

## *Sistemas de Tempo-Real*

### Aula 2

#### Modelos computacionais

Modelos de tarefas com restrições temporais explícitas, implementação  
Controlo lógico e temporal (por eventos -ET e por tempo -TT)

### *Aula anterior (1)*

- Noção de **tempo real** e de **sistema de tempo real**
- Antagonismo **tempo real** vs **best effort**
- Objectivo do estudo dos STR – obter **garantias** de **comportamento temporal adequado**
- Aspectos a considerar: **tempo de execução**, de **resposta**, e **regularidade** de eventos periódicos
- Requisitos dos STR: **funcionais**, **temporais** e de **dependabilidade**
- Noção de **base de dados de tempo real**
- Restrições **soft**, **firm** e **hard**, e **hard real time** vs **soft real time**
- Importância de ter em conta o **cenário de pior caso**

## Modelos computacionais

### Modelo transformacional

- Segundo o qual um programa inicia e termina, transformando dados de entrada em resultados ou dados de saída.



### Modelo reactivo

- Segundo o qual um programa pode executar indefinidamente uma sequência de interacções, por exemplo operando sobre um fluxo de dados.



### Modelo de tempo-real

- **Modelo reactivo** em que o programa tem de se manter sincronizado com o fluxo de dados, o qual impõe restrições temporais à execução do programa.

Sistemas de Tempo-Real

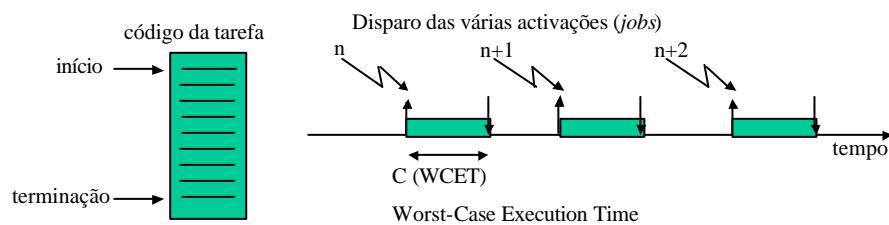
3

Luis Almeida, DETUA, Setembro de 2003

## Modelo de tempo-real

### Definição de tarefa (processo, actividade)

Sequência de activações (instâncias ou *jobs*), cada uma composta por um conjunto de instruções que, na ausência de outras actividades, é executada pelo CPU sem interrupção.



Sistemas de Tempo-Real

4

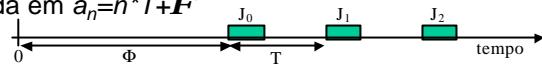
Luis Almeida, DETUA, Setembro de 2003

## Modelo de tempo-real

Quanto à periodicidade as tarefas podem ser

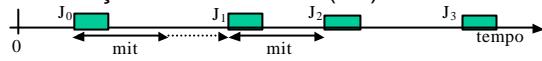
- periódicas

instância  $n$  activada em  $a_n = n * T + F$



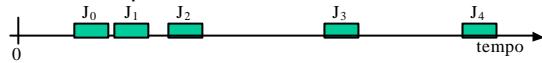
- esporádicas

tempo mínimo entre activações consecutivas (*mit*)



- aperiódicas

só se caracterizam de forma probabilística



Sistemas de Tempo-Real

5

Luis Almeida, DETUA, Setembro de 2003

## Modelo de tempo-real

Caracterização das tarefas

C – tempo máximo de execução (WCET)

T – período (periódica)

F – fase relativa = instante da 1ª activação (periódica)

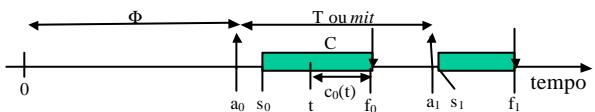
mit – *minimum interarrival time* (esporádica)

$a_n$  – instante de activação da  $n^{\text{a}}$  instância

$s_n$  – instante de início de execução da  $n^{\text{a}}$  instância

$f_n$  – instante de terminação da  $n^{\text{a}}$  instância

$c_n(t)$  – tempo máximo de execução residual da  $n^{\text{a}}$  instância no instante  $t$



Sistemas de Tempo-Real

6

Luis Almeida, DETUA, Setembro de 2003

## *Modelo de tempo-real*

Os requisitos das tarefas podem ser:

