

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

## **Aula 2**

# **Modelos computacionais**

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

Adaptado dos slides desenvolvidos pelo Prof. Doutor Luís Almeida para a disciplina “Sistemas de Tempo-Real”

# Aula anterior (1)

- Relembrando ...
  - Noção de **tempo real** e de **sistema de tempo real**
  - Antagonismo **tempo real** vs *best effort*
  - Objetivo 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

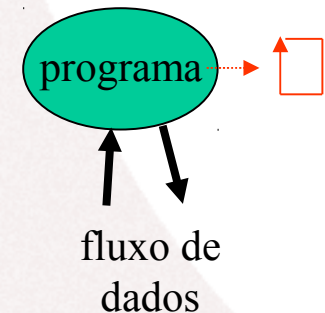
Segundo o qual um programa inicia e termina, transformando dados de entrada em resultados ou dados de saída.

## Modelo reativo

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

## Modelo de tempo-real

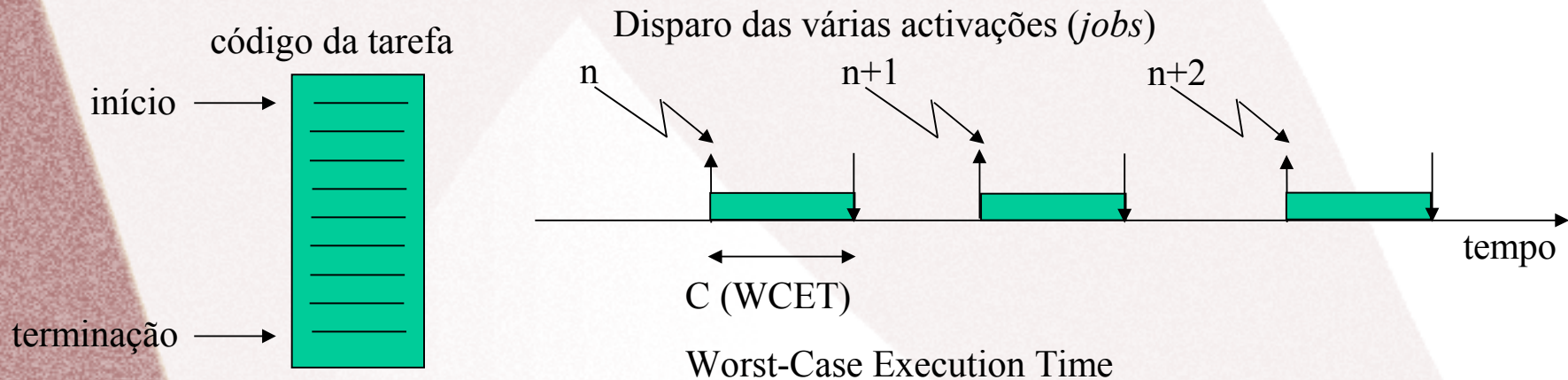
**Modelo reativo** em que o programa tem de se manter sincronizado com o fluxo de dados, o qual impõe restrições temporais à execução do programa.



# Modelo de tempo-real

## Definição de tarefa (processo, atividade)

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



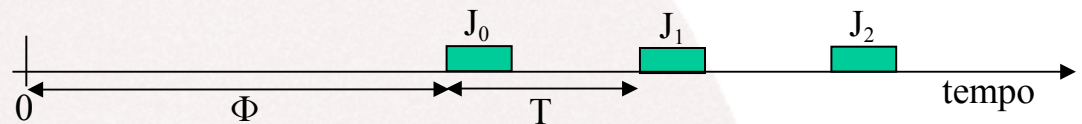


# Modelo de tempo-real

Quanto à periodicidade as tarefas podem ser

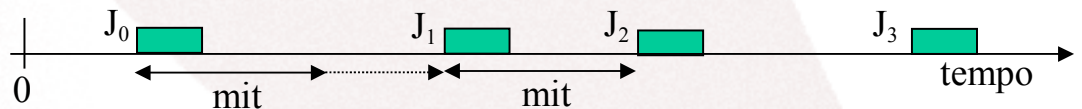
- **periódicas**

instância  $n$  ativada em  $a_n = n * T + \Phi$



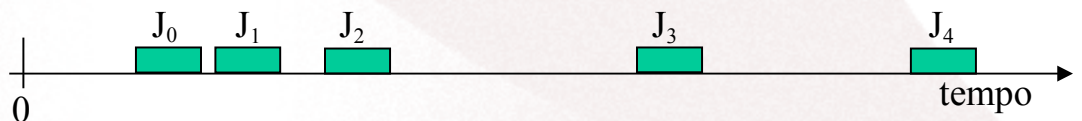
- **esporádicas**

tempo mínimo entre ativações consecutivas (*mit*)



