Modelos de Programação de Tempo Real

Eduardo Ferreira dos Santos

Ciência da Computação
Centro Universitário de Brasília – UniCEUB

Março, 2016

Sumário

Sistemas de Tempo Real [Chagas, 2016]

- Os sistemas de tempo real podem ser classificados conforme a interação em:

Reativos

- Sistemas cujo escalonamento é dirigido pela interação com seu ambiente.
- Exemplo: Sistema para controle de incêndios que reage ao pressionar de um botão.

Embarcados

- Fazem partes de sistemas maiores não computacionais.
- Exemplo: controle de injeção de combustível, airbag, freios, etc.

Tempo

- Principal característica dos sistemas de tempo real: obedecer às restrições temporais.
  - Período de ativação;
  - *deadlines*.
- O recurso deve estar disponível no momento que for requisitado;
- É necessário atender os aspectos ligados à concorrência.

Concorrência Acesso simultâneo ao recurso por mais de uma tarefa.

- A falha na solicitação por um recurso faz com que o sistema não atenda a um *deadline*

Corretude [Chagas, 2016]

- Um sistema falho é um sistema que não pode satisfazer um ou mais dos requisitos estipulados em sua especificação;
- O comportamento correto de um sistema de tempo real depende:

  Correção lógica (correctness) integridade dos resultados obtidos.
  Correção temporal (timeliness) valores de tempo em que são produzidos.
- Uma reação que ocorra além do prazo especificado pode ser inútil ou até representar uma ameaça.

Um Sistema de Tempo Real deve ser então capaz de oferecer garantias de correção temporal para o fornecimento de todos os seus serviços que apresentem restrições temporais.
[FARINES and MELO, 2000]

Classificação [Chagas, 2016]

- Os sistemas de tempo real podem ser classificados conforme o impacto gerado por uma falha ao atender seus requisitos de tempo em:

  SRT Tolerantes (Soft Real Time) Sistema em que o desempenho é degradado mas não resulta em falhas, no caso de não atendimento de suas restrições de tempo.

  STR Rigorosos (Hard Real Time) Sistema em que uma falha relacionada a um único deadline pode provocar falhas completas do sistema ou até mesmo catástrofes.

  STR Seguros (Firm Real Time) Sistema em que a perda de poucos deadlines não provocam falha total, no entanto, a perda de uma quantidade muito grande podem provocar falhas completa do sistema ou até mesmo catástrofes

Exemplos e classificação

Sistema	Tipo de Sistema	Cenário
Máquinas de auto-atendimento	Tolerante	Perda de muitos *deadlines* não provocarão falhas catastróficas, somente o desempenho é degradado.
Sistema de navegação embutida para controle de robôs autônomos agrícolas.	Seguro	Excessiva perda de *deadlines* podem fazer que o robô danifique toda uma plantação.
Sistema de controle de armas em caças.	Rigoroso	Perda de um único *deadline* pode fazer com que o alvo seja perdido.

Figura 1.1: Exemplos e Classificação [Chagas, 2016]

Estados dos processos

Durante o ciclo de vida de um processo ele passa por diferentes estados. Em sistemas Unix [Guarezi and Silva, 2010] são:

run Está sendo executado no processador;

ready ou executável Dispõe de todos os recursos que precisa e está pronto para ser executado;

sleep ou dormente Bloqueado à espera de algum recurso, e só pode ser desbloqueado se receber um sinal de outro processo;

zumbi Caso cada vez mais raro, onde um processo é criado por um programa, que por sua vez é finalizado antes de receber o resultado do processo;

parado Recebeu ordem do administrador para interromper a execução. Será reiniciado se receber um sinal de continuação (CONT).

Estados dos processos (Gráfico)