- **Temporais** – limites temporais aos **instantes de terminação** ou de geração de determinados eventos de saída.
- **Precedência** – estabelecem uma determinada **ordem de execução** entre tarefas.
- **Uso de recursos** – necessidade de utilização de **recursos partilhados** (e.g. portos de comunicação, um buffer em memória partilhada, variáveis globais, periféricos do sistema). Pode implicar uso de **operações atómicas** (cuja sequência não pode ser interrompida)

## *Modelo de tempo-real*

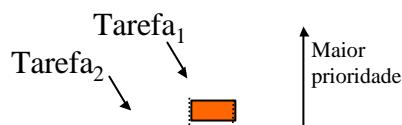
### Preempção

- Quando uma tarefa pode ser **interrompida temporariamente** para execução de outra **mais prioritária**, diz-se que admite **preempção**.
- Quando um sistema utiliza a propriedade de preempção das tarefas que executa diz-se **preemptivo**.
- Um conjunto de tarefas diz-se admitir **preempção total** quando todas as tarefas admitem preempção em qualquer ponto da sua execução (tarefas independentes)

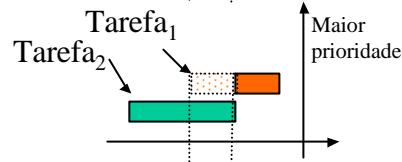
**Nota:** o acesso a **recursos partilhados** (tarefas com **dependências**) pode impor restrições sobre o grau de preempção que uma tarefa admite.

## *Modelo de tempo-real*

- Com preempção



- Sem preempção



Sistemas de Tempo-Real

9

Luis Almeida, DETUA, Setembro de 2003

## *Modelo de tempo-real*

Os requisitos temporais podem ser de vários tipos:

- **Deadline** – Limitação ao tempo máximo para terminação da tarefa.
- **Janela** – Delimitação máxima e mínima ao instante de terminação.
- **Sincronismo** – Limitação à diferença temporal entre a geração de dois eventos de saídas (existem outras formas).
- **Distância** – Limitação ao atraso (distância) entre a terminação, ou activação, de duas instâncias consecutivas  
(e.g., a mudança do óleo do motor num carro)

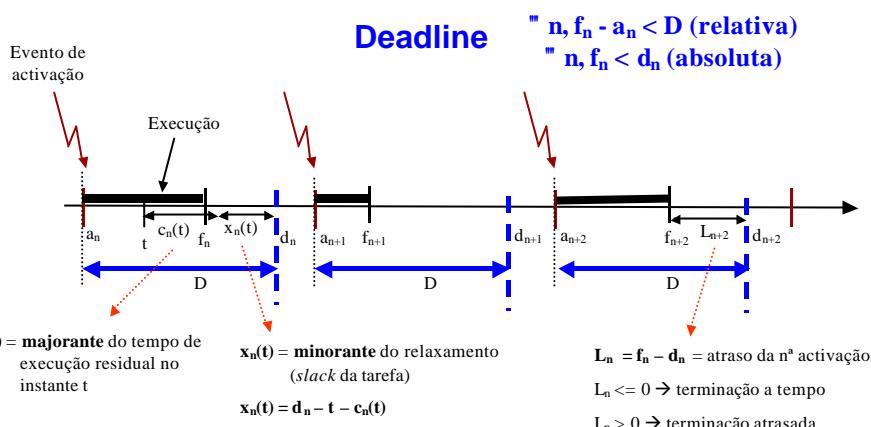
Tipo **deadline** é o mais comum!

