.	S_1	S_2	S_3
τ_1	1	2	0
τ_2	0	9	3
τ_3	8	7	0
τ_4	6	5	4

Protocolo de Tecto de Prioridades (PCP)

Análise de escalonabilidade (RM)

$$\left. \begin{aligned} \forall_{1 \leq i \leq n} \sum_{k=1}^i \frac{C_k}{T_k} + \frac{B_i}{T_i} &\leq i(2^{\frac{1}{i}} - 1) \\ \sum_{i=1}^n \frac{C_i}{T_i} + \max_{i=1 \dots n} \frac{B_i}{T_i} &\leq n(2^{\frac{1}{n}} - 1) \\ R_{wc_i} &= C_i + B_i + \sum_{k=1}^{i-1} \left\lceil \frac{R_{wc_i}}{T_k} \right\rceil C_k \end{aligned} \right\}$$

Apenas varia o cálculo de B_i

e.g.	C_i	T_i	B_i
τ_1	5	30	9
τ_2	15	60	8
τ_3	20	80	6
τ_4	20	100	0

e.g.	S_1	S_2	S_3
τ_1	1	2	0
τ_2	0	9	3
τ_3	8	7	0
τ_4	6	5	4



Protocolo de Tecto de Prioridades (PCP)

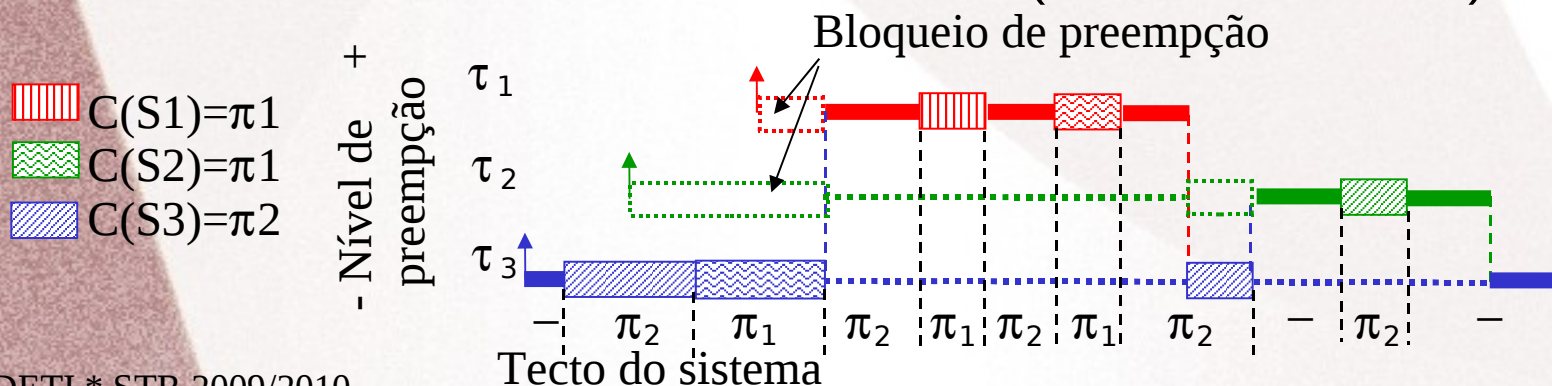
Propriedades:

- 😊 Bloqueios menores do que o PIP, **livre de bloqueios em cadeia, livre de deadlocks**,
- 😞 Mais difícil de concretizar (no TCB requer um campo para a prioridade herdada e outro para o semáforo onde a tarefa está bloqueada – para facilitar a transitividade da herança, requer ainda uma estrutura para os semáforos com os respectivos tectos e identificação das tarefas que os estão a usar – para facilitar a herança)
- 😞 Não é transparente para o programador (tectos dos semáforos não são locais às tarefas)

Existe uma **versão para EDF** em que as tarefas bloqueantes herdaram a deadline das bloqueadas e os tectos dos semáforos usam as deadlines relativas (nível de preempção – *preemption level*)

