FUNDAÇÃO EDSON QUEIROZ
UNIVERSIDADE DE FORTALEZA
ENSINANDO E APRENDENDO

T569 –SISTEMAS DE TEMPO REAL

Aula 14

Prof. Marcelo Sousa

Agenda

- Highest Locker Protocol (HLP)
- Priority Ceiling Protocol (PCP)
- Gerenciamento de dependências das tasks

Highest Locker Protocol (HLP)

- Evolução do PIP

- Cada **Recurso Crítico** é associado a uma **Prioridade Teto** (*ceiling priority*)

- A **Prioridade Teto** de cada CR é definida como o valor máximo dentre as prioridades de todas as tasks que podem utilizar o recurso.

Highest Locker Protocol (HLP)

- Modificação no Algoritmo:
  - Tão logo uma *task* trave um recurso, aquela task receberá o valor da *ceiling priority* daquele recurso.
  - Então, se uma task possuir vários recursos travados, ela herdará o valor mais alto de *ceiling priority*.
- As necessidades de utilização do recurso devem ser avaliadas antes do tempo de compilação.

Highest Locker Protocol (HLP)

- Definição do valor do teto
  - $Ceil(RC_i) = \max(pri(T_1), ..., pri(T_n))$

Highest Locker Protocol (HLP)

- **Quando em HLP, uma task que assumir o controle do recurso opera com a prioridade teto.**

- Isso soluciona o problema de inversão ilimitada de *prioridade*, *deadlock* e *chain blocking*.

Highest Locker Protocol (HLP)

- **Teorema 1**: *ao utilizar HLP para gerenciamento do compartilhamento de recursos, quando uma task trava um recurso necessário, ela não é bloqueada de modo algum.*

Highest Locker Protocol (HLP)

- **Corolário 1**: *Sob HLP, antes de uma task poder adquirir um recurso, todos os recursos que podem ser necessários devem estar livres.*

- **Corolário 2**: *Uma task não sofre de **chain bloking** em HLP*

Highest Locker Protocol (HLP)

- Deficiências:
  - Apesar de evitar *deadlock*, *chain blocked* e inversão ilimitada de prioridade, o HLP abre espaço para inversão por herança (*inheritance-related inversion*)
  - ***Inheritance-related Inversion****:* Quando uma task de baixa prioridade eleva sua prioridade ao travar um recurso e outra task de prioridade intermediária que não utiliza o recurso é impossibilitada de ser executada.

Highest Locker Protocol (HLP)

- Exemplo 1:
  - Sistema com Tasks T1, T2, T3, T4, T5 com prioridades 1, 2, 3, 4 e 5
  - T1 T2 T3 ➔ CR1 ➔ Ceil(max(1,2,3))
  - T1 T4 T5 ➔ CR2 ➔ Ceil(max(1,4,5))
  - Fluxo:
    - T1 ➔ CR2 (pri(5))
      - T3 e T2 não serão executadas e estão em *inheritance-related inversion*

Highest Locker Protocol (HLP)

- Protocolo raramente utilizado em aplicações reais, justamente pelo problema de *inheritance-related inversion.*

- Necessária para apresentação do ***Priority Cealing Protocol***

Priority Ceiling Protocol (PCP)

- Estende as idéias dos protocolos anteriores
- Evita *deadlock*, *chain blocked*, inversão ilimitada e minimiza inversão herdada.
- **Principal diferença:** Não garante o acesso ao recurso mesmo que o recurso esteja livre.

Priority Ceiling Protocol (PCP)

- Associa um valor de prioridade teto para cada recurso que obedece os mesmo critérios do HLP

- CSC - *Current System Ceiling*: Valor máximo de todos os recursos que estão sendo utilizados pelo sistema naquele momento.
  - Variável do sistema operacional utilizada para manter o controle do valor máximo do teto.