Sistemas de Tempo-Real

10

Luis Almeida, DETUA, Setembro de 2003

## Modelo de tempo-real

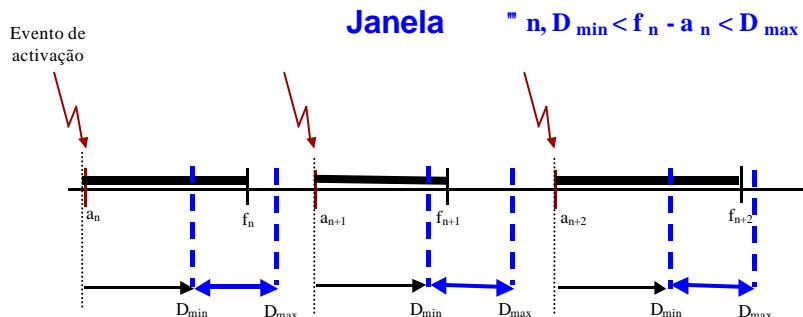


Sistemas de Tempo-Real

11

Luis Almeida, DETUA, Setembro de 2003

## Modelo de tempo-real

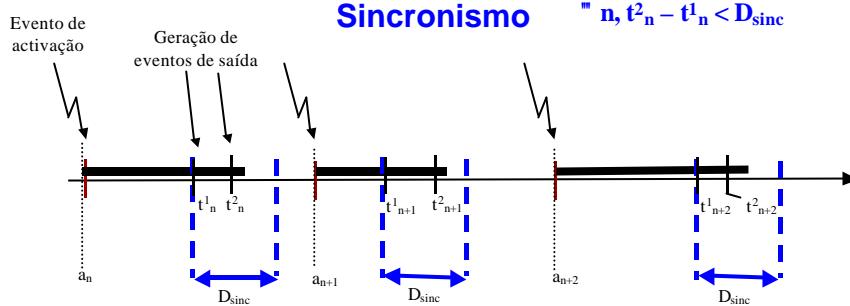


Sistemas de Tempo-Real

12

Luis Almeida, DETUA, Setembro de 2003

## Modelo de tempo-real

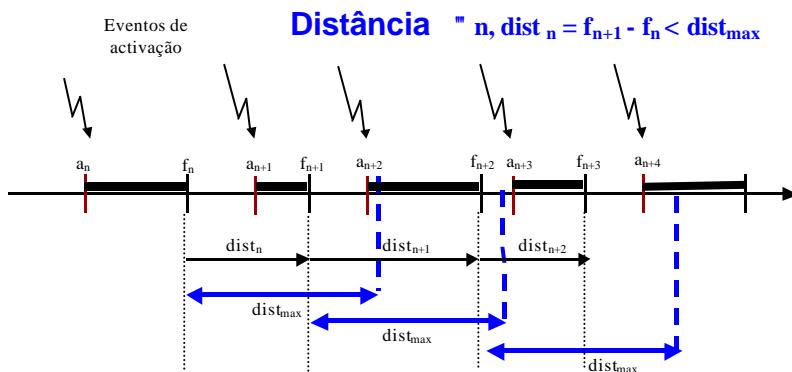


Sistemas de Tempo-Real

13

Luis Almeida, DETUA, Setembro de 2003

## Modelo de tempo-real



Sistemas de Tempo-Real

14

Luis Almeida, DETUA, Setembro de 2003

## *Modelo de tempo-real*

### Exemplo de caracterização de tarefas:

- **Periódicas:**  $\tau_i = \tau_i(C_i, \Phi_i, T_i, D_i)$

$$\tau_1 = \tau_1(2,5,10,10) \quad \tau_2 = \tau_2(3,10,20,20)$$

- **Esporádicas:** Semelhante às periódicas mas com  $mit_i$  em vez de  $T_i$  e  $\Phi_i$  não é habitualmente usado (poderia significar um tempo mínimo até à primeira activação).

$$\tau_i = \tau_i(C_i, mit_i, D_i)$$

$$\tau_1 = \tau_1(2,5,5) \quad \tau_2 = \tau_2(3,10,7)$$

## *Implementação de aplicações de tempo-real*

A **programação** de aplicações de tempo real quando envolve apenas:

- **um ciclo principal** e, eventualmente,
- um **número muito reduzido de actividades assíncronas** (que podem ser encapsuladas em rotinas de interrupção)

é normalmente efectuada de **forma directa sobre o CPU**, i.e., sem recurso a estruturas de SW intermédias tipo Sistema Operativo ou Executivo (*Kernel*).

## *Implementação de aplicações de tempo-real*

No caso de **programação directa** sobre o CPU, o **disparo** de actividades é normalmente feito por **interrupções**

- **Interrupções periódicas** (através de *timers*) para actividades periódicas. Estas interrupções são usadas para **contar tempo**.
- **Interrupções assíncronas** (comunicações, externas, etc.) para actividades disparadas por eventos (alterações do estado do sistema, e.g., disparo de um alarme, recepção de dados por um meio de comunicação, acção do operador)

## *Implementação de aplicações de tempo-real*

### Mas a utilização de interrupções:

- **Impõe um custo computacional** adicional necessário para a salvaguarda do estado do CPU no momento de cada interrupção (i.e., salvaguarda dos resgistros no *stack*).
- **Retira capacidade computacional** à execução do programa interrompido. Quanto mais interrupções surgirem mais devagar o programa executa pois está constantemente a ser interrompido. No limite, a execução do programa fica completamente bloqueada.

