Sistemas Operacionais

Gerencia de Tarefas

Índice

- Tarefas e programas
- Tipos de Tarefas
- Sistemas multitarefa
- Monitores
- Preempção
- Contexto
  - TCB

Tarefa

Tarefa

- Uma tarefa é definida como a execução de um fluxo sequencial de instruções, construído para atender uma finalidade para a qual foi programada.

  – A realização de um cálculo complexo
  – A edição de um gráfico
  – A formatação de um disco
  – …

Tarefa vs. Programa

- Programa
  - Conjunto de uma ou mais sequências de instruções
  - Estático (não muda de estado)
- Tarefa
  - É a execução, pelo processador, das sequências de instruções definidas em um programa para realizar seu objetivo
  - Dinâmico (um estado a cada instante)

Tipos de Tarefas (temporal)

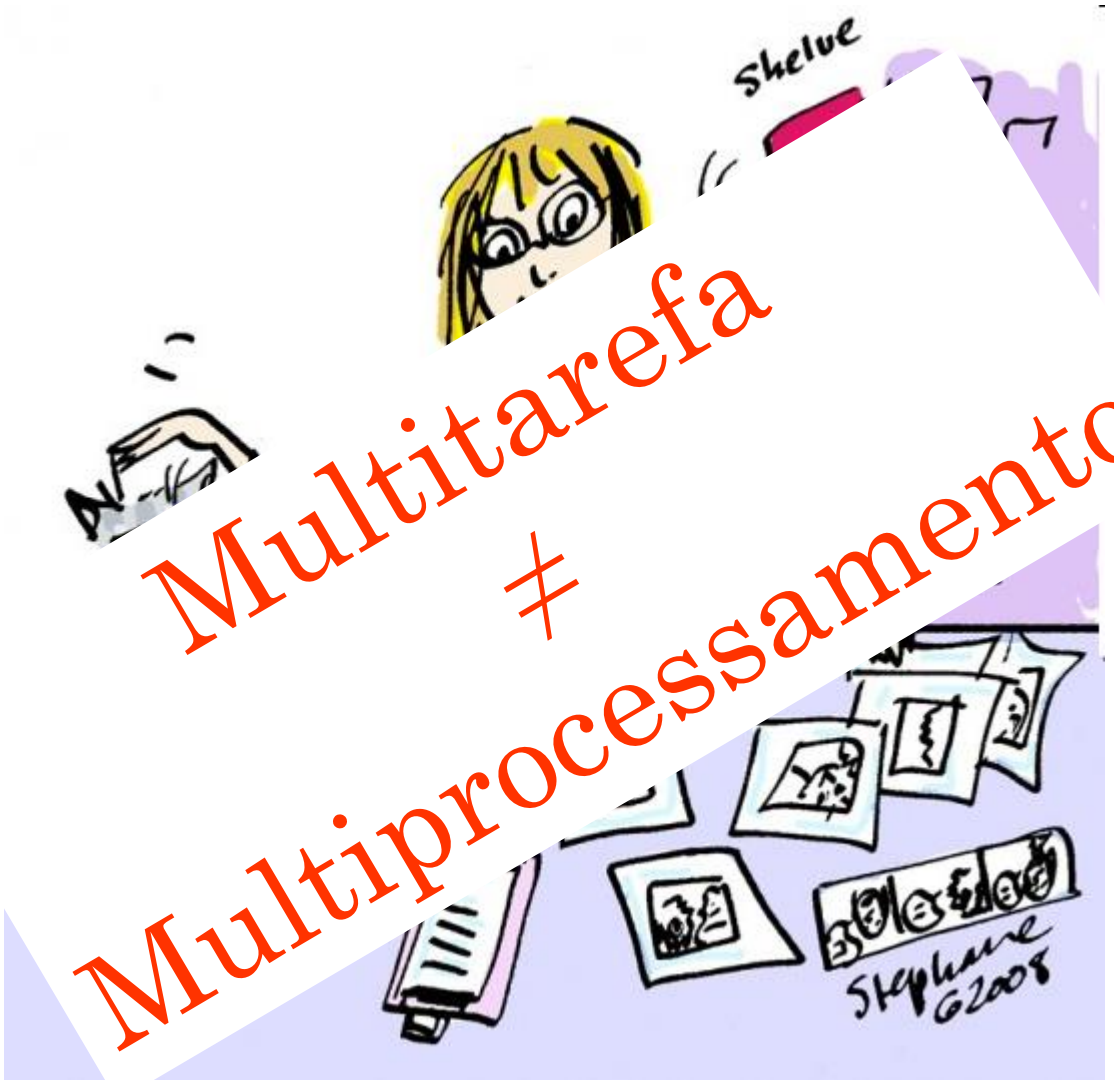
- Tarefas de tempo real
  - Previsibilidade em seus tempos de respostas (controle de freio ABS, reprodução de áudio)
- Tarefas interativas
  - Eventos externos solicitados por usuários (editores de texto, navegadores Internet, jogos)
- Tarefas em lote (batch)
  - Sem intervenção do usuário (procedimentos de backup, anti-vírus)

Tipos de Tarefas (processador)

- Tarefas orientadas a entrada/saída (I/O-bound tasks)
  - Se uma tarefa passa a maior parte do tempo esperando por dispositivos de E/S, diz-se que é limitada por E/S
- Tarefas orientadas a processamento (CPU-bound tasks)
  - Se, ao contrário, a tarefa gasta a maior parte do seu tempo usando a CPU ela é dita limitada por processador

Multitarefa & Concorrência

Multitarefa & Concorrência



Analogia: Receita de torta

Analogia: Receita de torta

Assim como uma receita de torta pode definir várias tarefas inter-dependentes para elaborar a torta:

- preparar a massa, fazer o recheio, decorar, etc.

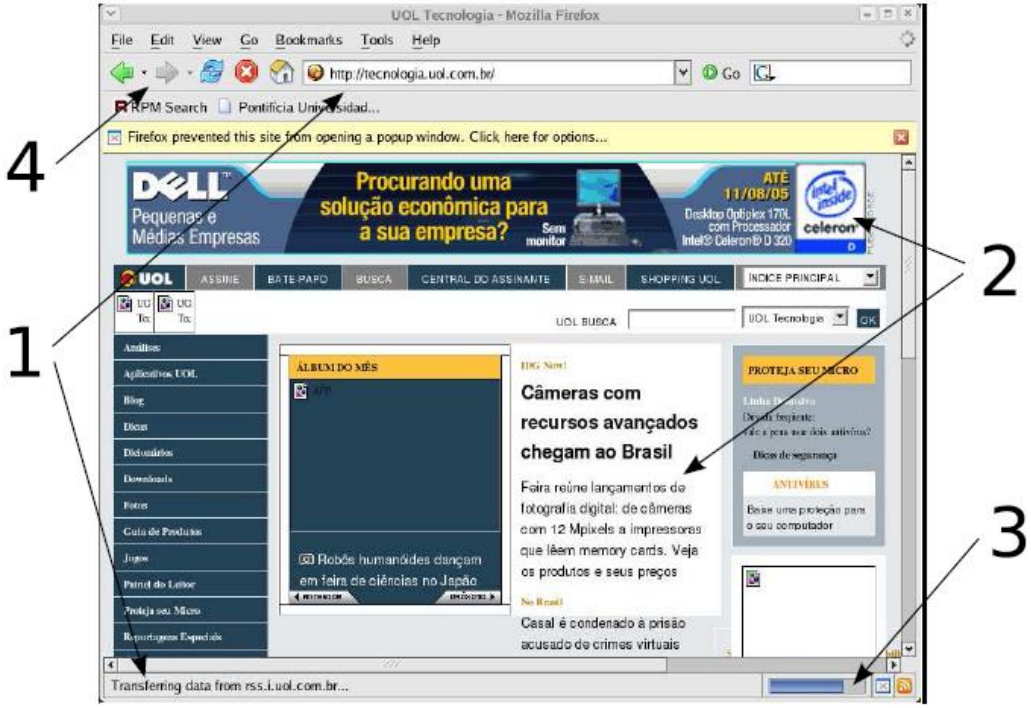
Analogia: Receita de torta

Assim como uma receita de torta pode definir várias tarefas inter-dependentes para elaborar a torta:

- preparar a massa, fazer o recheio, decorar, etc.

Um programa também pode definir várias sequências de execução inter-dependentes para atingir seus objetivos

Ex.: Tarefas de um navegador



1. Buscar via rede os vários elementos que compõem a página Web
2. Receber, analisar e renderizar o código HTML e os gráficos recebidos
3. Animar os diferentes elementos que compõem a interface do navegador
4. Receber e tratar os eventos do usuário (*clicks*) nos botões do navegador

Cenários...

- Um usuário pode realizar diversas atividades
  - Música
  - Editoração de texto
  - Navegar na Internet

Cenários...