Priority Ceiling Protocol (PCP)

- ***Resource Grant Rule****:* regra aplicada quando uma task solicita acesso a um recurso.
  - ***Resource request clause****:*
    - *Se a task Ti está utilizando um recurso com o mesmo valor de CSC, então o acesso é garantido;*
    - *Caso não esteja, Ti não terá acesso a CRj, a menos que a prioridade da task seja maior que CSC;*
    - *Se garantido o acesso, o valor de CSC é atualizado caso CSC < Ceil(CRj).*

Priority Ceiling Protocol (PCP)

- ***Resource Grant Rule**:*
  - *Inheritance clause:*
    - *Quando uma task falhou nos testes para acessar o recurso:*
      - *A task é bloqueada*
      - *A task que está com acesso ao recurso herda a prioridade da task que está bloqueada, se a prioridade da task que está com o recurso for menor que a da task bloqueada.*

Priority Ceiling Protocol (PCP)

- ***Resource Release Rule***
  - Se uma *task* libera um recurso e se o teto (*ceil*) desse recurso é igual ao CSC:
    - Uma análise de todas os recursos utilizados deve ser realizada e o valor do CSC deve assumir o maior valor/
  - Caso contrário:
    - O valor se mantém.

Priority Ceiling Protocol (PCP)

- Exemplo 1:
  - Task: T1, T2, T3, T4
  - Critical Resource: CR1 CR2
  - Prioridades: 10, 12, 15, 20

  - T1, T2, T3 ➔ CR1
  - T1 T4 ➔ CR2
  - Ceil(CR1) = 15; Ceil (CR2) = 20

Priority Ceiling Protocol (PCP)

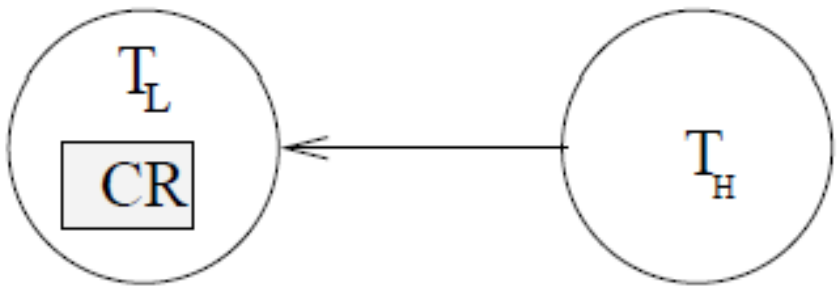
- Situação 1:
  - Instante: T1 está em execução após conseguir CR1
  - CSC = 15
  - T4 → Ready , tendo maior prioridade, interrompe T1
  - Após algum tempo T4 → CR2
    - Prioridade 20 é maior que CSC (15).
    - T4 garante CR2 pelo *resource request clause*
  - Quando completa sua execução, T1 volta a ser executada.

Priority Ceiling Protocol (PCP)

- Situação 2:
  - Instante: T1 está em execução após conseguir CR1
  - CSC = 15
  - T3 → Ready, tendo maior prioridade, interrompe T1
  - Após algum tempo T3 → CR1
    - Prioridade 15 não é maior que CSC (15).
    - T3 não recebe o acesso a CR1
    - T3 irá bloquear
    - T1 irá herdar a prioridade de T3, pri(T1) de 10 para 15

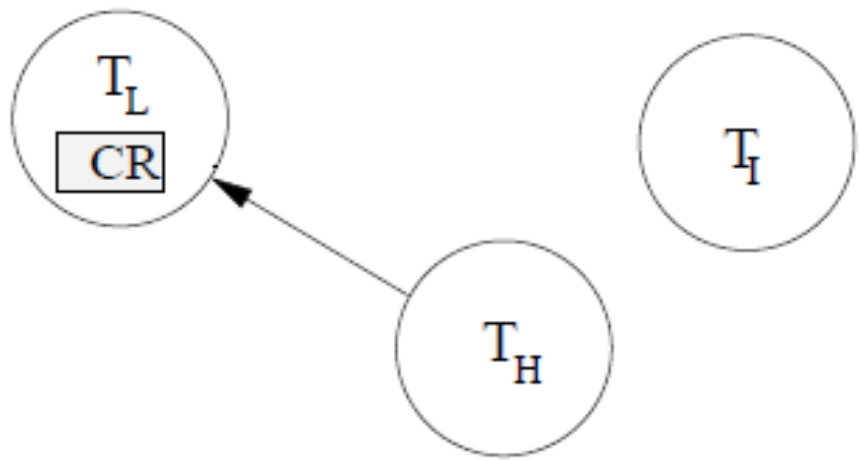
Inversão de Prioridade com PCP

- Inversão Direta:
  - Quando uma task de maior prioridade espera por uma task de menor prioridade liberar uma recurso necessário.