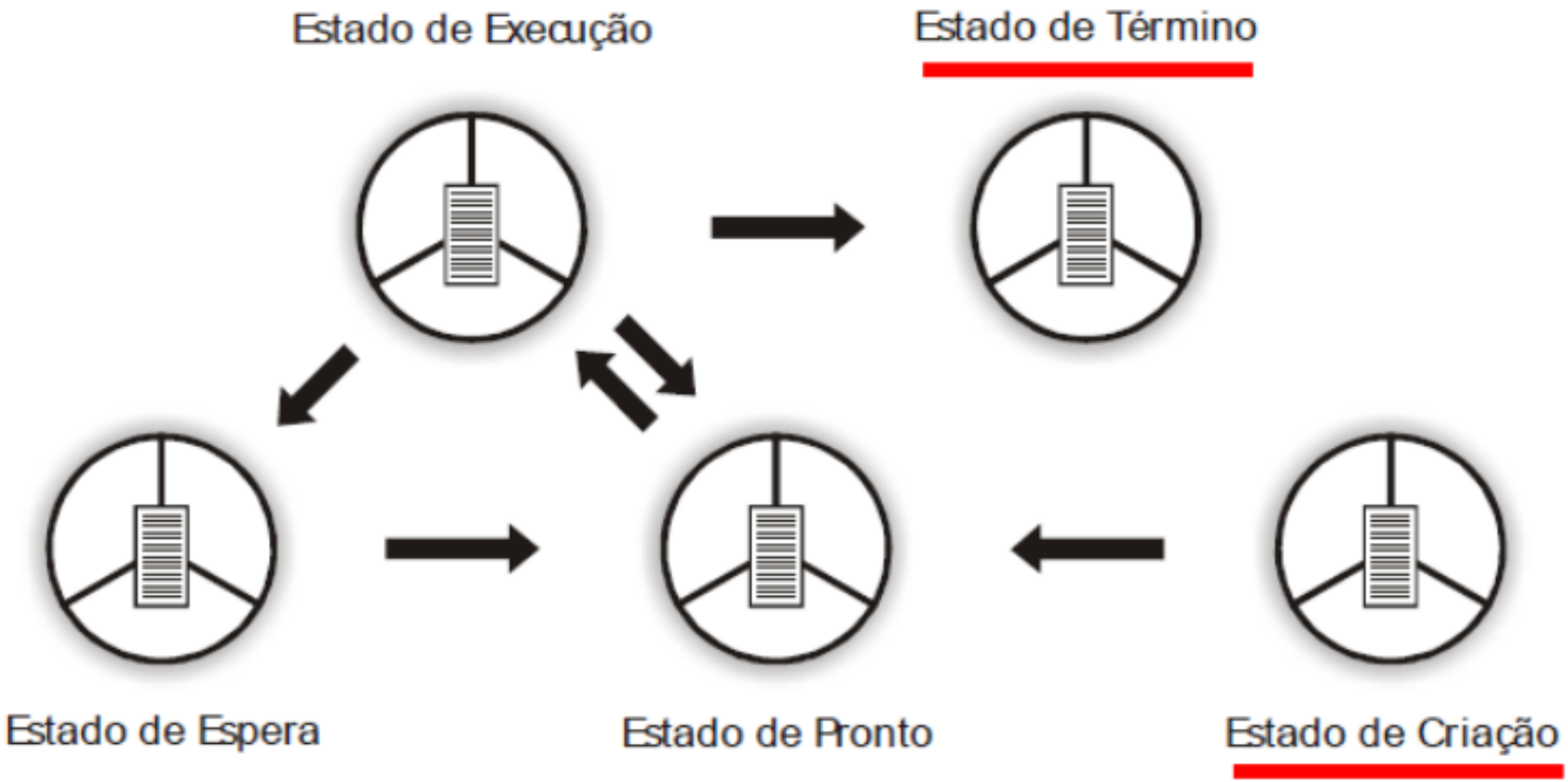


Figura 2.1: Estados dos processos [Chagas, 2016]

Programação Síncrona

- Concorrência [Chagas, 2016]:
  1. O programa perde o uso do processador;
  2. O programa retorna para continuar o processamento;
  3. O estado do programa deve ser idêntico ao do momento em que foi interrompido.
- O programa continua a execução exatemente na instrução seguinte.

Premissa 1 O ambiente não interfere com o sistema no momento da execução.

Premissa 2 Existe uma máquina suficientemente rápida para executar o processamento correspondente à reação em tempos não significativos.

O modelo é dito síncrono porque as saídas do sistema podem ser vistas como sincronizadas com as suas entradas.
[FARINES and MELO, 2000]

Modelos Síncronos

Uma das características mais importantes encontrada nos modelos síncronos é a rejeição do não determinismo. [FARINES and MELO, 2000, p.105]

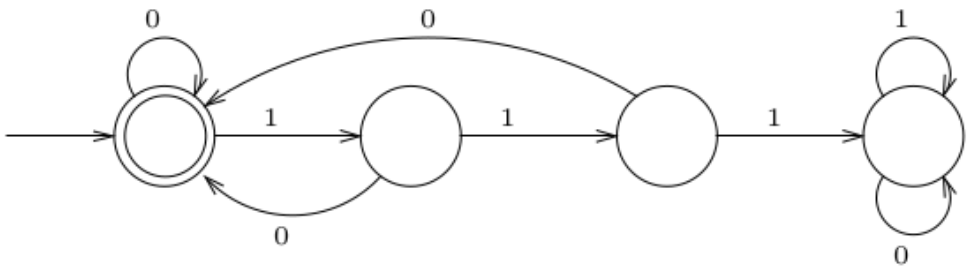
Determinismo Para cada estado do programa, existe somente uma possibilidade para a função de transição (próximo estado);

Não determinismo Podem existir várias escolhas para o próximo estado em qualquer ponto.