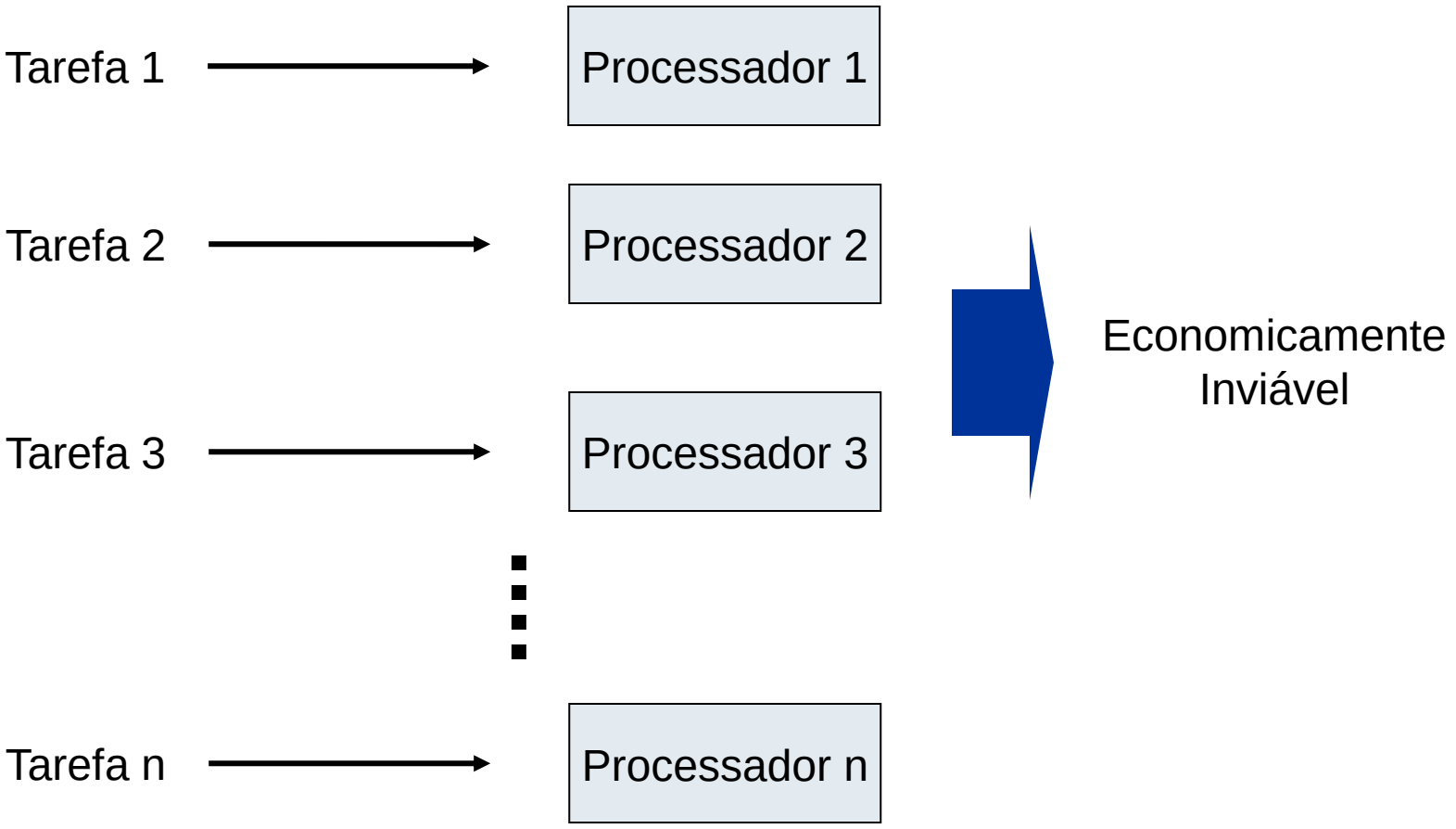
- Um usuário pode realizar diversas atividades
  - Música
  - Editoração de texto
  - Navegar na Internet
- Servidor pode ter vários usuários conectados
  - Acesso remoto
  - Servidor de e-mails

Cenário...

- Mesmo num computador executando um único programa o SO deve dar suporte a:
  - Gerenciamento de memória
  - Dispositivos de E/S (video, teclado, mouse)
  - Outras atividades programadas

Tudo isso deve ser executado como se fosse "ao mesmo tempo"

Como processar as tarefas?



Como processar as tarefas?

- Geralmente, há muito mais tarefas a realizar que processadores disponíveis
- As tarefas possuem diferentes importância, duração e comportamento

Como processar as tarefas?

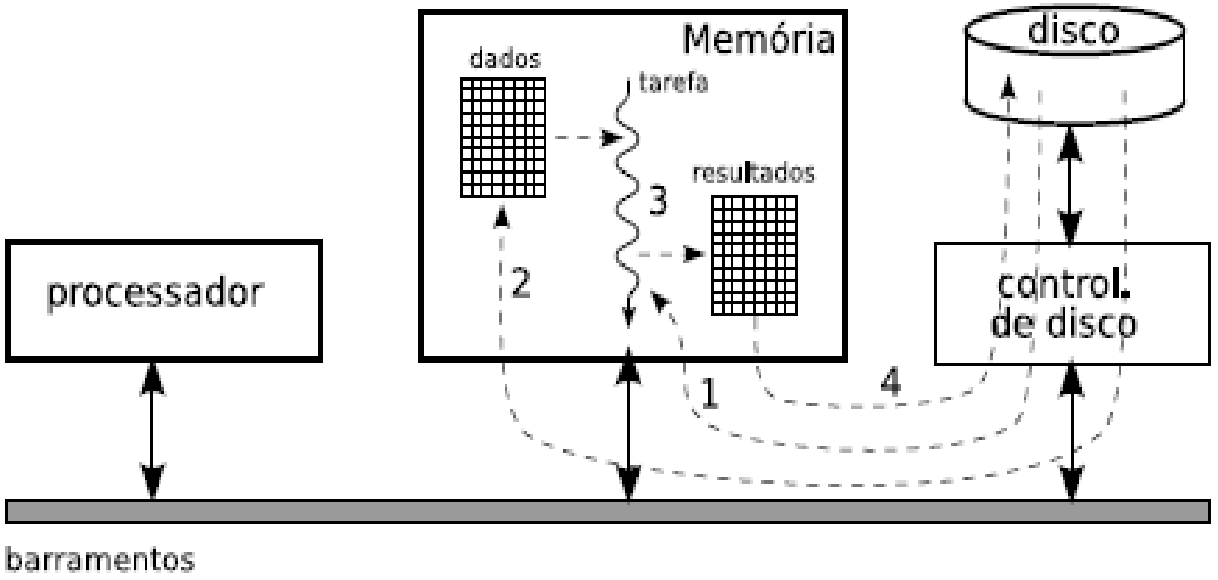
- Geralmente, há muito mais tarefas a realizar que processadores disponíveis

- As tarefas possuem diferentes importância, duração e comportamento

$\Rightarrow$ A **gerência** de tarefas tem uma grande importância dentro de um sistema operacional multitarefa

Histórico

Sistemas mono-tarefa



1. Carga do código na memória
2. Carga dos dados na memória
3. Processamento - consumindo dados e produzindo resultados
4. Término da execução - descarga dos resultados no disco

Programas Monitores

Surgiram com a evolução do HW

⇒ gerenciar uma fila de programas a executar, mantida no disco

Repetir

carregar um programa do disco para a memória
carregar os dados de entrada do disco para a memória
transferir a execução para o programa recém carregado
aguardar o término da execução do programa
escrever os resultados gerados pelo programa no disco

Até processar todos os programas da fila

⇒ **Precursor do SO**

Histórico

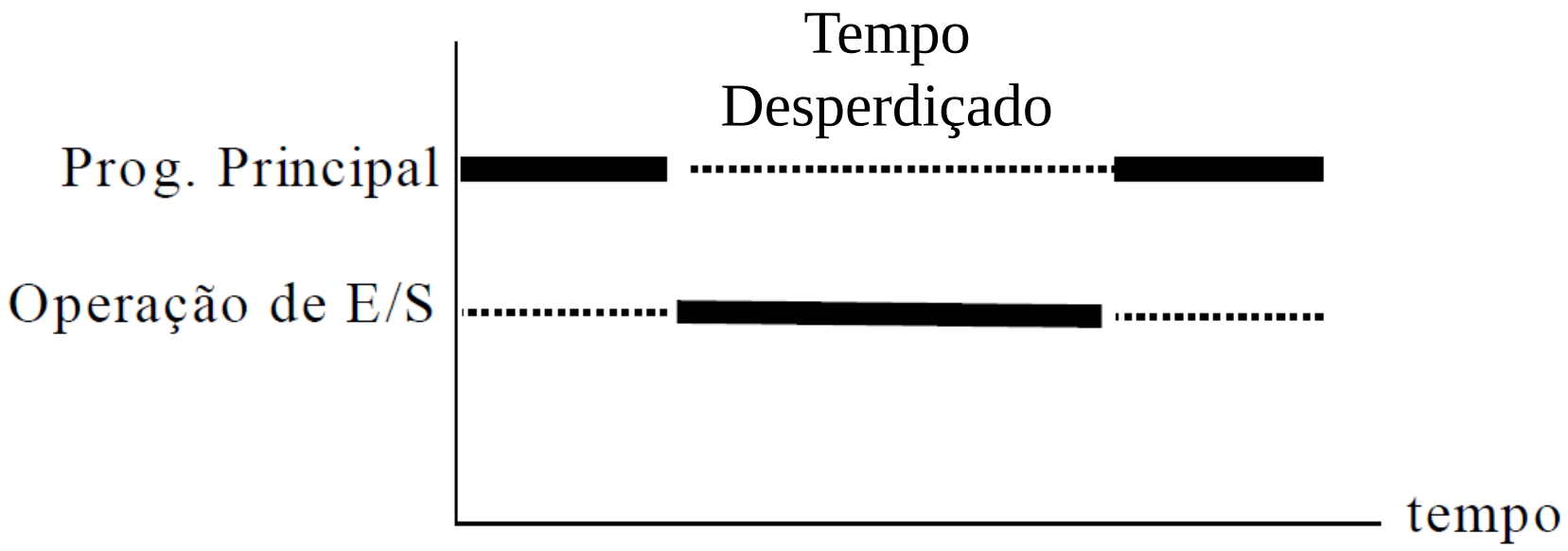
Diagrama de estados – sistemas mono-tarefa

Histórico

Diagrama de estados – sistemas mono-tarefa

- **Ociosidade do processador**
  - **Espera de recursos mais lentos como leitura em disco**