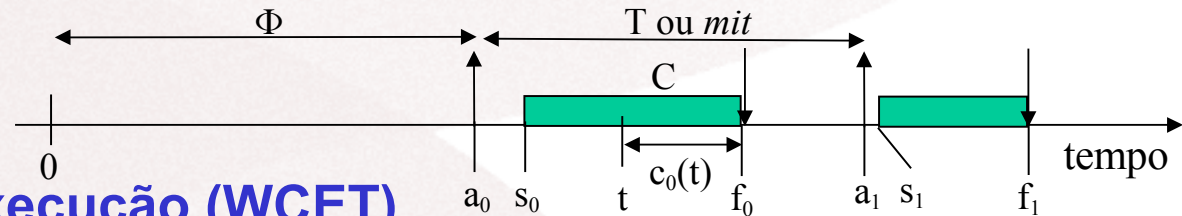
- **aperiódicas**

só se caracterizam de forma probabilística



# Modelo de tempo-real

## Caracterização das tarefas



- **C** – tempo máximo de execução (WCET)
- **T** – período (periódica)
- **$\Phi$**  – fase relativa = instante da 1ª ativação (periódica)
- **mit** – *minimum interarrival time* (esporádica)
- **$a_n$**  – instante de ativação da  $n^a$  instância
- **$s_n$**  – instante de início de execução da  $n^a$  instância
- **$f_n$**  – instante de terminação da  $n^a$  instância
- **$c_n(t)$**  – tempo máximo de execução residual da  $n^a$  instância no instante  $t$



# 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

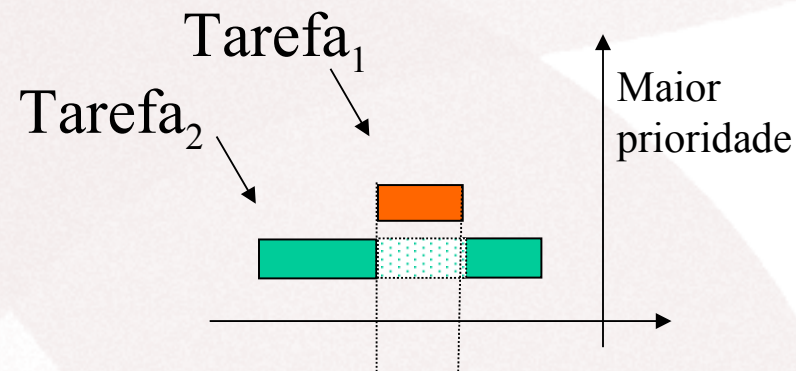
## Preempção

- 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.

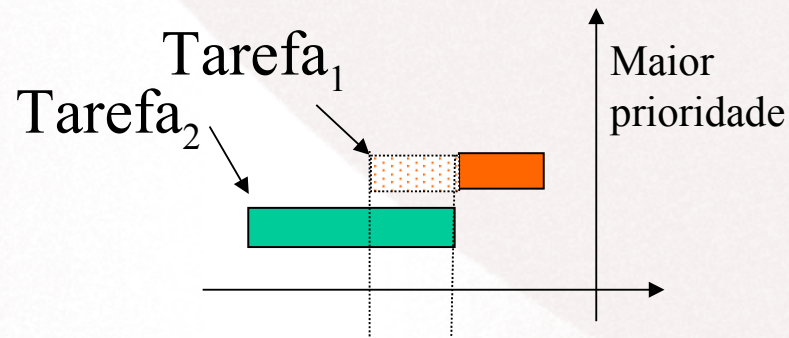


# Modelo de tempo-real

**Com preempção**



**Sem preempção**



# 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 saída (existem outras formas).
- **Distância** – Limitação ao atraso (distância) entre a terminação, ou ativação, de duas instâncias consecutivas (e.g., a mudança do óleo do motor num carro)
- Tipo **deadline** é o mais comum!