## *Implementação de aplicações de tempo-real*

A utilização de interrupções pode ser feita **com** ou **sem encadeamento (nesting)**

- **Com encadeamento** – é permitida a interrupção de rotinas de atendimento a interrupção (ISRs) por interrupções de maior prioridade.
  - Maior dificuldade de dimensionamento do *stack*
  - Melhor resposta temporal das ISRs de maior prioridade
- **Sem encadeamento** – cada ISR executa até final sem interrupção. Outras interrupções pendentes são atrasadas.
  - Características opostas do caso anterior
  - Notar o bloqueio das ISRs de maior prioridade pelas de menor.

## *Implementação de aplicações de tempo-real*

Por outro lado, quando a aplicação envolve **múltiplas actividades**, assíncronas ou não:

a respectiva programação é facilitada pela utilização de **Sistemas Operativos** ou **Executivos multi-tarefa (muti-tasking)** os quais suportam directamente múltiplas tarefas que podem executar de forma independente, ou partilhando recursos do sistema,

cada **actividade** é encapsulada numa **tarefa**.

## *Executivos Multi-Tarefa*

A **programação** de aplicações com recurso a estruturas de SW tipo **Sistema Operativo** ou **Executivo** permite:

- **Maior nível de abstracção**
- **Menor dependência relativamente ao HW**
- **Maior facilidade de manutenção do SW**

**Nota:** Mas mesmo nestes casos, o disparo das tarefas é feito por interrupções. Há uma interrupção periódica que fornece uma medida de tempo ao SO ou Executivo e é possível usar interrupções assíncronas embora, normalmente, estejam encapsuladas em *device drivers*.

## *Executivos Multi-Tarefa*

O **processamento** associado a uma dada actividade pode ser efectuado:

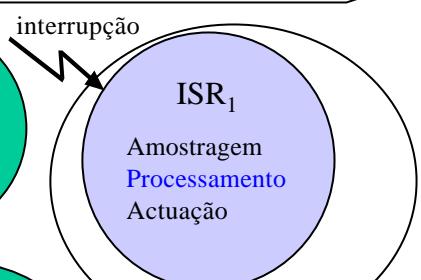
- **Ao nível de uma ISR (*Interrupt Service Routine*)**
  - Não se tira partido de algumas vantagens do SO ou Executivo (programação de baixo nível – muito dependente do HW)
  - Elevada reactividade a eventos externos (micro-segundos...)
  - Grande interferência ao nível das tarefas
  - N<sup>º</sup> limitado de ISRs
- **Ao nível de uma tarefa**
  - Tira partido das vantagens do SO ou Executivo (programação de alto nível, menor dependência do HW, melhor manutenção)
  - Menor reactividade a eventos externos (maior *overhead*)
  - ISRs reduzidas para menor perturbação sobre as tarefas

## Executivos Multi-Tarefa

### Processamento:

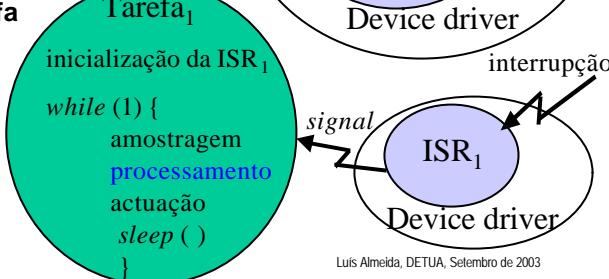
- Ao nível de uma ISR  
(não standard)

Tarefa<sub>1</sub>  
inicialização da ISR<sub>1</sub>  
*while (1) { }*



- Ao nível de uma tarefa  
(standard)

Tarefa<sub>1</sub>  
inicialização da ISR<sub>1</sub>  
*while (1) {*  
amostragem  
processamento  
actuação  
*sleep ( )*  
*}*