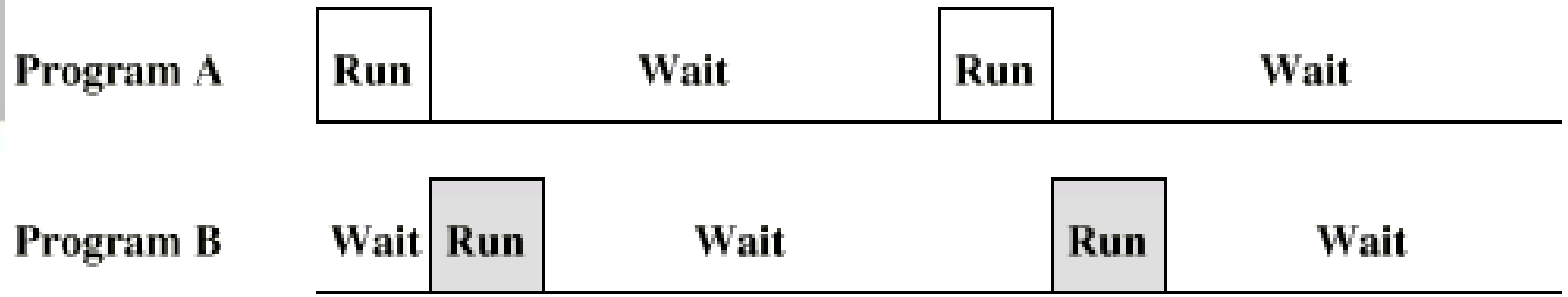
Ociosidade da CPU



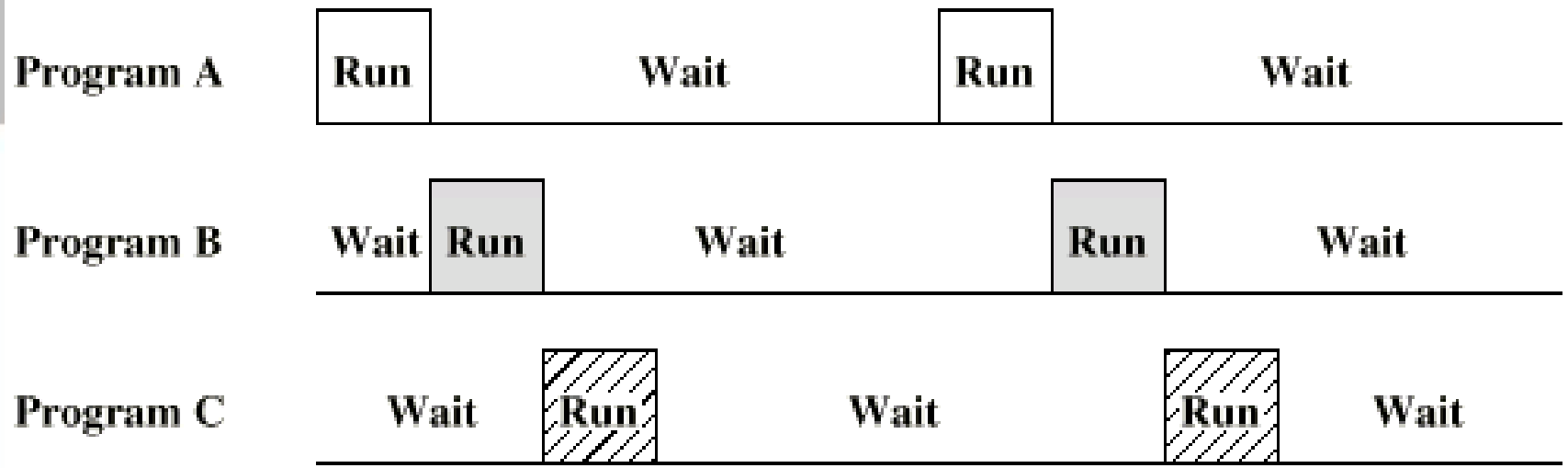
Ex.: Acesso à memória → 10 ns; acesso a disco rígido → 10 ms
⇒ Um milhão de vezes mais lento!