Determinismo

Determinístico

Exatamente uma trajetória sobre uma $w \in \Sigma^*$.



Não-determinístico

Nenhuma, uma ou várias trajetórias sobre uma $w \in \Sigma^*$.

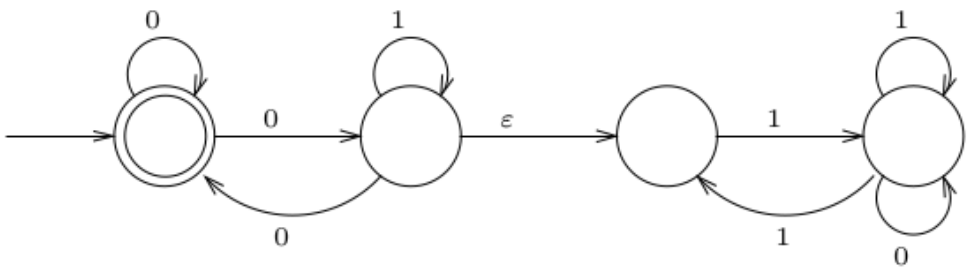


Figura 2.2: Autômatos Finitos Determinísticos e Não Determinísticos [Rezende, 2016]

Concorrência

- Paradigma: controlar/restringir o acesso ao recurso em determinado espaço de tempo;
- O controle de acesso aos recursos é realizado através de eventos;
- Eventos inesperados pode causar um desvio inesperado no fluxo de execução.

Interrupção Realizada por algum evento externo ao programa, independente da instrução. Ex.: *SYSCALL*

- Podem ser geradas por eventos assíncronos;
- Até várias vezes ao mesmo tempo.

Exceção Erro na instrução de algum programa. Ex.: falha de segmentação (*segfault*).

- Sempre gerada por um evento síncrono;
- Somente um evento de cada vez.