Sistemas de Tempo-Real

Luis Almeida, DETUA, Setembro de 2003

## Executivos Multi-Tarefa

### Classificação dos SOs e Executivos relativamente às garantias temporais

#### • Não Tempo-Real (*time-sharing*)

- (e.g., Unix, Linux, Windows NT) (seguem modelo transformacional)  
Não é possível majorar o tempo de resposta a um evento (e.g., devido a *swapping*, bloqueio no acesso a periféricos, escalonamento que favorece a distribuição equitativa do CPU)

#### • Soft Real-Time

- (e.g. OS9) Usam técnicas de tempo-real (exclusão de memória virtual, mecanismos de IPC rápidos e com bloqueios reduzidos, chamadas ao sistema curtas) mas não oferecem garantias temporais (tipo *best-effort*)

#### • Hard Real-Time

- (e.g. SHaRK, RTLinux, QNX) Oferecem garantias temporais

Sistemas de Tempo-Real

24

Luis Almeida, DETUA, Setembro de 2003

## *Controlo lógico e controlo temporal*

### Controlo lógico

Controlo do fluxo de programa, i.e., sequência efectiva das operações a ser executadas (e.g., descrito através de um fluxograma) – **fundamental para se determinar C (WCET)**

### Controlo temporal

Controlo dos **instantes de execução** das operações do programa (e.g., disparo de actividades, verificação do cumprimento de restrições temporais,...)

## *Controlo temporal*

### Disparo de actividades (funções)

#### **Por tempo (*time-triggered*)**

A execução de actividades (funções) é disparada por intermédio de um sinal de controlo baseado na progressão do tempo (e.g., através de uma interrupção periódica).

#### **Por eventos (*event-triggered*)**

A execução de actividades (funções) é disparada por intermédio de um sinal de controlo assíncrono baseado na alteração do estado do sistema (e.g., através de uma interrupção externa).

## *Controlo temporal*

### Sistemas disparados por tempo

#### *time-triggered (TT) systems*

- Típicos em aplicações de controlo (amostragem de variáveis contínuas).
- Existe uma referência temporal comum (permite estabelecer uma relação de fase)
- Taxa de utilização do CPU constante mesmo quando não há variações no estado do sistema.
  - Situação de **pior caso bem definida**

## *Controlo temporal*

### Sistemas disparados por eventos

#### *event-triggered (ET) systems*

- Típicos na monitorização de condições esporádicas no estado do sistema (e.g., verificação de alarmes ou de solicitações assíncronas).
- Taxa de utilização do sistema computacional (e.g. CPU) variável consoante a frequência de ocorrência de eventos.
  - Situação de **pior caso mal definida**  
ou se utilizam *argumentos probabilísticos*  
ou se impõe uma *limitação à máxima taxa de eventos*

## *Controlo temporal*

Exemplo, considere os seguintes conjuntos de tarefas e calcule o atraso máximo que cada tarefa pode sofrer

- TT  $\{\tau_i = \tau_i \ (C_i=1, \Phi_i=i, T_i=5, D_i=T_i \ i=1..5)\}$
- ET  $\{\tau_i = \tau_i \ (C_i=1, (\Phi_i=0), m_{it_i}=5, D_i=m_{it_i} \ i=1..5)\}$

Determine também a taxa média e máxima de **utilização de CPU**

**\*\* utilização =  $\sum_{i=1..N}(\text{tempo de execução}/\text{período de activação})$  \*\***  
para ambos os casos, considerando no caso médio que as tarefas ET são activadas em média de 100 em 100 unidades de tempo.

## *Resumo da Aula 2*

- Modelos computacionais (**modelo de tempo-real**)
- Tarefas de tempo-real: periódicas, esporádicas e aperiódicas
- Restrições temporais do tipo **deadline**, janela, sincronismo e distância
- Implementação de tarefas e utilização de um **kernel multitasking**
- **Controlo lógico** e **controlo temporal**
- Tarefas **event-triggered** e **time-triggered**

## Aula prática 2

### Utilização do kernel ReTMiK para a plataforma Kit188

- Utilização de múltiplas tarefas com períodos de activação diversos
- Utilização da propriedade de reentrância de código
- Utilização da tarefa de *background*

## Trabalho para a Aula 3

### Tema:

#### Porquê e como usar um sistema operativo ou *kernel* de tempo-real?

- Textos base em  
<http://www.smxinfo.com/articles/articles.htm>
  - How to Use a Real-Time Multitasking Kernel
  - Why use a commercial Kernel

Preparar apresentação (definir grupo) e discussão (todos)