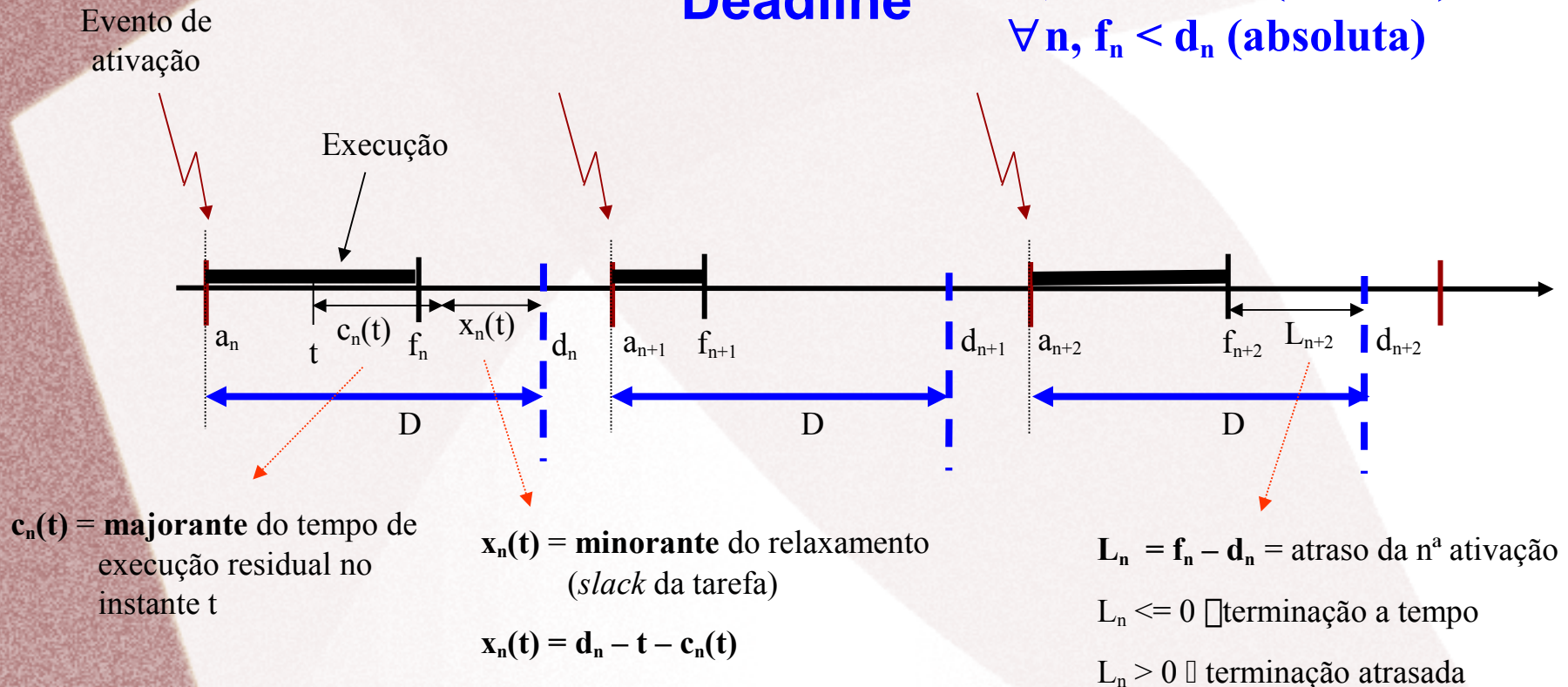


# Modelo de tempo-real

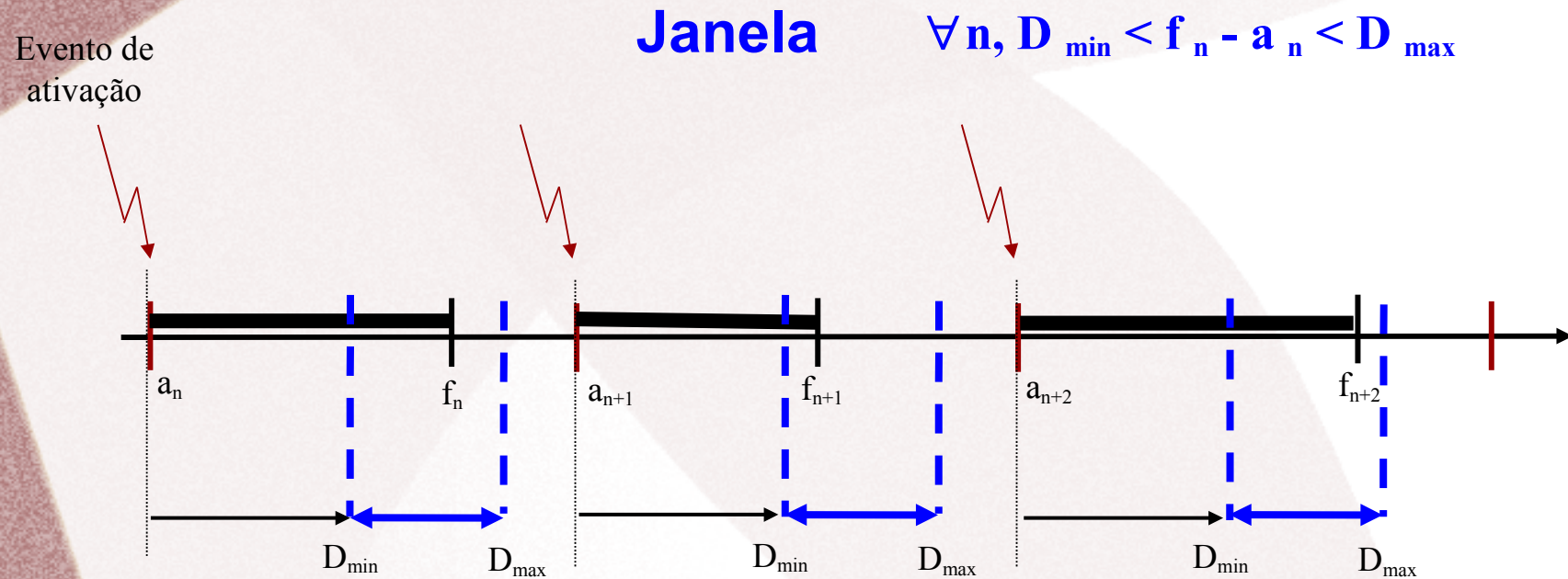
## Deadline

$$\forall n, f_n - a_n < D \text{ (relativa)}$$

$$\forall n, f_n < d_n \text{ (absoluta)}$$

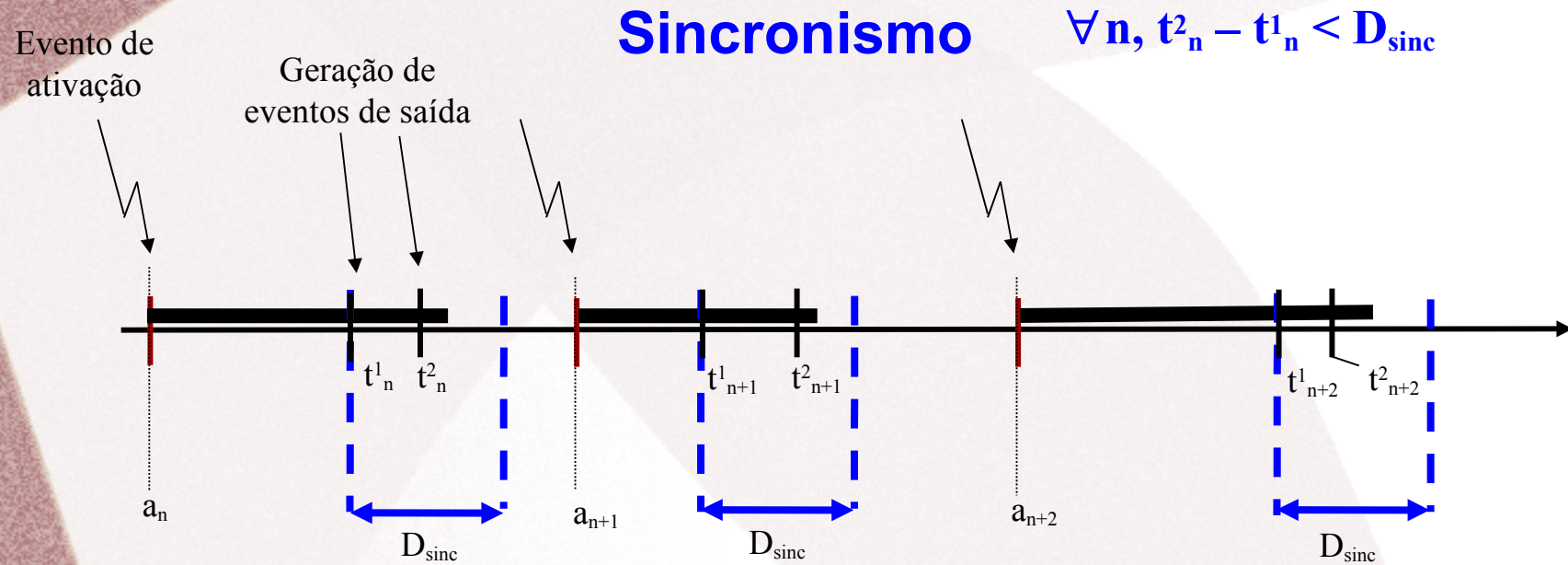


# Modelo de tempo-real

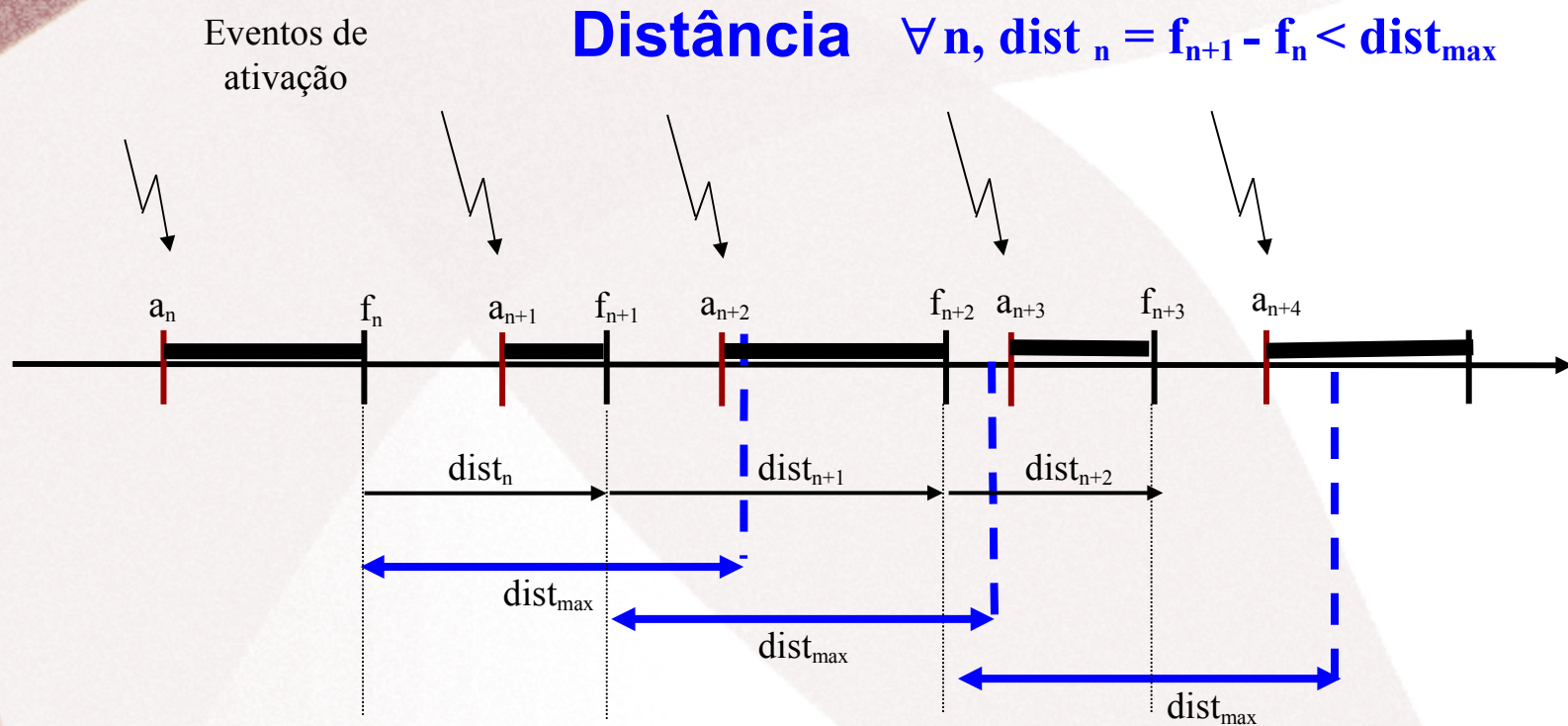




# Modelo de tempo-real



# Modelo de tempo-real





# 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 ativação).

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

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