Inversão de Prioridade com PCP

- *Inheritance Related Inversion:*
  - Considere a situação onde T_L está travando um CR e uma T_H está esperando por CR. Então, T_L subirá para a prioridade de T_H por *inheritance clause* do PCP. Como resultado, T_I que não necessitará do recurso, sofrerá de ***inversão por herança.***

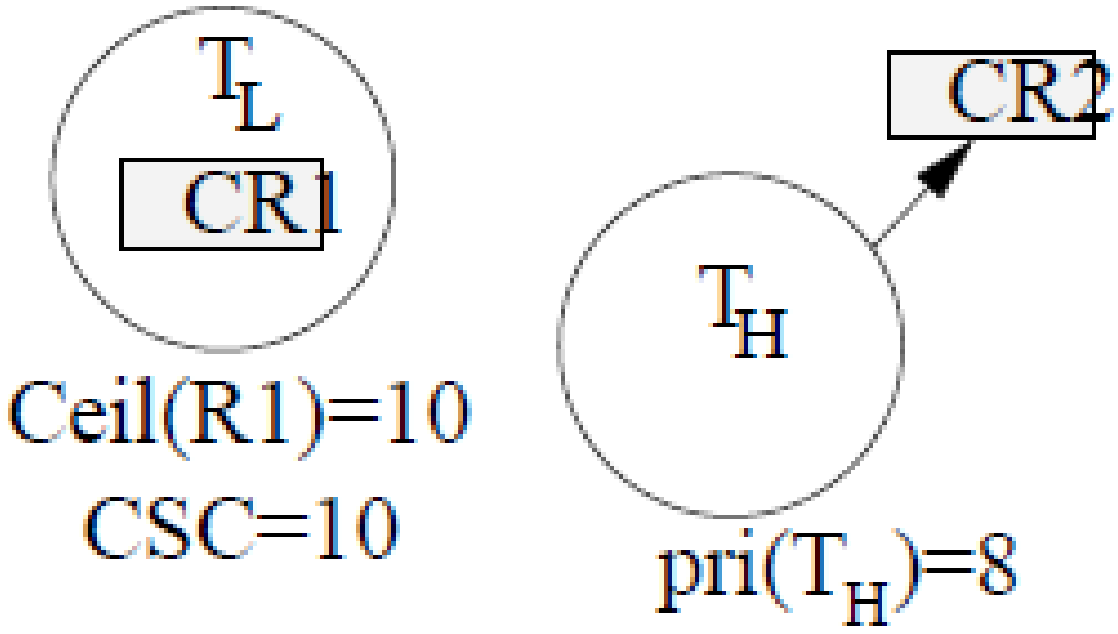


Inversão de Prioridade com PCP

- *Avoidance-Related Inversion:*
  - Quando uma task requisita um recurso, sua prioridade é checada com CSC. A task só terá acesso ao recurso se sua prioridade for maior que CSC. O acesso ao recurso pode ser negado mesmo se o recurso estiver liberado.
  - Uma *task* que possua maior prioridade que a *task* em execução, mas menor que CSC e necessita de uma recurso livre, sofre de *Avoidance-Related Inversion*

