Sistemas de Tempo Real:

Escalonamento com Garantia

Rômulo Silva de Oliveira
Departamento de Automação e Sistemas - DAS – UFSC

romulo@das.ufsc.br
http://www.das.ufsc.br/~romulo

Escalonamento com Garantia

- Executivo Cíclico
- Prioridades + Teste de Escalonabilidade
  - Prioridades Variáveis com Teste de Escalonabilidade
  - Prioridades Fixas com Teste de Escalonabilidade

Executivo Cíclico – Introdução

- Também chamado de
  - Time-driven scheduler ou Clock-driven scheduler
- Indicado para tarefas com deadline hard
- Indicado para tarefas periódicas
- Parâmetros das tarefas precisam ser conhecidos a priori
- Parâmetros das tarefas não mudam durante a execução

- Tabela descrevendo a escala de execução é criada no projeto
  - Durante a execução esta escala é repetida ciclicamente
- Garantia da escalonabilidade está na inspeção da tabela
  - Se todos os deadlines são cumpridos, sistema é escalonável

- É a forma tradicional de implementar sistemas de tempo real críticos
- Tem a vantagem de ser completamente determinístico

Executivo Cíclico – Introdução

- O design do software é concorrente
- A implementação pode ser através de
  - Um conjunto de procedimentos
- Procedimentos são mapeados sobre
  um conjunto de ciclos menores (minor cycles)
  que formam juntos a escala completa ou ciclo maior (major cycle)

- O Ciclo Menor determina o mínimo ciclo de tempo possível
- O Ciclo Maior determina o máximo ciclo de tempo possível

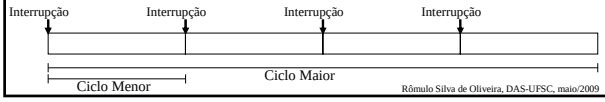
- É alocado tempo para o WCET de cada tarefa a cada período dela
- Não existe preempção (exclusão mútua assegurada)

Executivo Cíclico – Solução Clássica

- Toda a sequência de execução é repedida a cada
  Major Cycle (Ciclo Maior)
  - Todas as execuções de Ciclo Maior são iguais

- O Ciclo Maior é dividido em uma sequência de
  Minor Cycles (Ciclo Menor)
  - Ciclos Menores podem ser diferentes entre si

- Interrupção do Timer indica início de cada Ciclo Menor
- Decisões de escalonamento somente no início de cada Ciclo Menor



Executivo Cíclico – Solução Clássica

- Duração do Ciclo Maior indica instante no tempo
  equivalente
  ao instante inicial da tabela

- É possível voltar ao início da tabela e repetir tudo novamente

- Valor natural para a duração do Ciclo Maior (L):
  Mínimo Múltiplo Comum dos períodos das tarefas

Executivo Cíclico – Solução Clássica

- Ciclo menor (f) precisa ser longo o suficiente para permitir que todos os jobs terminem sua execução dentro dele
  - Evita preempções
  - Evita chaveamento de contexto
  - Fornece exclusão mútua

$$f \geq \underset{1 \leq i \leq n}{Max}(C_i)$$

- Conveniente que todos os períodos de tarefa sejam números múltiplos da duração do Ciclo Menor (f)
  - Interrupção do timer a cada Ciclo Menor

$$\left\lfloor \frac{P_i}{f} \right\rfloor - \frac{P_i}{f} = 0$$

- Vários jobs podem ser executados dentro de um mesmo Ciclo Menor

Executivo Cíclico – Exemplo

Tarefa Ti	Período Pi	Tempo de computação Ci
T1	25	10
T2	25	8
T3	50	5
T4	50	4
T5	100	2

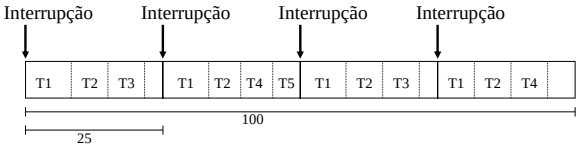
Mímimo múltiplo comum dos períodos = 100
Máximo divisor comum dos períodos = 25

Executivo Cíclico – Exemplo

```
While( true ) {
  wait_for_interrupt;
  funcao_t1(); funcao_t2(); funcao_t3();
  wait_for_interrupt;
  funcao_t1(); funcao_t2(); funcao_t4(); funcao_t5();
  wait_for_interrupt;
  funcao_t1(); funcao_t2(); funcao_t3();
  wait_for_interrupt;
  funcao_t1(); funcao_t2(); funcao_t4();
}
```