Sistemas multi-tarefa

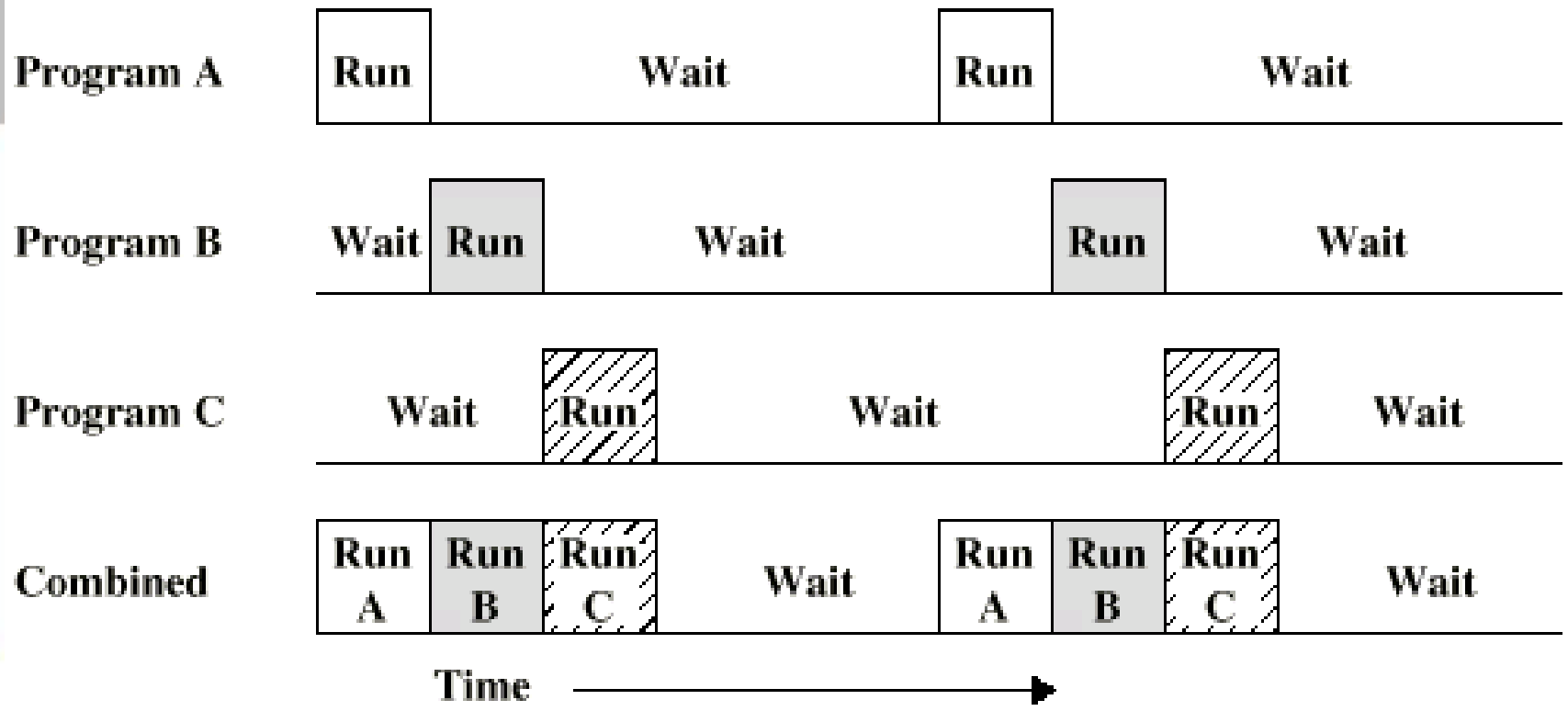
Sistemas multi-tarefa



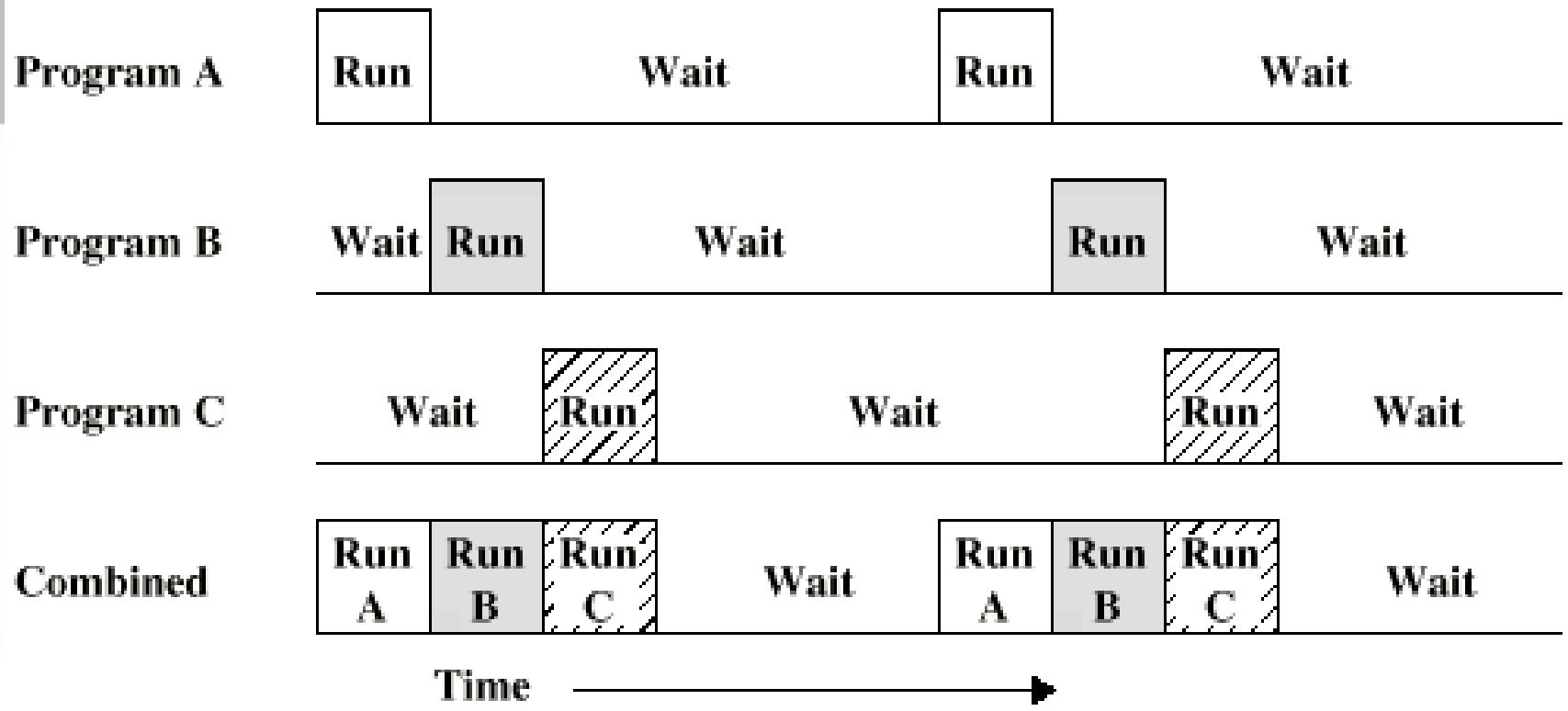
Sistemas multi-tarefa



Sistemas multi-tarefa

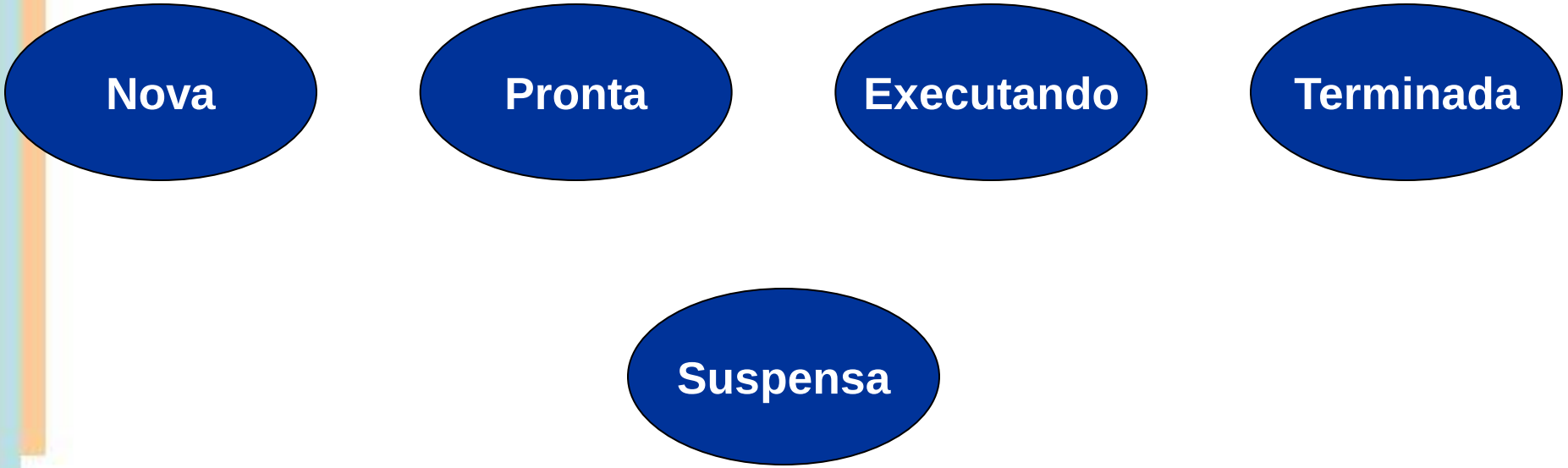


Sistemas multi-tarefa

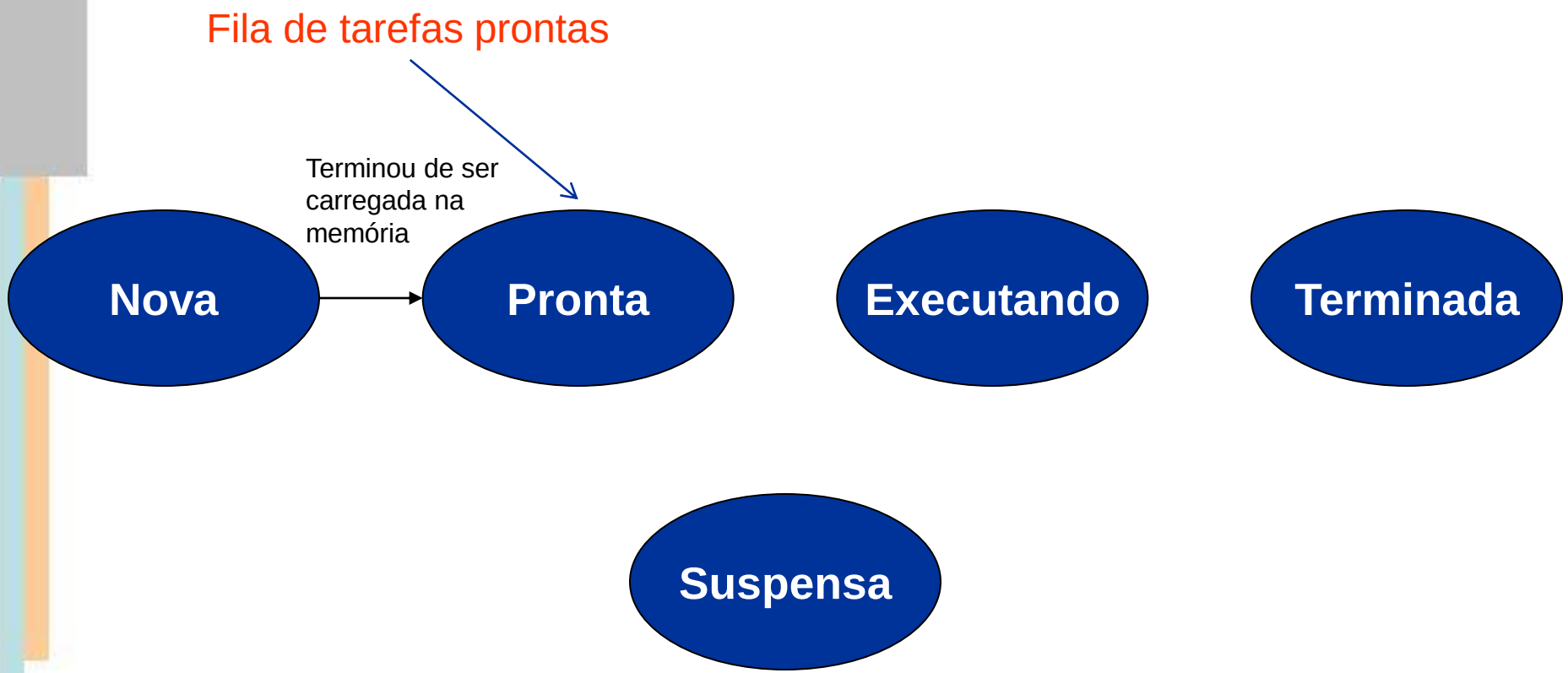


* O tempo gasto entre as trocas das tarefas NÃO está ilustrado

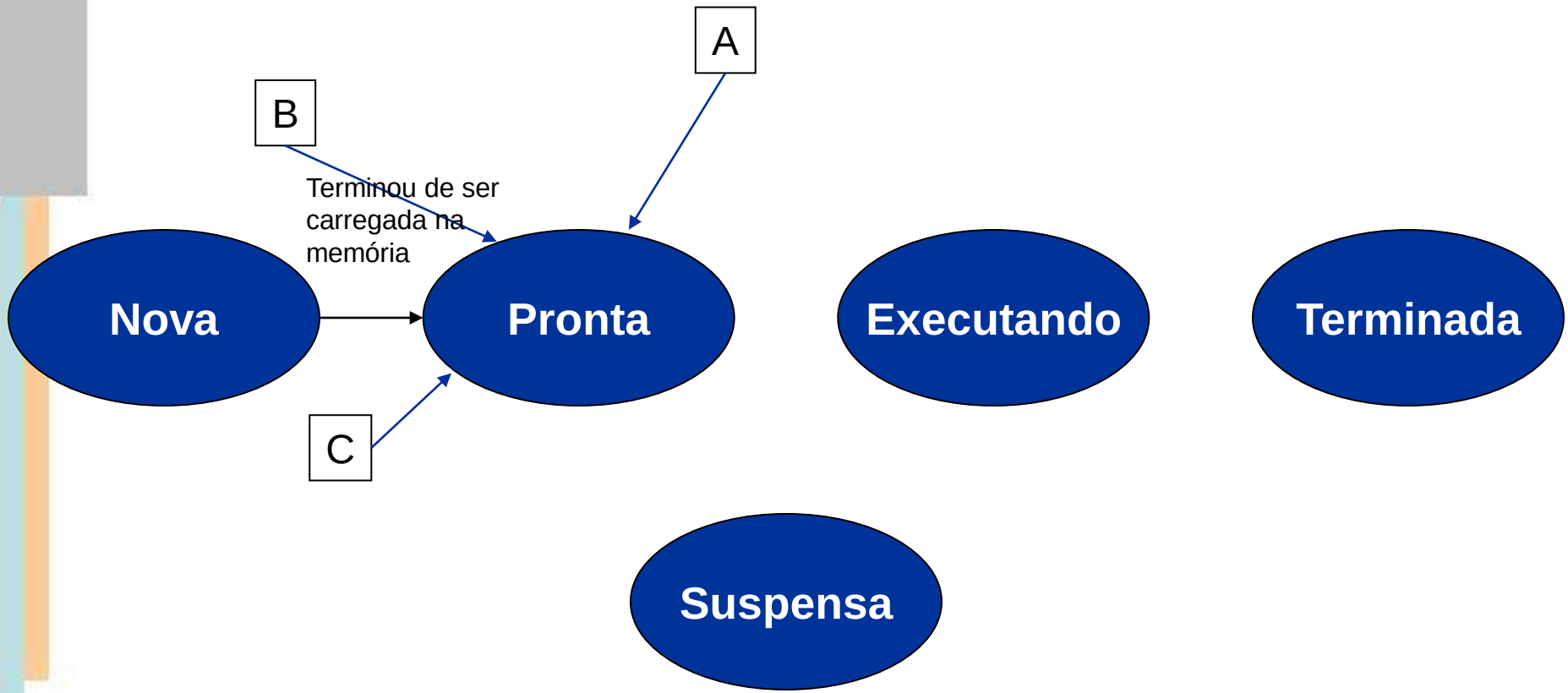
Sistemas multi-tarefa



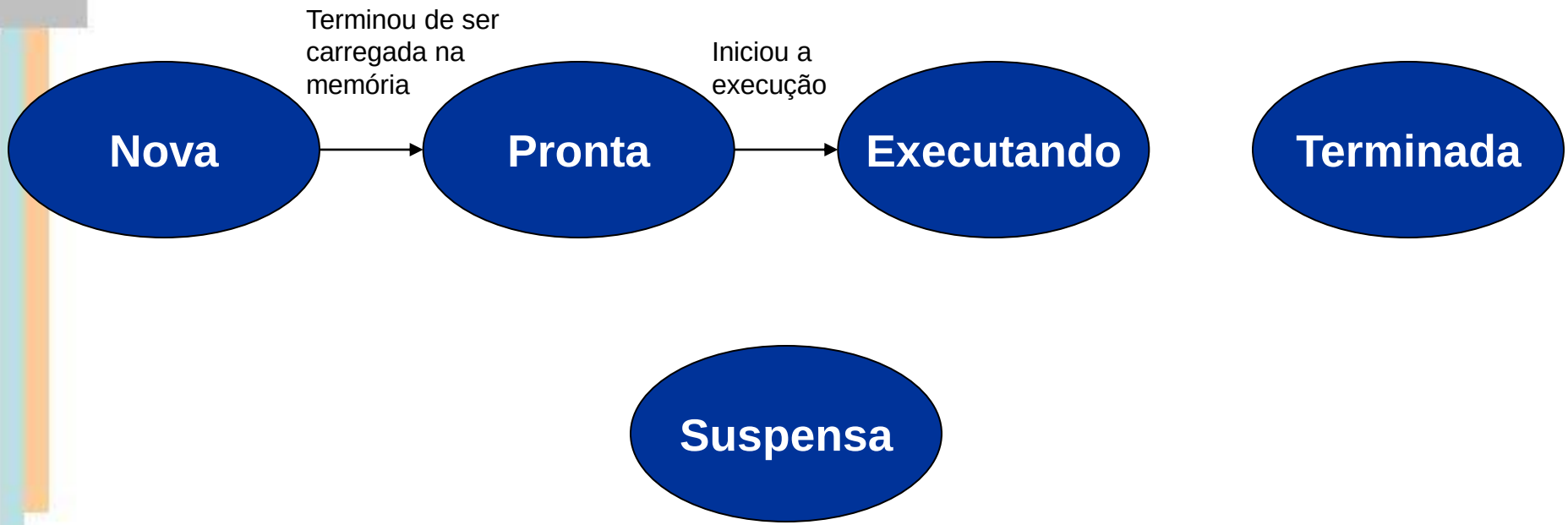
Sistemas multi-tarefa



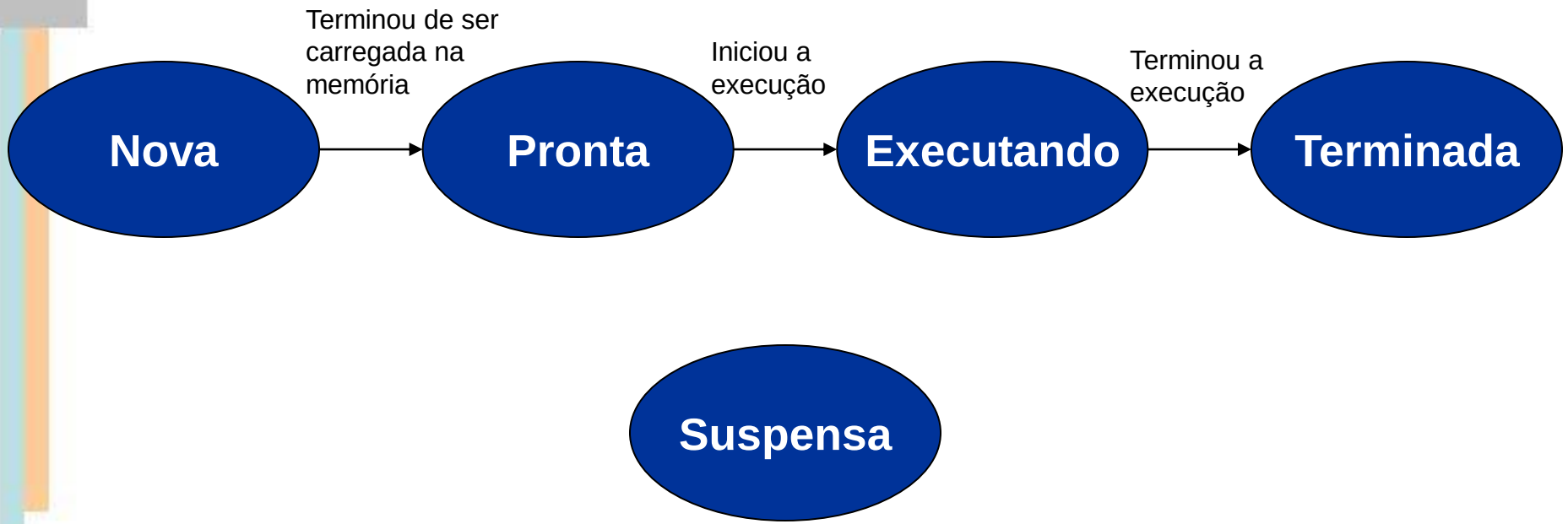
Sistemas multi-tarefa



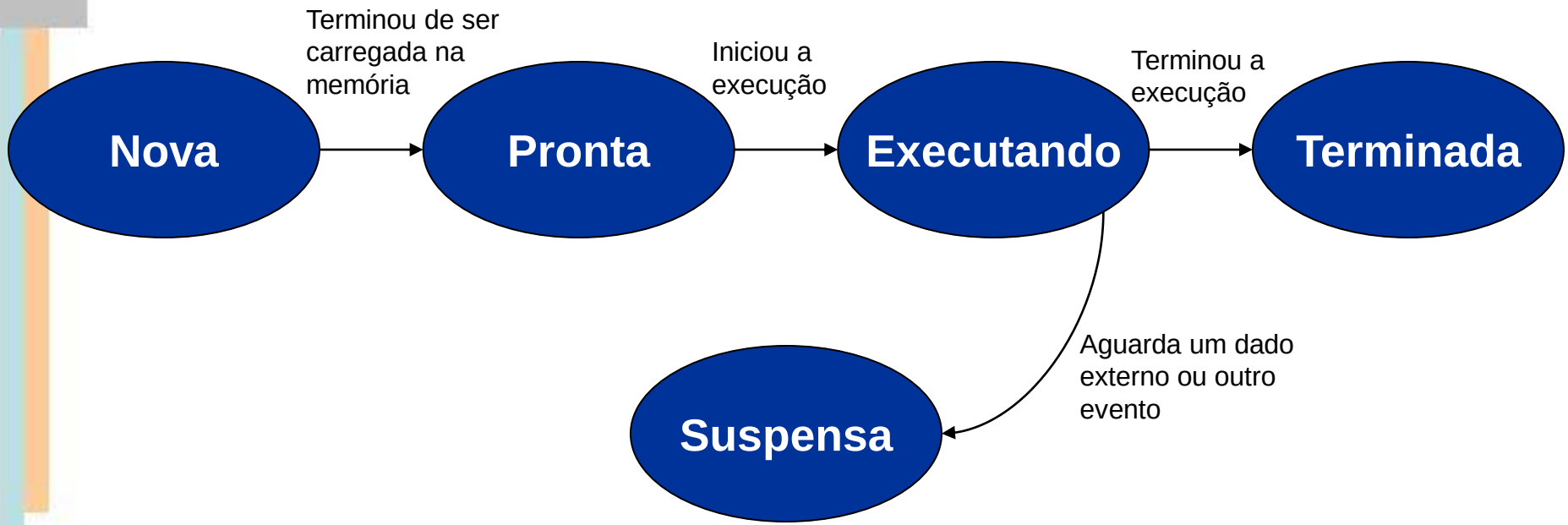
Sistemas multi-tarefa



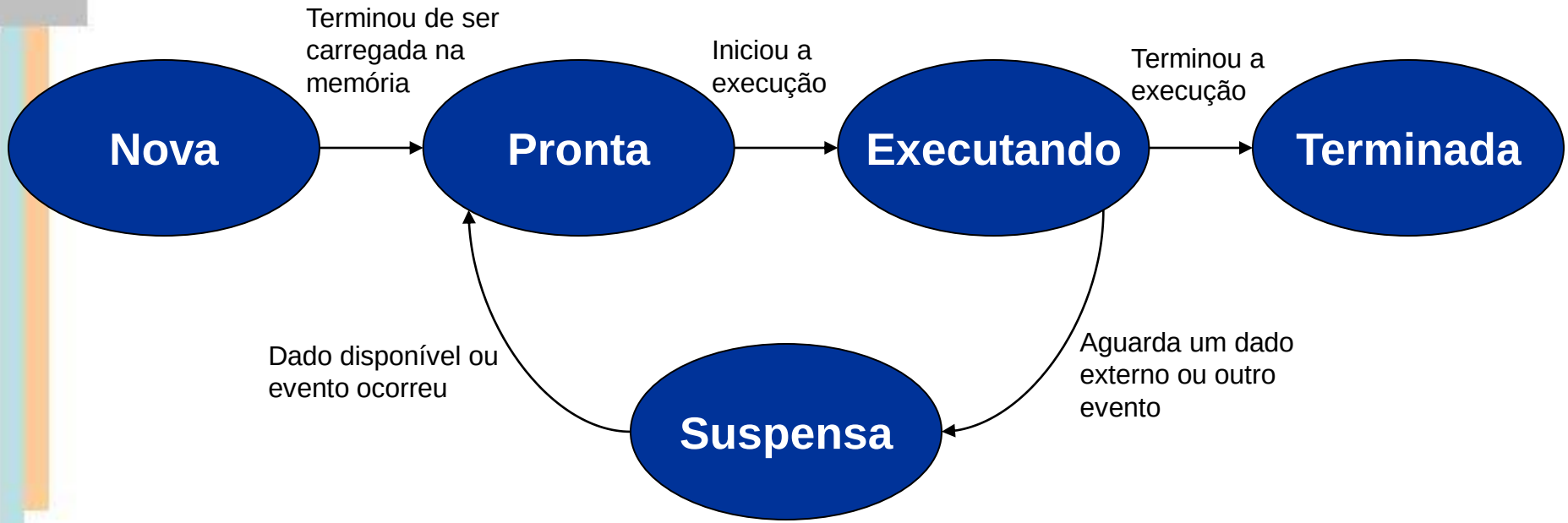
Sistemas multi-tarefa



Sistemas multi-tarefa



Sistemas multi-tarefa



Problema