# ***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 atividades assíncronas** (que podem ser encapsuladas em rotinas de interrupção)

é normalmente efetuada de **forma direta 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 direta sobre o CPU, o disparo de atividades é normalmente feito por interrupções

- **Interrupções periódicas** (através de *timers*) para atividades periódicas. Estas interrupções são usadas para contar tempo.
- **Interrupções assíncronas** (comunicações, externas, etc.) para atividades disparadas por eventos (alterações do estado do sistema, e.g., disparo de um alarme, receção de dados por um meio de comunicação, ação 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 registos na *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 atividades, assíncronas ou não:

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

Cada **atividade** 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 abstração**
- **Menor dependência relativamente ao HW**
- **Maior facilidade de manutenção do SW**

## **Notas:**

- Mesmo nestes casos, o **disparo das tarefas é feito por interrupções**, via interrupção periódica que fornece uma medida de tempo ao SO ou Executivo (tick).
- É também possível usar **interrupções assíncronas** embora, normalmente, estejam **encapsuladas em device drivers**.



# Executivos Multi-Tarefa

○ **processamento** associado a uma dada atividade pode ser efetuado:

- **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 reatividade a eventos externos (micro-segundos...)
- Grande interferência ao nível das tarefas
- N° limitado de ISRs

- **Ao nível de uma tarefa**

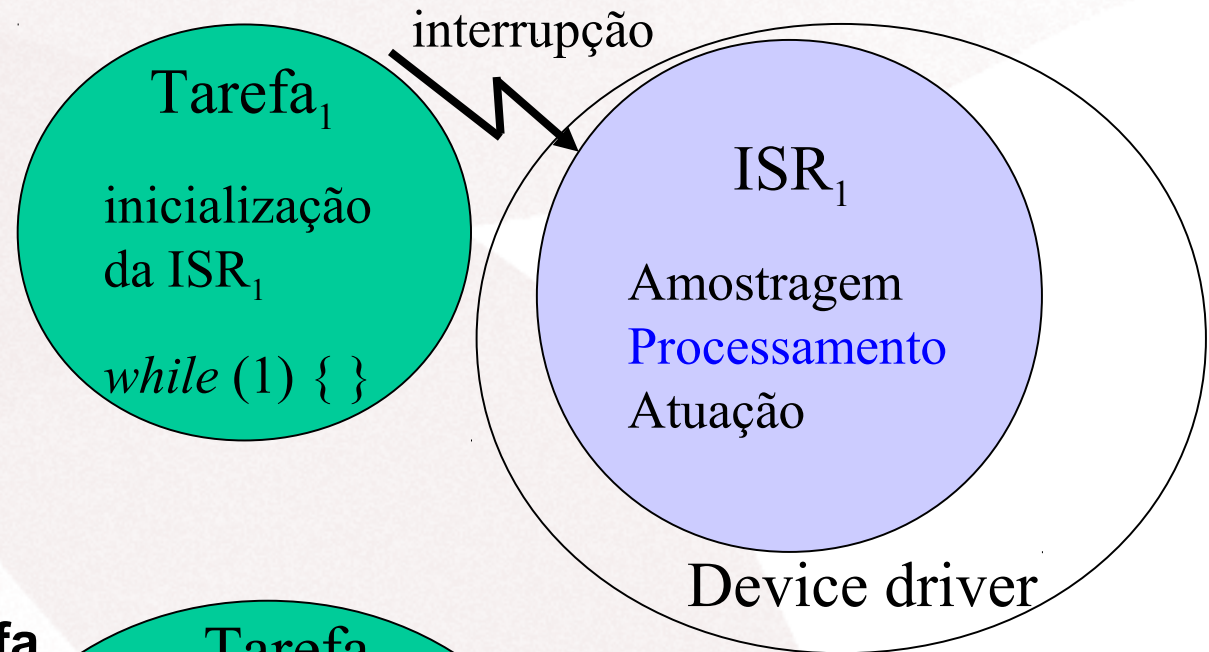
- Tira partido das vantagens do SO ou Executivo (programação de alto nível, menor dependência do HW, melhor manutenção)
- Menor reatividade a eventos externos (maior *overhead*)
- ISRs reduzidas para menor perturbação sobre as tarefas