Executivo Cíclico – Exemplo

Tarefa Ti	Período Pi	Tempo de computação Ci
T1	25	10
T2	25	8
T3	50	5
T4	50	4
T5	100	2



Executivo Cíclico – Solução Clássica

- Tabela indica que tarefas fazem parte de cada Ciclo Menor
  - Para toda a duração do Ciclo Maior
- Existem conjuntos de tarefas para os quais é impossível aplicar a solução clássica
  - Necessário adaptar as tarefas
- Tarefas aperiódicas podem ser executadas dentro de cada Ciclo Menor depois que todas os jobs garantidos alocados àquele Ciclo Menor já foram concluídos
  - Executam até a próxima interrupção do timer (novo Ciclo Menor)

Executivo Cíclico – Comentários

- Overhead de escalonamento no início de cada Ciclo Menor
- Se ocorreu overrun do Ciclo Menor anterior, ação é necessária
- Job interrompido era aperiódica não garantida
  - Simplesmente preempta o job
- Job interrompido era periódico garantido mas não crítico
  - Preempta o job, sinaliza ocorrência de uma falta temporal
- Job interrompido era periódico garantido e crítico
  - Grave falta temporal
  - Continua executando este job para garantir consistência dos dados
    - Espera que folga do próximo Ciclo Menor resolva o problema
  - Dispara tratamento de exceção, pânico

Executivo Cíclico – Comentários

- Pode ser desenvolvido a mão sem nenhum método especial
  - No caso de sistemas pequenos/simples
- Dado que a tabela de execução é construída em projeto
  - Algoritmos complexos também podem ser usados
  - Trata-se de um problema de otimização com complexidade exponencial
  - Soluções ótimas são possíveis até um certo tamanho de sistema
  - Para sistemas grandes são usadas heurísticas (sub-ótimas)
  - Uso de meta-heurísticas é popular (busca tabu, genético, simulated annealing)
- Objetivos secundários podem existir
  - Distribuir as folgas uniformemente para favorecer aperiódicas
  - Reduzir jitter de saída das tarefas

Executivo Cíclico – Comentários

- Na verdade nenhum processo precisa realmente existir em tempo de execução
  - Cada Ciclo Menor pode ser apenas uma sequência de chamadas de subrotinas
- As subrotinas podem compartilhar um espaço de endereçamento comum
  - Podem passar dados entre eles
  - Esses dados não precisam ser protegidos (via um semáforo, por exemplo) porque o acesso concorrente não é possível
- Todos os períodos das "tarefas" devem ser múltiplos do tempo de ciclo menor

Executivo Cíclico – Comentários

- Dificuldade em incorporar processos com períodos longos
- Atividades esporádicas são difíceis (as vezes impossíveis) de serem incorporadas
- O executivo cíclico é difícil de construir e manter — é um problema NP-hard
- Qualquer "tarefa" com tempo de computação maior precisará ser dividido em um número fixo de procedimentos com tamanho mediano
  - Isto pode prejudicar a estrutura do código, sendo mais sujeito a bugs
- Métodos mais flexíveis de escalonamento são difíceis de suportar
- Na realidade,
  Determinismo da escala de execução não é necessário
  necssário é previsibilidade quanto ao cumprimento dos deadlines

Escalonamento com Garantia

- Executivo Cíclico
- Prioridades + Teste de Escalonabilidade
  - Prioridades Variáveis com Teste de Escalonabilidade
  - Prioridades Fixas com Teste de Escalonabilidade

Prioridades Variáveis – Introdução

- Cada tarefa recebe uma prioridade que varia ao longo do tempo
- Prioridade leva em conta informações relativas à execução
- Diferentes jobs da mesma tarefa podem receber prioridades diferentes
- Cada Job em particular pode receber prioridade fixa ou variável
- A escala de execução só é conhecida durante a execução
- Necsssário Teste de Escalonabilidade
  - Para saber antes se todos os deadlines estão garantidos ou não
- Existem três tipos principais de testes de escalonabilidade
  - Baseados na **taxa de utilização** do CPU
  - Baseados na **carga imposta** ao CPU (*processor demand*)
  - Baseados no **tempo de resposta**