Um programa que contém um laço infinito jamais se encerra ou é suspenso

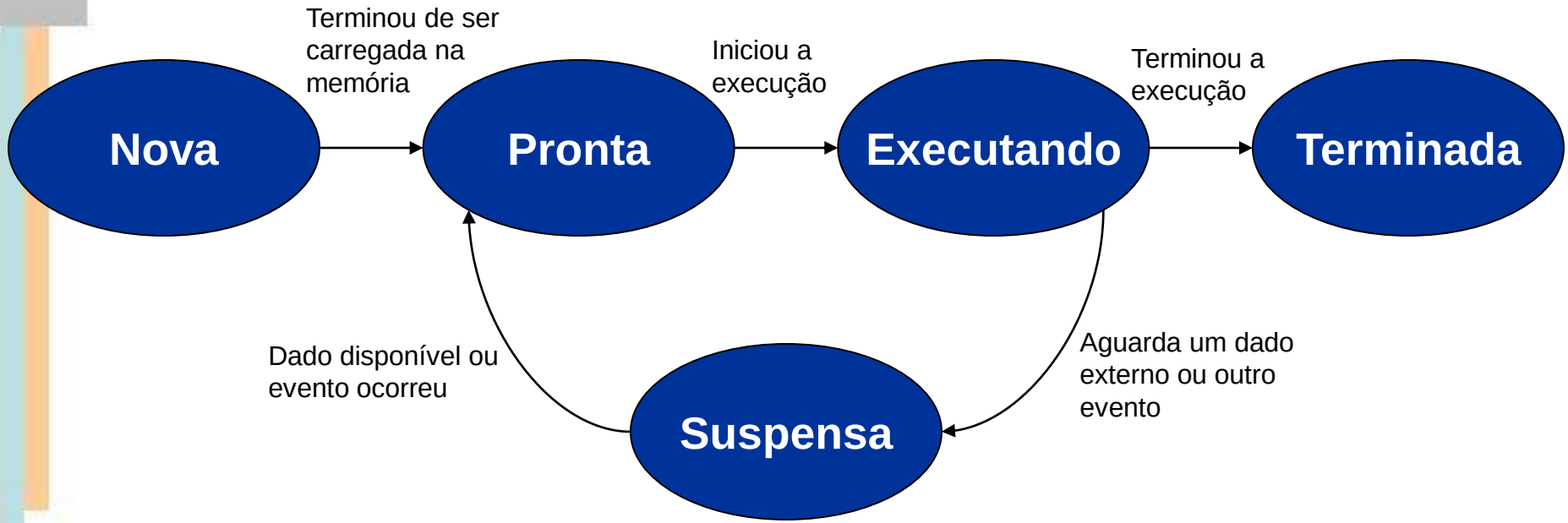
Loop infinito

```
void main ()
{
   int i ,soma = 0 ;
   while (i < 1000)
   {
      soma += i ;
  /* erro : o  contador i não foi
  incrementado*/
   }
   printf ("A soma vale %d\n", soma);
}
```

Problema

Como fazer para abortar a tarefa ou, ao menos, transferir o controle ao monitor para que ele decida o que fazer?

Sistemas multi-tarefa



Preempção

- Diz-se que um algoritmo/sistema operacional é preemptivo quando um processo que entra na CPU pode ser retirado antes do término da sua execução

Preempção

Em computação, **preempção** é o ato de interromper temporariamente uma tarefa sendo resolvido por um sistema computacional, sem