Interrupção

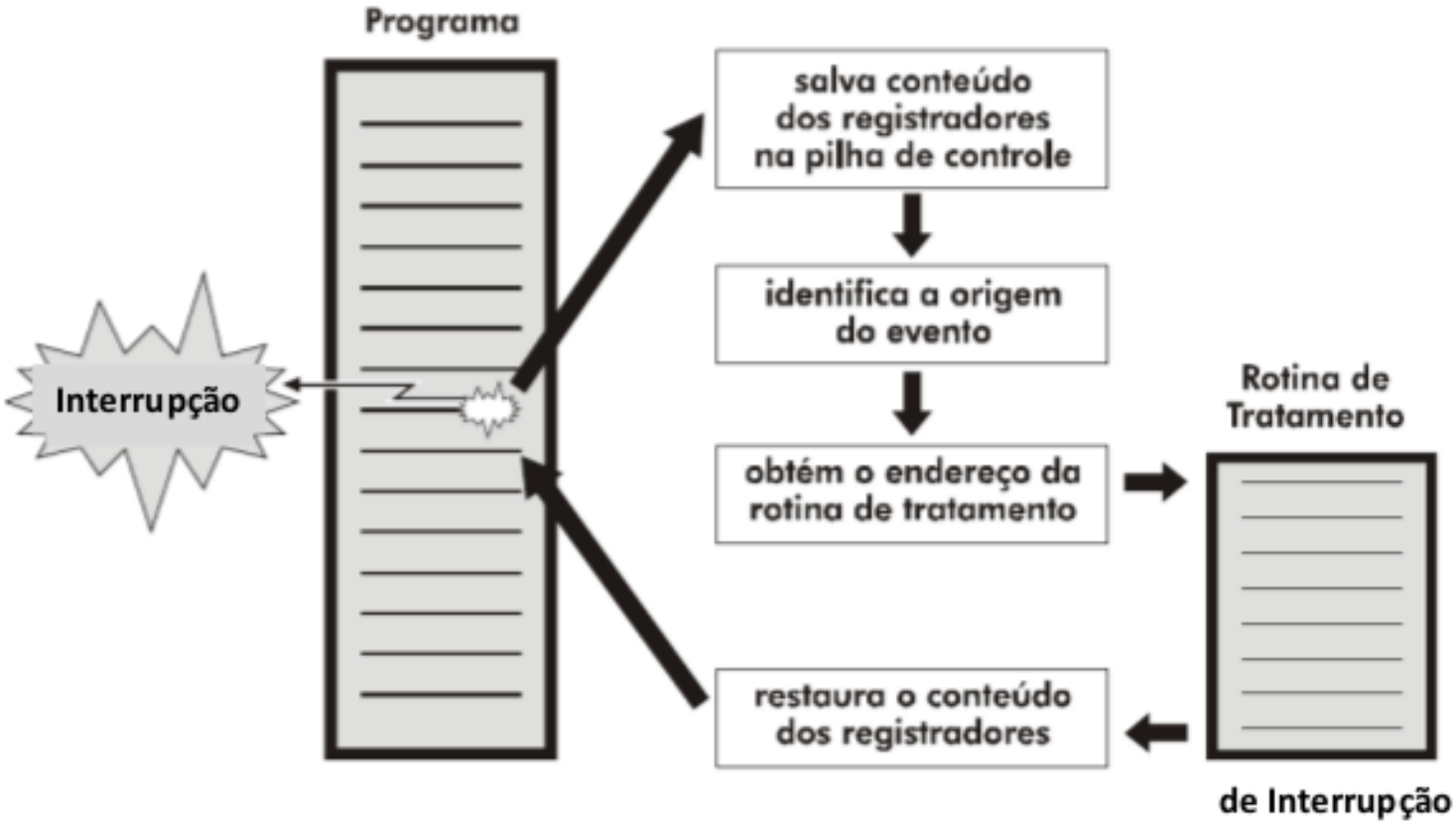


Figura 2.3: Tratamento de Interrupção [Chagas, 2016]

Exceção

Figura 2.4: Tratamento de Exceção [Rezende, 2016]

Conceitos

- Na programação concorrente existe mais de uma tarefa sendo executado ao mesmo tempo. Ex.: Fatorial
- No caso de múltiplas tarefas é necessário haver comunicação entre elas.

Memória compartilhada As tarefas compartilham área de memória;

Troca de mensagens Sinais trocados entre processos.

Comunicação entre processos

- Gerência de recursos de memória compartilhada: condição de corrida
  - Exclusão mútua;
  - Semáforo;
  - Monitor.
- Comunicação por troca de mensagens: deadlocks
  - Leitura assíncrona;
  - Método *rendezvous*.

Condição de corrida

Situações onde dois ou mais processos estão acessando dados compartilhados, e o resultado final do processamento depende de quem roda quando.

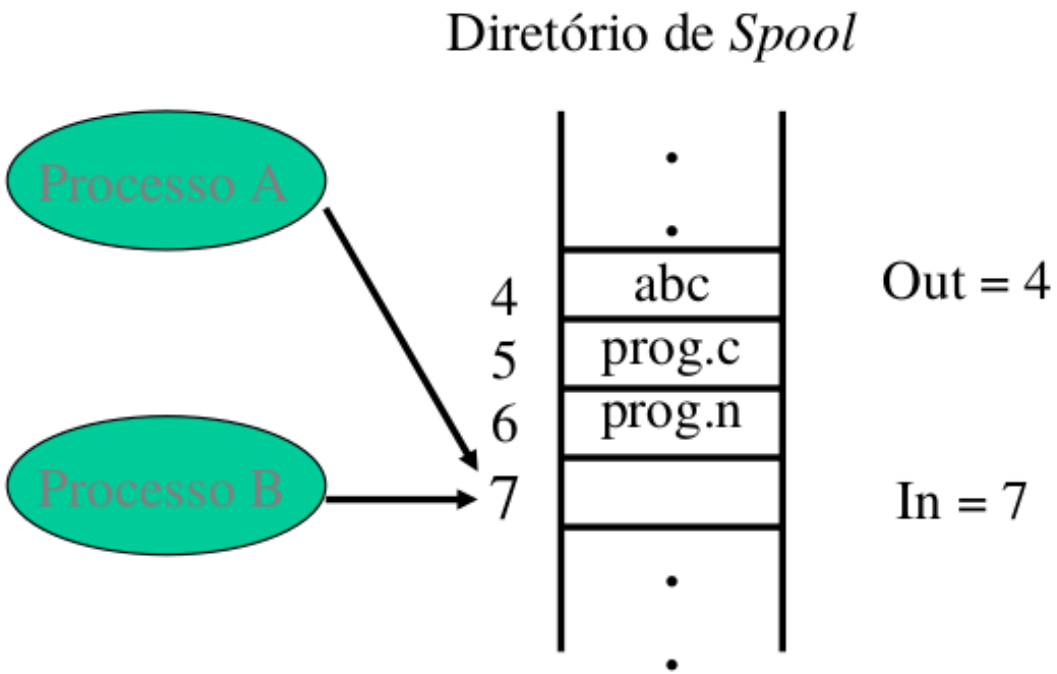


Figura 3.1: Exemplo da condição de corrida

Exclusão mútua [Chagas, 2016]

- Solução: impedir que mais de um processo acesso o dado ao mesmo tempo.
- Deve ser executada somente quando um dos processos estiver acessando o recurso compartilhado;
- A parte do código onde o acesso ao recurso é feito é chamada de região crítica.