Prioridades Variáveis - Introdução

- ***EDF – Earliest Deadline First***
  - Inversamente proporcional ao deadline absoluto
  - Ótimo em relação aos critérios de prioridades variáveis
- ***LSF (LST ou LLF) – Least Slack First***
  - Inversamente proporcional ao tempo livre (*laxity* ou *slack*)
  - Ótimo em relação aos critérios de prioridades variáveis
  - Overhead maior que EDF
- ***FCFS –First Come First Served***
  - Inversamente proporcional ao tempo de espera por serviço
  - Não é ótimo com respeito ao cumprimento de deadlines
- **EDF é o mais usado**

Prioridades Variáveis – Teste para EDF

- Supondo um conjunto de **n** tarefas
  - independentes e periódicas
- EDF como política de atribuição de prioridades
- Se D=T, sistema é escalonável quando: $\sum_{i=1}^{N} (\frac{C_i}{P_i}) \leq 1$
  - Permite usar 100% do processador mantendo os deadlines
  - Teste exato
- Se D<T, sistema é escalonável quando: $\sum_{i=1}^{N} (\frac{C_i}{D_i}) \leq 1$
  - Teste suficiente
- Para D arbitrário, sistema é escalonável quando: $\sum_{i=1}^{N} (\frac{C_i}{\min(D_i, Pi)}) \leq 1$
  - Teste suficiente

Prioridades Variáveis – Teste para EDF

- Existem testes de escalonabilidade mais complexos para EDF
- São menos pessimistas
- Porém requerem esforço computacional maior

Prioridades Variáveis – LSF

- **Least Slack First - LSF**
- Quanto menos tempo livre (slack), maior a prioridade
- Ótimo quando EDF é ótimo
- Prioridade das tarefas na fila de aptos aumenta com passar do tempo
- Prioridade da tarefa em execução mantém-se constante
- Gera número maior de chaveamento de contextos que EDF
  - maior overhead
- **Não apresenta vantagens face a EDF**
- EDF:
  - As prioridades de todas as tarefas aptas e em execução aumentam
    - de igual modo com o passar do tempo

Prioridades Variáveis – Comentários

- Implementação é mais complexa do que com prioridade fixa
  - requer um kernel que aceita prioridades variáveis
- *Overhead* de execução pode ser elevado caso seja necessária reordenação dinâmica da fila de aptos (depende do algoritmo)
- Instabilidade face a sobrecargas
  - Não é possível saber antecipadamente quais tarefas vão perder deadline
- Escalonabilidade é superior em EDF do que em prioridade fixa
  - Qualquer sistema escalonável com prioridade fixa também será escalonável com EDF
  - O contrário não é verdadeiro
- Entretanto, prioridade fixa é mais usado na prática

Exemplo escalonabilidade: EDF versus RM

tarefas periódicas	C_i	P_i	D_i
tarefa A	10	20	20
tarefa B	25	50	50

$$U = \sum_{i=1}^{n} C_i / P_i \leq 1. \quad [3]$$

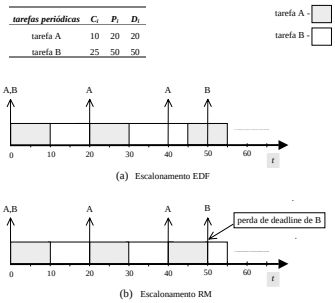
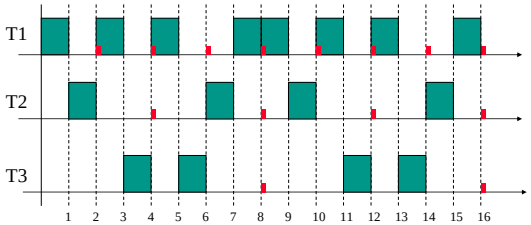


Figura 2.6: Escalas produzidas pelo (a) EDF e (b) RM

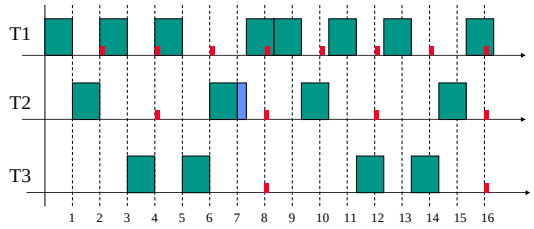
Exemplo sobrecarga: EDF versus RM

- T1: C1=1 P1=D1=2
- T2: C2=1 P2=D2=4
- T3: C3=2 P3=D3=8
- Execução normal com EDF