precisar de sua cooperação, e com a intenção de retomar a tarefa depois. Tal mudança é conhecida como uma troca de contexto. É normalmente resolvida por uma tarefa privilegiada ou parte de um sistema conhecido como um escalonador preemptivo, que tem o poder de **preeminar**, ou interromper, e depois retomar, outras tarefas no sistema.

Fonte: Wikipedia: http://pt.wikipedia.org/wiki/Preemptividade

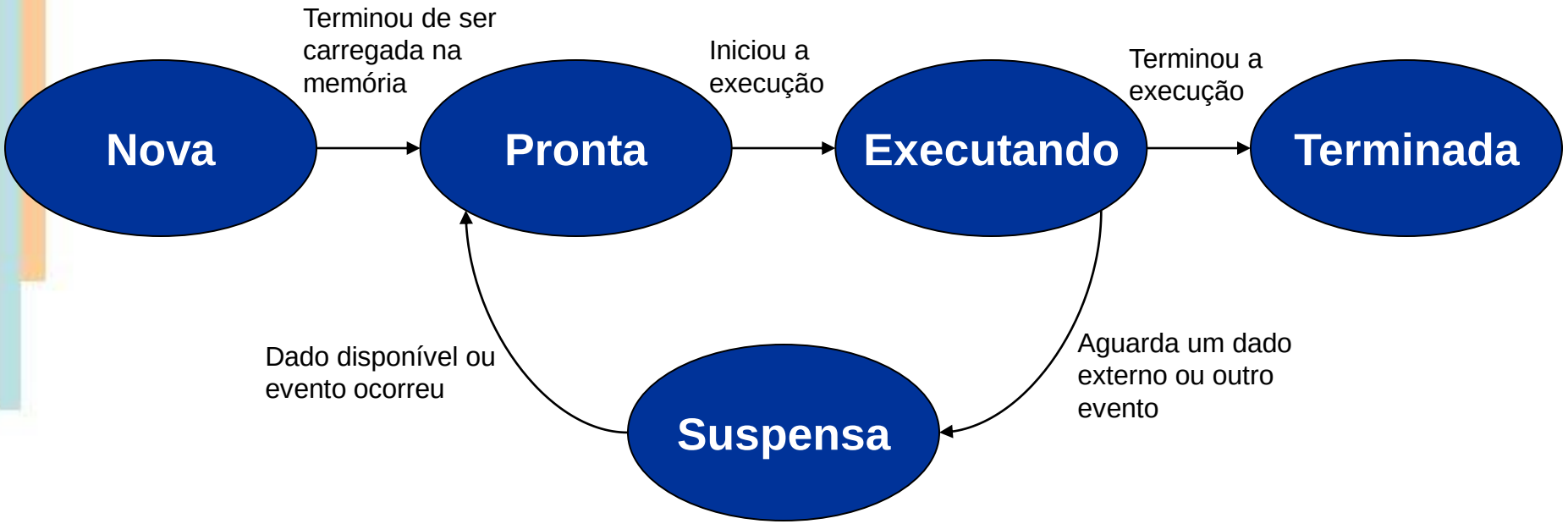
Time-sharing

Time-sharing

Cada tarefa que detém o processador recebe um limite de tempo de processamento, denominado *quantum*. Esgotado o *quantum* a tarefa em execução perde o processador e volta para a fila de tarefas “prontas” que estão na memória aguardando sua oportunidade de executar