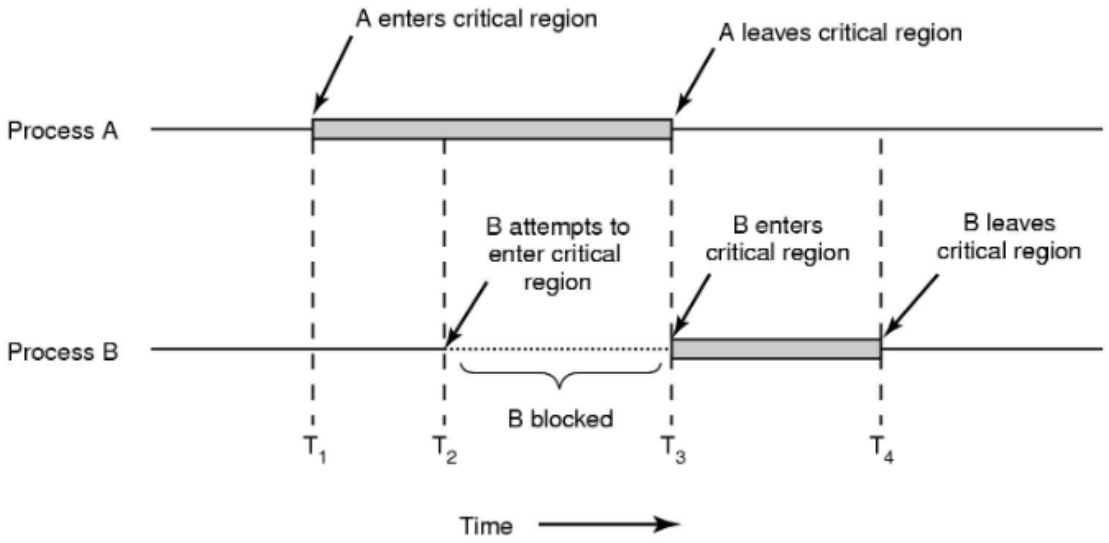


Figura 3.2: Região crítica

Semáforos [Favacho, 2009]

- Baseado em um tipo de variável que possui dois estados: UP e DOWN.
  1. O semáforo fica associado a um recurso compartilhado;
  2. Se o valor da variável semáforo for diferente de zero, nenhum processo está utilizando o recurso; caso contrário,o processo fica impedido do acesso;
  3. Sempre que deseja entrar em sua região crítica, o processo executa uma instrução DOWN;
  4. Se o semáforo for maior que 0, este é decrementado de 1, e o processo que solicitou a operação pode executar sua região crítica;
  5. Entretanto, se uma instrução DOWN é executada em um semáforo cujo valor seja igual a 0, o processo que solicitou a operação ficará no estado de espera;

Semáforos (cont.) [Favacho, 2009]

6. Além disso, o processo que está acessando o recurso, ao sair de sua região crítica, executa uma instrução UP, incrementando o semáforo de 1 e liberando o acesso ao recurso;
7. A verificação do valor do semáforo, a modificação do seu valor e, eventualmente a colocação do processo para dormir são operações atômicas;
8. Operações atômicas são únicas e indivisíveis;
9. Os semáforos aplicados ao problema da exclusão mútua são chamados de mutex (*mutual exclusion*) ou binários, por apenas assumirem os valores 0 e 1.


Chagas, F. (2016).
Notas de aula do Prof. Fernando Chagas.


FARINES, J. M. and MELO, R. (2000).
Sistemas de Tempo Real, volume 1.
IME-USP.


Favacho, A. (2009).
Notas de aula da Profa. Aletéia Favacho.


Guarezi, D. J. and Silva, E. B. (2010).
Processos em windows e unix.
Disponível em:
http://www.inf.ufsc.br/~magro/PROCESSOS%20EM%20WINDOWS%2
Acessado em 28/01/2011.


Rezende, P. (2016).
Notas de aula do Prof. Pedro Rezende.
Disponível em: `http://cic.unb.br/~rezende/tc.html` Acessado em 14/03/2016.

UniCEUB
Centro Universitário de Brasília

OBRIGADO!!!
PERGUNTAS???