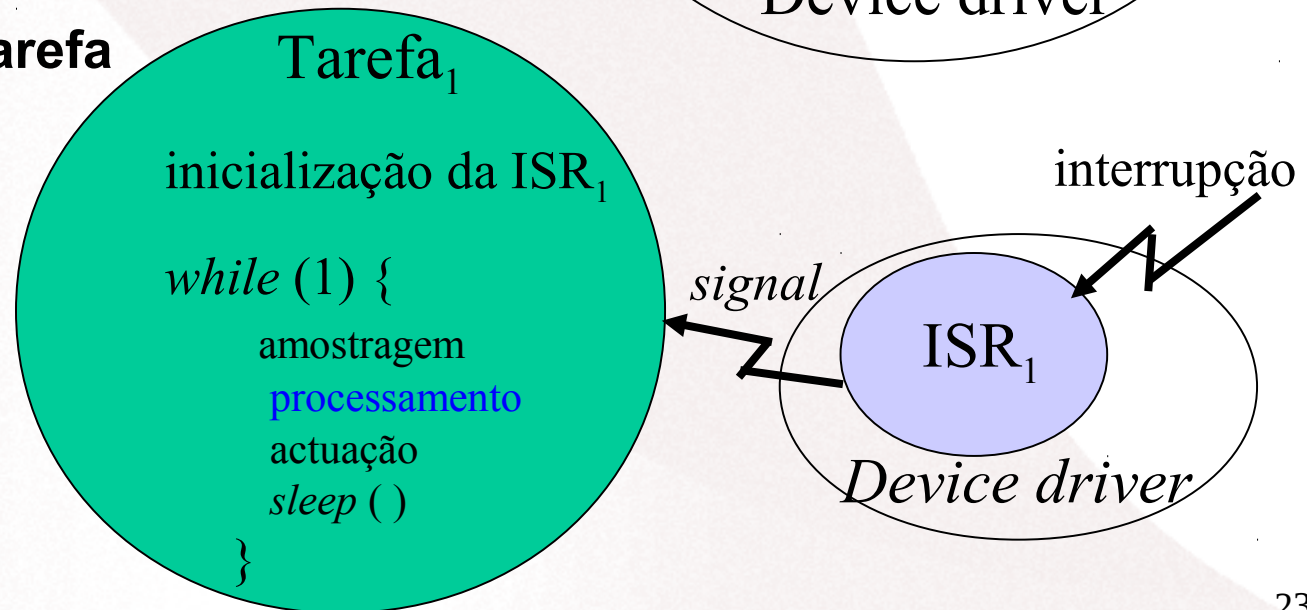
# Executivos Multi-Tarefa

## Processamento:

- Ao nível de uma ISR  
(não standard)



- Ao nível de uma tarefa  
(standard)



# 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, alguns serviços do Linux/Windows)

- 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, VxWorks, ...)

- Oferecem garantias temporais



# Controlo lógico e controlo temporal

- Controlo lógico

- Controlo do fluxo de programa, i.e., sequência efetiva 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 atividades, verificação do cumprimento de restrições temporais,...)

# Controlo temporal

## Disparo de atividades (funções)

- **Por tempo (*time-triggered*)**
  - A execução de atividades (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 atividades (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  $\{\tau_i = \tau_i (C_i=1, (\Phi_i=0), \text{mit}_i=5, D_i=\text{mit}_i \ i=1..5)\}$
  - Determine também a taxa média e máxima de **utilização de CPU** para ambos os casos, considerando no caso médio que as tarefas ET são ativadas em média de 100 em 100 unidades de tempo.

Nota: a taxa de utilização de CPU é dada por:

$$U = \sum_{i=1}^N \frac{C_i}{T_i}$$

# 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**