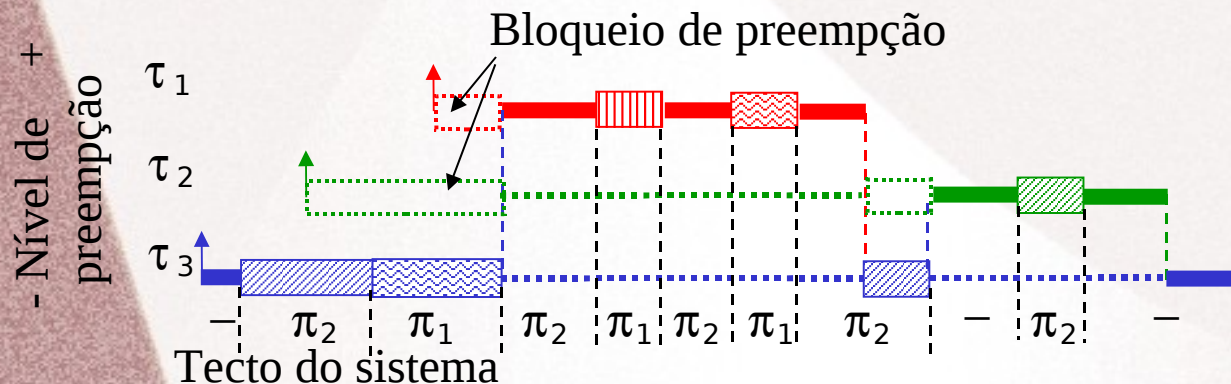
Política de Pilha de Recursos (SRP – Stack Resource Policy)

- Semelhante ao PCP mas com uma regra sobre o **início de execução**, para garantir que todos os semáforos necessários estão livres
- Usa igualmente o conceito de **tecto (ceiling) de prioridade**
- Define o **nível de preempção** (π *preemption level*) como a capacidade de uma tarefa causar preempção sobre outra (parâmetro estático).
- Uma tarefa só pode **iniciar execução** se o seu **nível de preempção for maior do que o da tarefa em execução** e do que **os tectos de todos os semáforos correntemente fechados (tecto do sistema)**.



Política de Pilha de Recursos (SRP)

- O protocolo SRP apenas permite o início de execução de uma tarefa quando todos os recursos de que necessita estão livres
- O majorante do **tempo de bloqueio** (B) é igual ao do protocolo PCP, apenas ocorre num momento diferente (início de execução)
- Cada tarefa pode ser bloqueada apenas uma vez por qualquer tarefa de **menor nível de preempção** que use um semáforo cujo **tecto seja pelo menos igual ao seu nível de preempção**



e.g.	S_1	S_2	S_3
τ_1	1	2	0
τ_2	0	9	3
τ_3	8	7	0
τ_4	6	5	4

Política de Pilha de Recursos (SRP)

Análise de escalonabilidade (RM)

$$\left. \begin{aligned} \forall_{1 \leq i \leq n} \sum_{k=1}^i \frac{C_k}{T_k} + \frac{B_i}{T_i} &\leq i(2^{\frac{1}{i}} - 1) \\ \sum_{i=1}^n \frac{C_i}{T_i} + \max_{i=1 \dots n} \frac{B_i}{T_i} &\leq n(2^{\frac{1}{n}} - 1) \\ R_{wc_i} &= C_i + B_i + \sum_{k=1}^{i-1} \left\lfloor \frac{R_{wc_i}}{T_k} \right\rfloor C_k \end{aligned} \right\} \text{Apenas varia cálculo de } B_i$$

Análise de escalonabilidade (EDF)

$$\begin{aligned} \forall_{1 \leq i \leq n} \sum_{k=1}^i \frac{C_k}{T_k} + \frac{B_i}{T_i} &\leq 1 \\ \sum_{i=1}^n \frac{C_i}{T_i} + \max_{i=1 \dots n} \frac{B_i}{T_i} &\leq 1 \end{aligned}$$

e.g.	C _i	T _i	B _i
τ ₁	5	30	9
τ ₂	15	60	8
τ ₃	20	80	6
τ ₄	20	100	0

e.g.	S ₁	S ₂	S ₃
τ ₁	1	2	0
τ ₂	0	9	3
τ ₃	8	7	0
τ ₄	6	5	4



Política de Pilha de Recursos (SRP)

Propriedades:

- 😊 Bloqueios menores do que o PIP, **livre de bloqueios em cadeia, livre de deadlocks**
- 😊 **Menor número de preempções** do que com o PCP, adequação intrínseca a **prioridades fixas ou a dinâmicas**, e a recursos com múltiplas unidades (e.g. array de buffers)
- 😊 Como não há bloqueios a meio da execução, as tarefas **não necessitam de um stack individual** (pode usar-se um *stack* único resultando em menores requisitos de memória)
- 😞 Mais difícil de concretizar (teste de preempção mais complicado, requer o cálculo do tecto do sistema, mais complexo ainda se se usam recursos com múltiplas unidades)
- 😞 Não é transparente para o programador (tectos dos semáforos...)

Resumo da Aula 7

- **Acesso exclusivo** a recursos partilhados: **bloqueio**
- A **inversão de prioridades**: necessidade de a limitar e analisar
- **Técnicas básicas** para acesso exclusivo a recursos partilhados
- **Herança** de prioridades (*Priority Inheritance Protocol – PIP*)
- Protocolo de **tecto de prioridades** (*Priority Ceiling Protocol – PCP*)
- Política de **pilha de recursos** (*Stack Resource Protocol- SRP*)