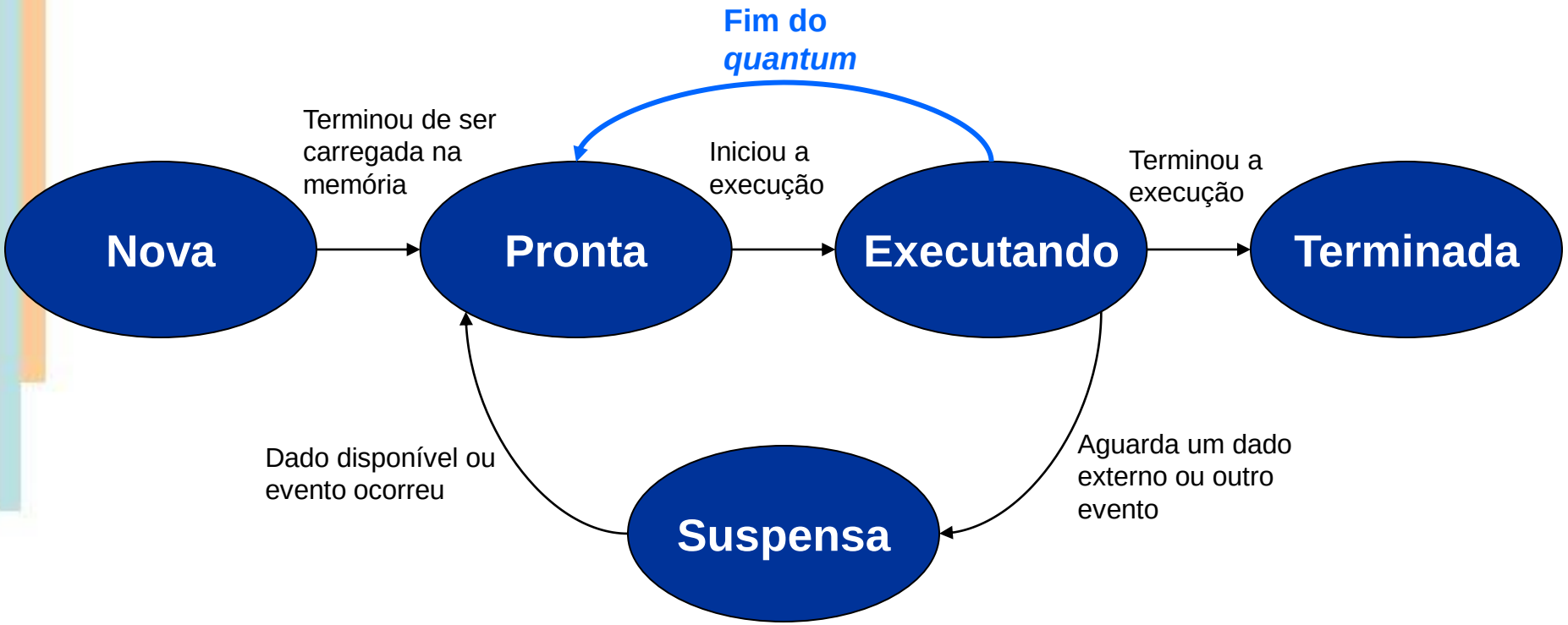
Time-sharing

Cada tarefa que detém o processador recebe um limite de tempo de processamento, denominado *quantum*. Esgotado o *quantum* a tarefa em execução perde o processador e volta para a fila de tarefas “prontas” que estão na memória aguardando sua oportunidade de executar



Time-sharing

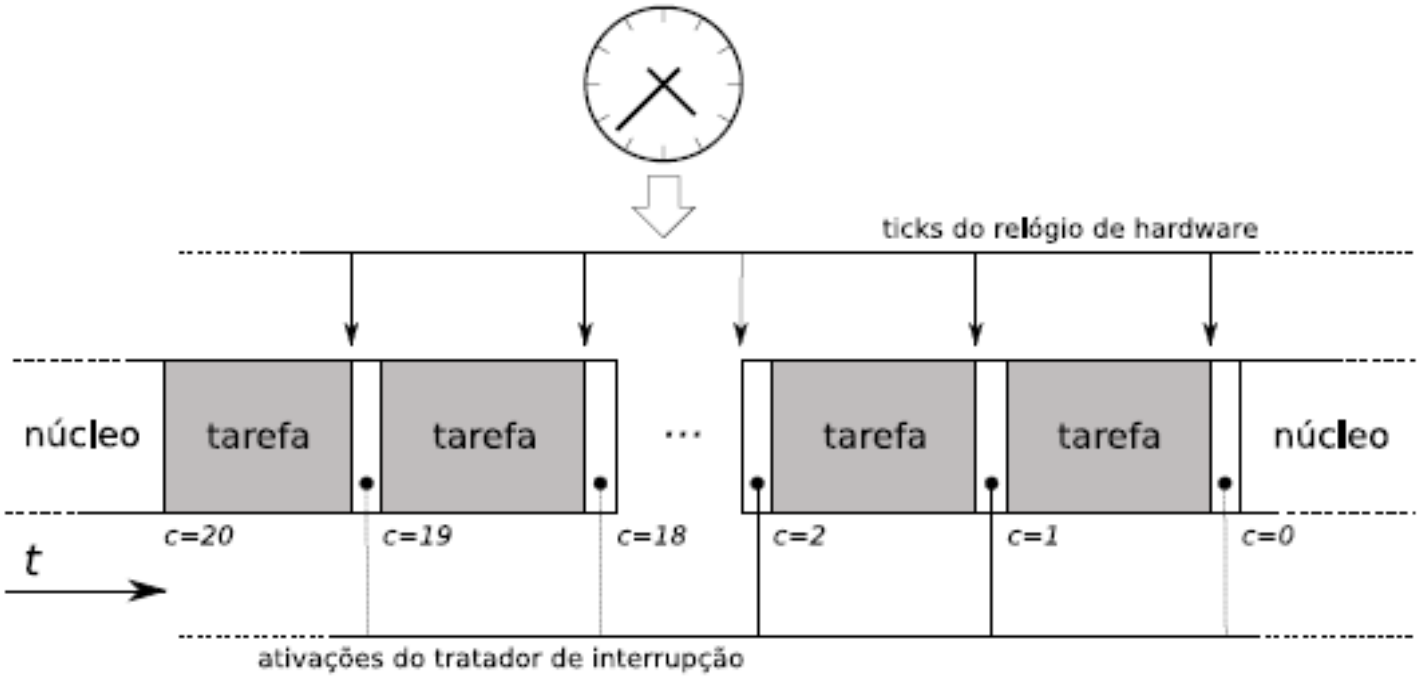
Cada tarefa que detém o processador recebe um limite de tempo de processamento, denominado *quantum*. Esgotado o *quantum* a tarefa em execução perde o processador e volta para a fila de tarefas “prontas” que estão na memória aguardando sua oportunidade de executar



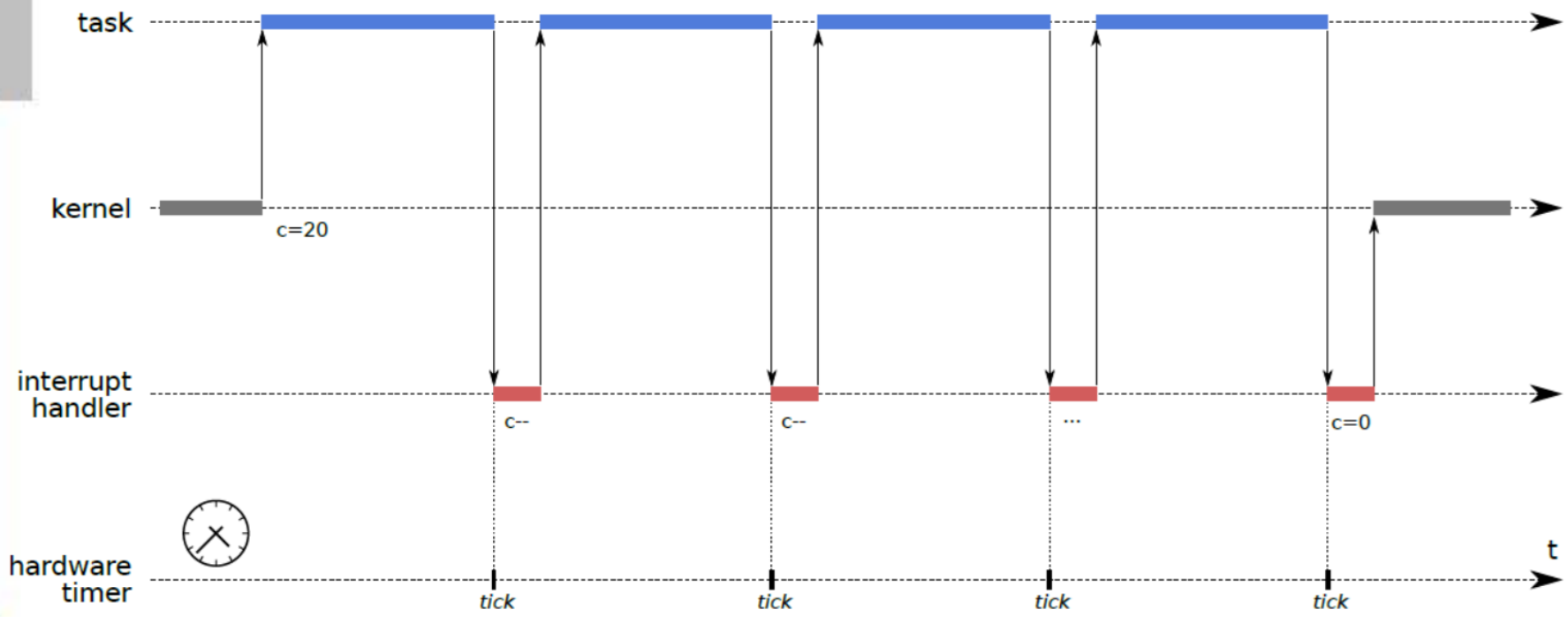
Time-sharing - implementação

Ticks do relógio $\rightarrow$ ativações periódicas do tratador de interrupção de timer