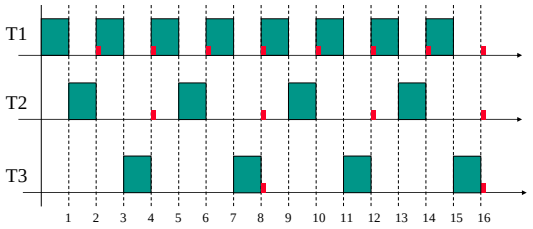
Exemplo sobrecarga: EDF versus RM

- T1: C1=1 P1=D1=2
- T2: C2=1 P2=D2=4
- T3: C3=2 P3=D3=8
- Sobrecarga devido a falha do projeto com EDF



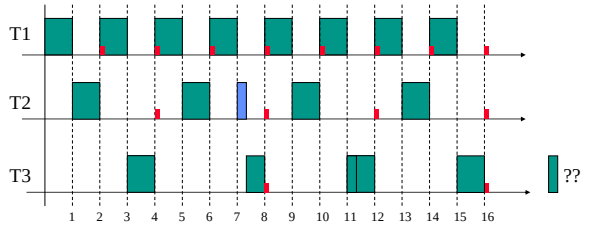
Exemplo sobrecarga: EDF versus RM

- T1: C1=1 P1=D1=2
- T2: C2=1 P2=D2=4
- T3: C3=2 P3=D3=8
- Execução normal com RM



Exemplo sobrecarga: EDF versus RM

- T1: C1=1 P1=D1=2
- T2: C2=1 P2=D2=4
- T3: C3=2 P3=D3=8
- Sobrecarga devido a falha do projeto com RM



Escalonamento com Garantia

- Executivo Cíclico
- Prioridades + Teste de Escalonabilidade
  - Prioridades Variáveis com Teste de Escalonabilidade
  - Prioridades Fixas com Teste de Escalonabilidade

Escalonamento Baseado em Prioridades Fixas

- Aplicação composta por tarefas (processos)
- Estados de uma tarefa:
  - Liberada (pronta para executar, apta, ready)
  - Executando (running)
  - Suspensa esperando pela próxima ativação
  - Outros estados serão acrescentados mais adiante
- Em geral escalonamento é preemptivo
- Tarefas possuem **prioridade fixa** definida em projeto
- Garantia exige
  - Tarefas periódicas ou esporádicas
  - Tempo máximo de computação conhecido
  - Teste de escalonabilidade apropriado

Prioridade Fixa – Rate Monotonic

- Quanto menor o período, mais alta a prioridade
- Ótimo quando
  - Tarefas são periódicas
  - Deadline é sempre igual ao período
- Exemplo:
  - Tarefas T1 T2 T3
  - Períodos P1=30 P2=40 P3=50
  - Prioridades p1=1 p2=2 p3=3
- Cuidado!
  - Número menor indica prioridade maior
  - Muitas vezes é o contrário

Prioridade Fixa – Análise da Utilização

- Utilização de uma tarefa:
  - Tempo máximo de computação dividido pelo período Ui = Ci / Pi
  - T1 tem C1=12 e P1=50, então U1 = C1 / P1 = 12 / 50 = 0.24
- Teste para Rate Monotonic, sistema é escalonável se:

$$\sum_{i=1}^{N} \left(\frac{C_i}{P_i}\right) \leq N\left(2^{1/N} - 1\right)$$

- Para N=1 utilização máxima é 100%
- Para N grandes utilização máxima tende para 69.3%
- Baseado no conceito de Instante Crítico
- Teste é suficiente mas não necessário

Prioridade Fixa – Análise da Utilização

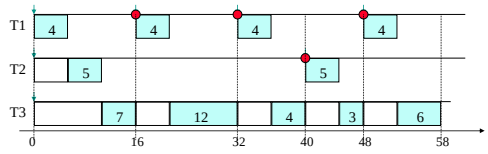
N	Limiar de Utilização
1	100.0%
2	82.8%
3	78.0%
4	75.7%
5	74.3%
10	71.8%
infinito	69.3%

Prioridade Fixa – Análise da Utilização

- Exemplo:

	T1	T2	T3
– Períodos	P1=16	P2=40	P3=80
– Computação	C1=4	C2=5	C3=32
– Utilização	U1=0.250	U2=0.125	U3=0.400
– Prioridades	p1=1	p2=2	p3=3