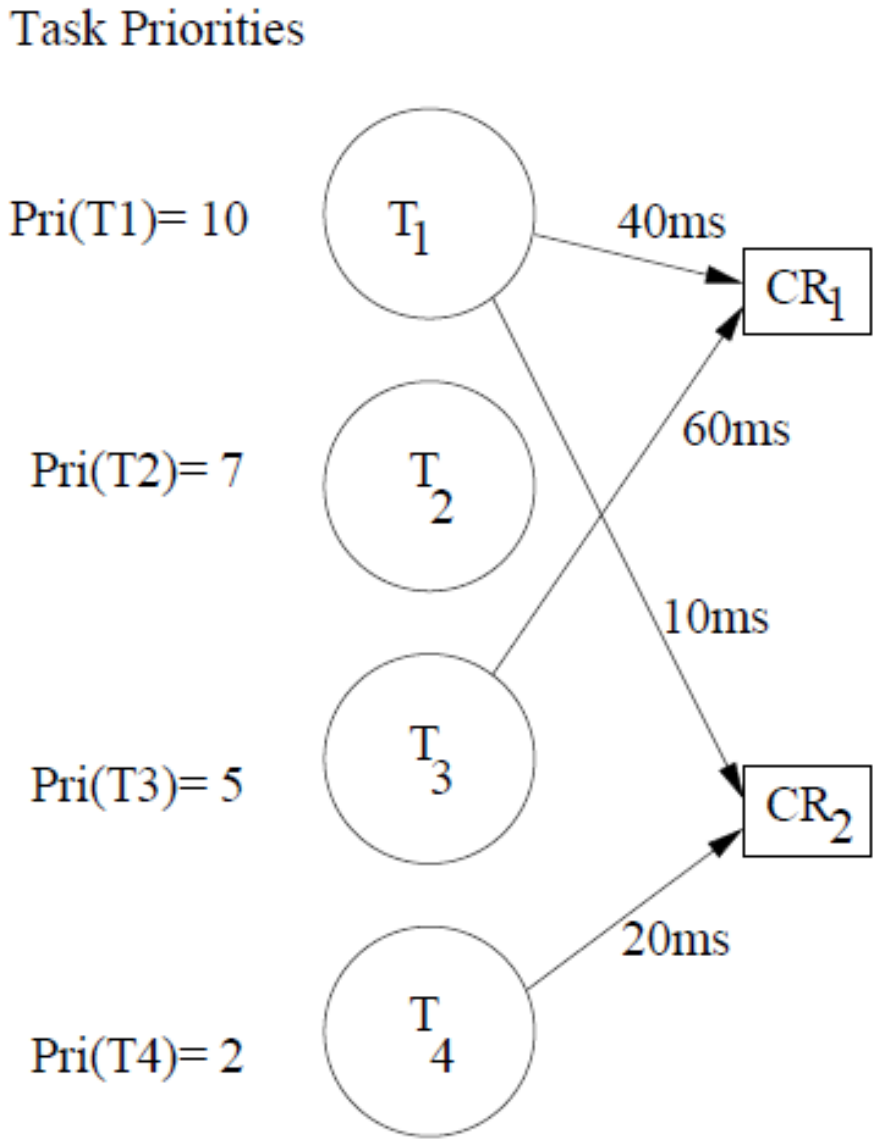
Inversão de Prioridade com PCP

- *Avoidance-Related Inversion:*



Inversão de Prioridade com PCP

- Exemplo 2:
  - Compute os diferentes tipos de inversões de prioridades que cada uma das tasks pode vir a sofrer no pior caso.



Inversão de Prioridade com PCP

- Exemplo 2:
  - T1:
    - Direta: T3 por 60ms
    - Direta: T4 por 40ms
  - T2:
    - Herança: T3 por 60ms
    - Herança: T4 por 20ms
  - T3
    - Herança: T4 por 20ms
    - Avoidance: T4 por 20ms

Inversão de Prioridade com PCP

- Exemplo 2:

Task	Direct			Inheritance			Avoidance		
	T_2	T_3	T_4	T_2	T_3	T_4	T_2	T_3	T_4
T_1	x	60	40	x	x	x	x	x	x
T_2	x	x	x	x	60	20	x	x	x
T_3	x	x	x	x	x	20	x	x	20

Próxima Aula

- Prática:
  - Compatilhamento de recursos no FreeRTOS