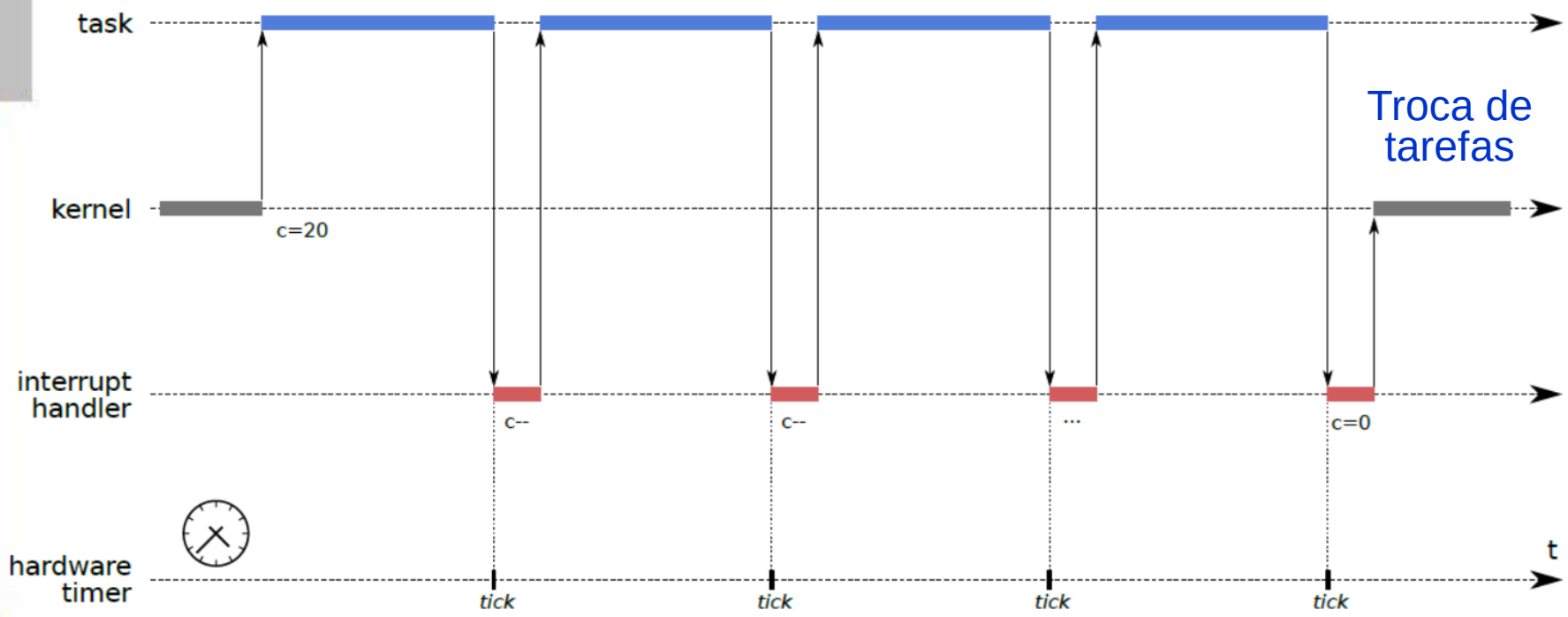
Quando uma tarefa recebe o processador o *núcleo* ajusta um contador de *ticks* que essa tarefa pode usar



Time-sharing - implementação



Time-sharing - implementação



Troca de tarefas

Contexto: informações que permitem definir completamente o estado de uma tarefa

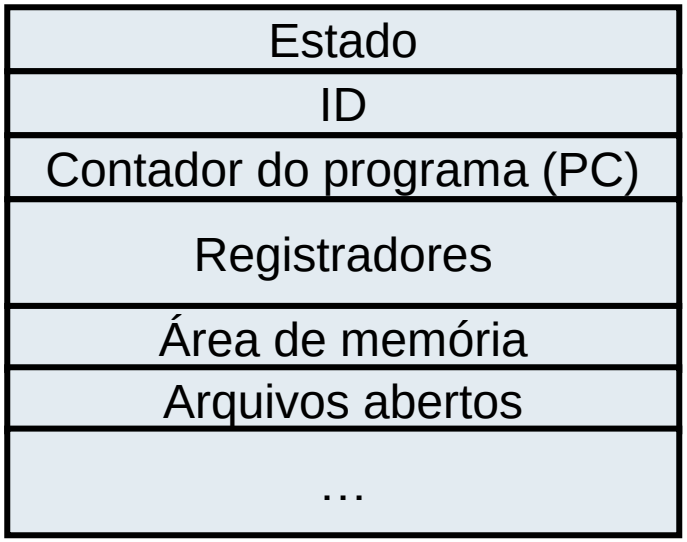
Troca de tarefas

Contexto: informações que permitem definir completamente o estado de uma tarefa

- **Caracterização de um estado**
- **Registradores do processador**
  - **PC**
  - **SP**
  - **Status do processador**
  - **Acumulador, Registrador de uso geral**
- **Área de memória**
- **Arquivos abertos, conexões de rede**

TCB – *Task Control Block*

Estrutura de dados associada a uma tarefa onde são armazenadas as informações relativas ao seu contexto e outros dados necessários à sua gerência

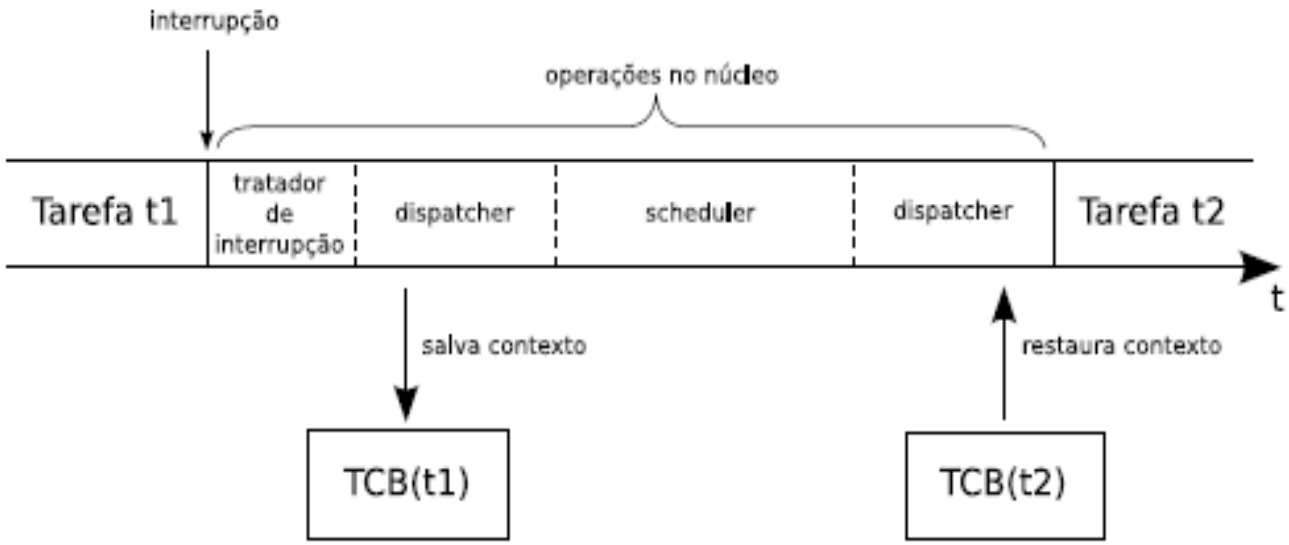


Ex.: TCB – *BRTOS*

```
typedef struct {
      pfTaskEntry     pfEntryPoint;      /* task entry point    */
      unsigned char   ucPriority;        /* task priority       */
      unsigned char   ucTaskState;       /* current task state  */
      unsigned short  usTimeSlice;       /* desired time slice  */
      unsigned short *pusStackBeg;       /* stack beginning     */
      unsigned short *pusStackPtr;       /* stack pointer       */
      unsigned short  usSleepTicks;      /* count sleep ticks   */
      unsigned short  usTicks;           /* count slice ticks   */
} BRTOS_TCB;
```

Fonte:
Fundamentos de Sistemas Operacionais de Tempo Real
Criando o seu próprio escalonador de tarefas
Marcelo Barros de Almeida

Troca de contexto



Despachante (*Dispatcher*) – Aspectos mecânicos

Escalonador (*Scheduler*) – Aspectos estratégicos

Eficiência da troca de contexto

$$\eta = \frac{t_q}{t_q + t_{tc}}$$

t_q t_{tc}

$t_q \Rightarrow$ Quantum de tempo

$t_{tc} \Rightarrow$ Tempo da troca de contexto

Ex.: Problema da troca de contexto

- Mudar de um processo para outro requer um certo tempo para a administração — salvar e carregar registradores e mapas de memória, atualizar tabelas e listas do SO, etc.
- Isto se chama troca de contexto
- Suponha que essa troca dure 5 ms
- Suponha também que o quantum está ajustado em 20 ms
- Com esses parâmetros, após fazer 20 ms de trabalho útil, a CPU terá que gastar 5 ms com troca de contexto
- Assim, 20% do tempo de CPU é gasto com o *overhead* administrativo

Ex.: Problema da troca de contexto

t_q t_{tc}

$$\eta = \frac{t_q}{t_q + t_{tc}} = \frac{20}{25} = 0.8$$

Solução ?

- Para melhorar a eficiência da CPU, poderíamos ajustar o quantum para 500 ms
- Agora o tempo gasto com troca de contexto é de 1 %

Solução ?

t_q t_{tc}

$$\eta = \frac{t_q}{t_q + t_{tc}} = \frac{500}{505} = 0.99$$

Solução ?

- Para melhorar a eficiência da CPU, poderíamos ajustar o quantum para 500 ms
- Agora o tempo gasto com troca de contexto é de 1 %
- Considere o que aconteceria se dez usuários apertassem a tecla <ENTER> exatamente ao mesmo tempo, disparando cada um processo
- Dez processos serão colocados na lista de processo aptos a executar

Solução ?

- Se a CPU estiver ociosa, o primeiro começará imediatamente, o segundo não começará antes de ½ segundo depois, e assim por diante
- O azarado do último processo somente começará a executar 5 segundos depois do usuário ter apertado <ENTER>, isto se todos os outros processos tiverem utilizado todo o seu quantum
- 5 segundos para um comando simples é “muita” coisa

Conclusão

- Ajustar um quantum muito pequeno causa muitas trocas de contexto e diminui a eficiência da CPU, mas ajustá-lo para um valor muito alto causa um tempo de resposta inaceitável para pequenas tarefas interativas

Conclusão

- Ajustar um quantum muito pequeno causa muitas trocas de contexto e diminui a eficiência da CPU, mas ajustá-lo para um valor muito alto causa um tempo de resposta inaceitável para pequenas tarefas interativas

- Um quantum em torno de 100 ms frequentemente é um valor razoável