- Utilização total é 0.775, abaixo do limite 0.780

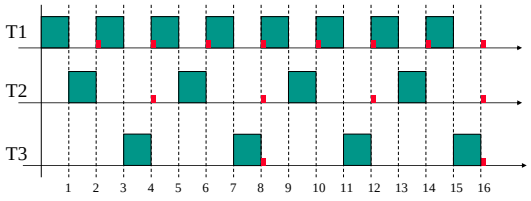


Prioridade Fixa – Análise da Utilização

- Exemplo:

	T1	T2	T3
– Períodos	P1=2	P2=4	P3=8
– Computação	C1=1	C2=1	C3=2
– Utilização	U1=0.500	U2=0.250	U3=0.250
– Prioridades	p1=1	p2=2	p3=3

- Utilização total é 1, acima do limiar 0.780, mas conjunto é escalonável



Prioridade Fixa – Análise da Utilização

- Testes baseados em utilização
  - Não são gerais
  - Não são exatos
  - Mas são rápidos, O(N)
- Exemplo de teste necessário mas não suficiente para este modelo de tarefas

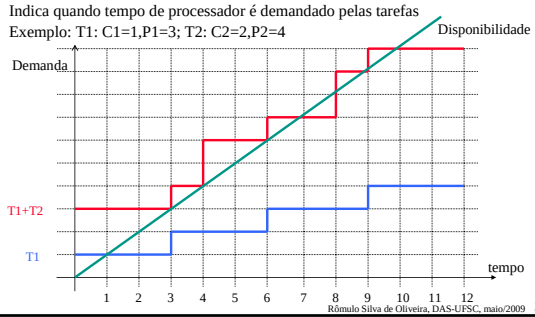
$$\sum_{i=1}^{N} \left(\frac{C_i}{P_i}\right) \leq 1$$

Prioridade Fixa – Análise do Tempo de Resposta

- Limitações da análise baseada em Utilização
  - Não é exata
  - Aplicável apenas a modelos de tarefas muito simples
- Análise baseada em **Tempo de Resposta**
  - Abordagem analítica calcula tempo de resposta no pior caso
  - Tempo de resposta de cada tarefa é comparado com o deadline da tarefa
  - Baseada no conceito de Função Demanda de Tempo

Prioridade Fixa – Análise do Tempo de Resposta

- Função Demanda de Tempo (Time-Demand Function)
  - Base da análise do tempo de resposta
  - Indica quando tempo de processador é demandado pelas tarefas
  - Exemplo: T1: C1=1,P1=3; T2: C2=2,P2=4



Prioridade Fixa – Análise do Tempo de Resposta

- Como calcular o tempo de resposta de cada tarefa ?
- Para a tarefa mais prioritária temos R1 = C1
- Demais tarefas sofrem **Interferência** das tarefas com prioridade maior
- Neste caso, Ri = Ci + Ii
- Interferência é máxima a partir do **Instante Crítico**
  - Todas as tarefas são liberadas simultaneamente
  - Suposto instante zero na análise

Prioridade Fixa – Análise do Tempo de Resposta

- Seja Tj uma tarefa com prioridade maior que Ti
- Quantas vezes Tj pode acontecer durante a execução de Ti ?

$$\left\lceil \frac{R_i}{P_j} \right\rceil$$

  - Qual a interferência total de Tj sobre Ti ?

$$\left\lceil \frac{R_i}{P_j} \right\rceil \times C_j$$

  - Qual a interferência total sobre Ti ?

$$\sum_{j \in hp(i)} \left\lceil \frac{R_i}{P_j} \right\rceil \times C_j$$

Prioridade Fixa – Análise do Tempo de Resposta

- O tempo máximo de resposta de Ti é Ri = Ci + Ii

$$R_i = C_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{P_j} \right\rceil \times C_j$$

  - Equação é recursiva
  - Calculada através de iterações sucessivas, até:
    - Tempo de resposta passar do deadline
    - Resultado convergir, iteração x+1 igual a iteração x

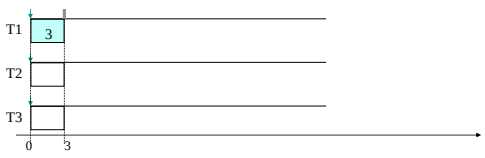
$$w_i^{x+1} = C_i + \sum_{j \in hp(i)} \left\lceil \frac{w_i^x}{P_j} \right\rceil \times C_j \qquad w_i^0 = C_i$$

Prioridade Fixa – Análise do Tempo de Resposta

- Exemplo:

	T1	T2	T3
– Períodos	P1=7	P2=12	P3=20
– Computação	C1=3	C2=3	C3=5
– Prioridades		p1=1	p2=2 p3=3

- R1 = C1 = 3



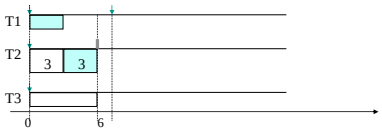
Prioridade Fixa – Análise do Tempo de Resposta

- Análise da tarefa T2:

$$w_2^0 = C_2 = 3$$

$$w_2^1 = C_2 + \left\lceil \frac{w_2^0}{P_1} \right\rceil \times C_1 = 3 + \left\lceil \frac{3}{7} \right\rceil \times 3 = 6$$

$$w_2^2 = C_2 + \left\lceil \frac{w_2^1}{P_1} \right\rceil \times C_1 = 3 + \left\lceil \frac{6}{7} \right\rceil \times 3 = 6$$



Prioridade Fixa – Análise do Tempo de Resposta

- Análise da Tarefa T3:

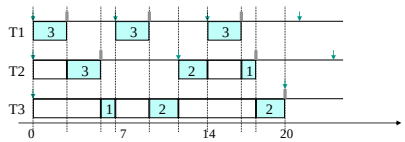
$$w_3^0 = C_3 = 5$$

$$w_3^1 = C_3 + \sum_{j \in hp(3)} \left\lceil \frac{w_3^0}{P_j} \right\rceil \times C_j = 5 + \left\lceil \frac{5}{7} \right\rceil \times 3 + \left\lceil \frac{5}{12} \right\rceil \times 3 = 11$$

$$w_3^2 = 5 + \left\lceil \frac{11}{7} \right\rceil \times 3 + \left\lceil \frac{11}{12} \right\rceil \times 3 = 14 \qquad w_3^3 = 5 + \left\lceil \frac{14}{7} \right\rceil \times 3 + \left\lceil \frac{14}{12} \right\rceil \times 3 = 17$$

$$w_3^4 = 5 + \left\lceil \frac{17}{7} \right\rceil \times 3 + \left\lceil \frac{17}{12} \right\rceil \times 3 = 20 \qquad w_3^5 = 5 + \left\lceil \frac{20}{7} \right\rceil \times 3 + \left\lceil \frac{20}{12} \right\rceil \times 3 = 20$$

Prioridade Fixa – Análise do Tempo de Resposta



- Exemplo:

	T1	T2	T3
– Períodos	P1=7	P2=12	P3=20
– Computação	C1=3	C2=3	C3=5
– Prioridades	p1=1	p2=2	p3=3
– Tempo Máximo de Resposta	R1=3	R2=6	R3=20

Prioridade Fixa – Análise do Tempo de Resposta

- Exemplo:

	T1	T2	T3
– Períodos	P1=2	P2=4	P3=8
– Computação	C1=1	C2=1	C3=2
– Utilização	U1=0.500	U2=0.250	U3=0.250
– Prioridades	p1=1	p2=2	p3=3

- Utilização total é 1, acima do limiar 0.780
  mas conjunto é escalonável
- Aplicando o cálculo do tempo de resposta temos
- Análise da tarefa T1:
  R1 = C1 = 1

Prioridade Fixa – Análise do Tempo de Resposta

- Exemplo:

	T1	T2	T3
– Períodos	P1=2	P2=4	P3=8
– Computação	C1=1	C2=1	C3=2
– Utilização	U1=0.500	U2=0.250	U3=0.250
– Prioridades	p1=1	p2=2	p3=3

- Análise da tarefa T2:

$$w_2^0 = C_2 = 1 \qquad w_2^1 = C_2 + \left\lceil \frac{w_2^0}{P_1} \right\rceil \times C_1 = 1 + \left\lceil \frac{1}{7} \right\rceil \times 1 = 2$$

$$w_2^2 = C_2 + \left\lceil \frac{w_2^1}{P_1} \right\rceil \times C_1 = 1 + \left\lceil \frac{2}{7} \right\rceil \times 1 = 2$$

Prioridade Fixa – Análise do Tempo de Resposta

- Exemplo:

	T1	T2	T3
– Períodos	P1=2	P2=4	P3=8
– Computação	C1=1	C2=1	C3=2
– Utilização	U1=0.500	U2=0.250	U3=0.250
– Prioridades	p1=1	p2=2	p3=3

- Análise da Tarefa T3:

$$w_3^0 = C_3 = 2 \qquad w_3^1 = C_3 + \sum_{j \in hp(3)} \left\lceil \frac{w_3^0}{P_j} \right\rceil \times C_j = 2 + \left\lceil \frac{2}{2} \right\rceil \times 1 + \left\lceil \frac{2}{4} \right\rceil \times 1 = 4$$

$$w_3^2 = 2 + \left\lceil \frac{4}{2} \right\rceil \times 1 + \left\lceil \frac{4}{4} \right\rceil \times 1 = 5 \qquad w_3^3 = 2 + \left\lceil \frac{5}{2} \right\rceil \times 1 + \left\lceil \frac{5}{4} \right\rceil \times 1 = 7$$

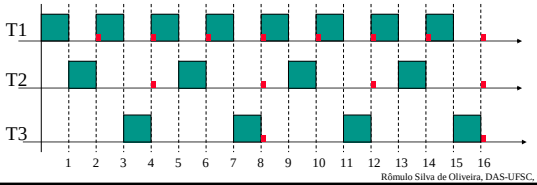
$$w_3^4 = 2 + \left\lceil \frac{7}{2} \right\rceil \times 1 + \left\lceil \frac{7}{4} \right\rceil \times 1 = 8 \qquad w_3^5 = 2 + \left\lceil \frac{8}{2} \right\rceil \times 1 + \left\lceil \frac{8}{4} \right\rceil \times 1 = 8$$

Prioridade Fixa – Análise do Tempo de Resposta

- Exemplo:

	T1	T2	T3
– Períodos	P1=2	P2=4	P3=8
– Computação	C1=1	C2=1	C3=2
– Utilização	U1=0.500	U2=0.250	U3=0.250
– Prioridades	p1=1	p2=2	p3=3
– Resposta	R1=1	R2=2	R3=8

- Utilização total é 1, acima do limiar 0.780, mas conjunto é escalonável



Prioridade Fixa – Análise do Tempo de Resposta

- Teste de escalonabilidade **exato** (suficiente e necessário)
- Deadline pode ser menor que o período
  - Basta comparar o tempo de resposta com o deadline
- Deadline maior que o período exige análise mais complexa
  - Tarefa pode interferir com ela mesma
- Tarefas esporádicas podem ser tratadas como periódicas
  - Intervalo mínimo entre ativações é usado como período
- A forma como prioridades são atribuidas NÃO é importante
  - Funciona pois "hp(i)" sempre indica as tarefas mais prioritárias do que a tarefa "i"

Prioridade Fixa – Deadline Monotonic

- Quanto menor o deadline, maior a prioridade
- Ótimo quando deadline é menor ou igual ao período
- Exemplo:

– Tarefas	T1	T2	T3	T4
– Períodos	P1=20	P2=15	P3=10	P4=20
– WCET	C1=3	C2=3	C3=4	C4=3
– Deadline	D1=5	D2=7	D3=10	D4=20
– Prioridades	p1=1	p2=2	p3=3	p4=4
– Tempo máximo de resposta	R1=3	R2=6	R3=10	R4=20
– Caso fosse RM	R1=10	R2=7	R3=4	R4=20

Prioridade Fixa – Resumo

- Cada tarefa recebe uma prioridade fixa
- Teste é desenvolvido para examinar a escalonabilidade
- Dois tipos de análise
- **Análise da Utilização**
  - Utiliza o valor C/P
- **Análise do Tempo de Resposta**
  - Tenta calcular o tempo de resposta no pior caso

Escalonamento com Garantia - Resumo

- Executivo Cíclico
- Prioridades + Teste de Escalonabilidade
  - Prioridades Variáveis com Teste de Escalonabilidade
  - Prioridades Fixas com Teste de Escalonabilidade
- Todos são capazes de garantir deadlines
  - Executivo cíclico é menos flexível porém oferece determinismo de escala
  - EDF oferece escalonabilidade superior em relação à prioridade fixa, porém é caótico em sobrecarga e análise é mais complexa
  - Prioridade fixa permite extender mais facilmente a análise de escalonabilidade