Sistemas Operacionais de Tempo Real

Fundamentos dos Sistemas de Tempo Real

Rômulo Silva de Oliveira

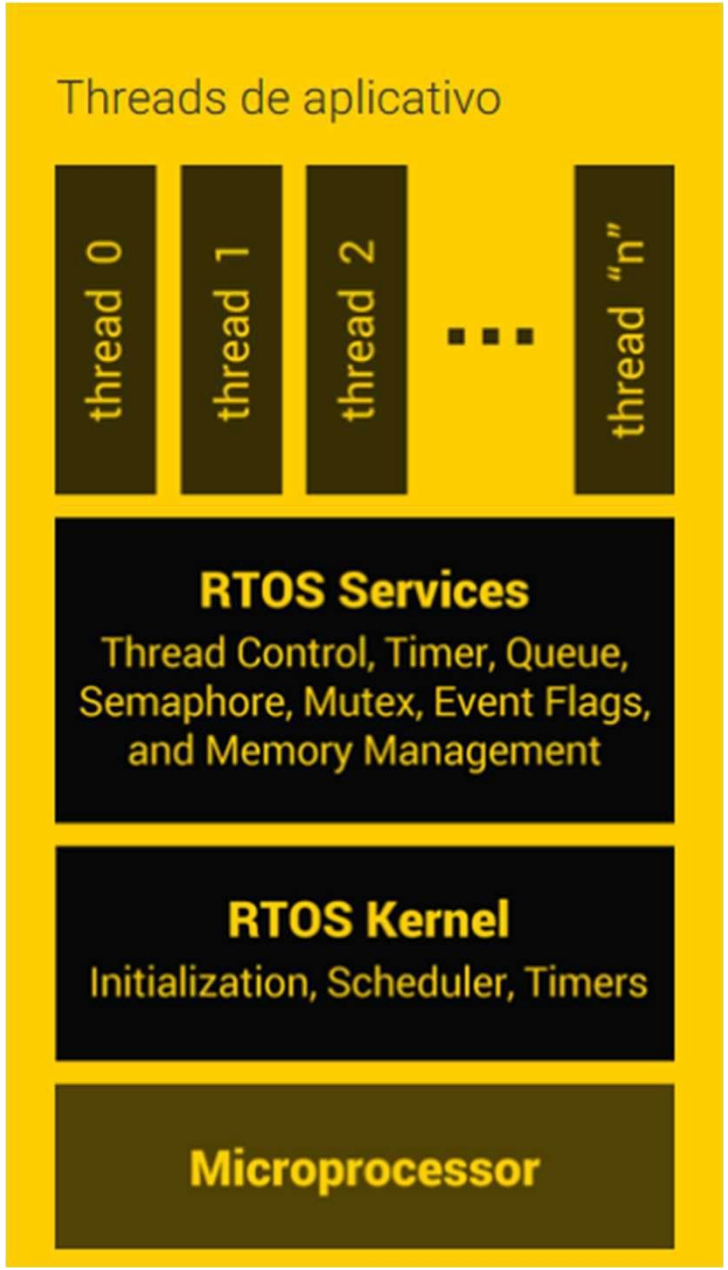
Edição do Autor, 2018

www.romulosilvadeoliveira.eng.br/livrotemporeal

Outubro/2018

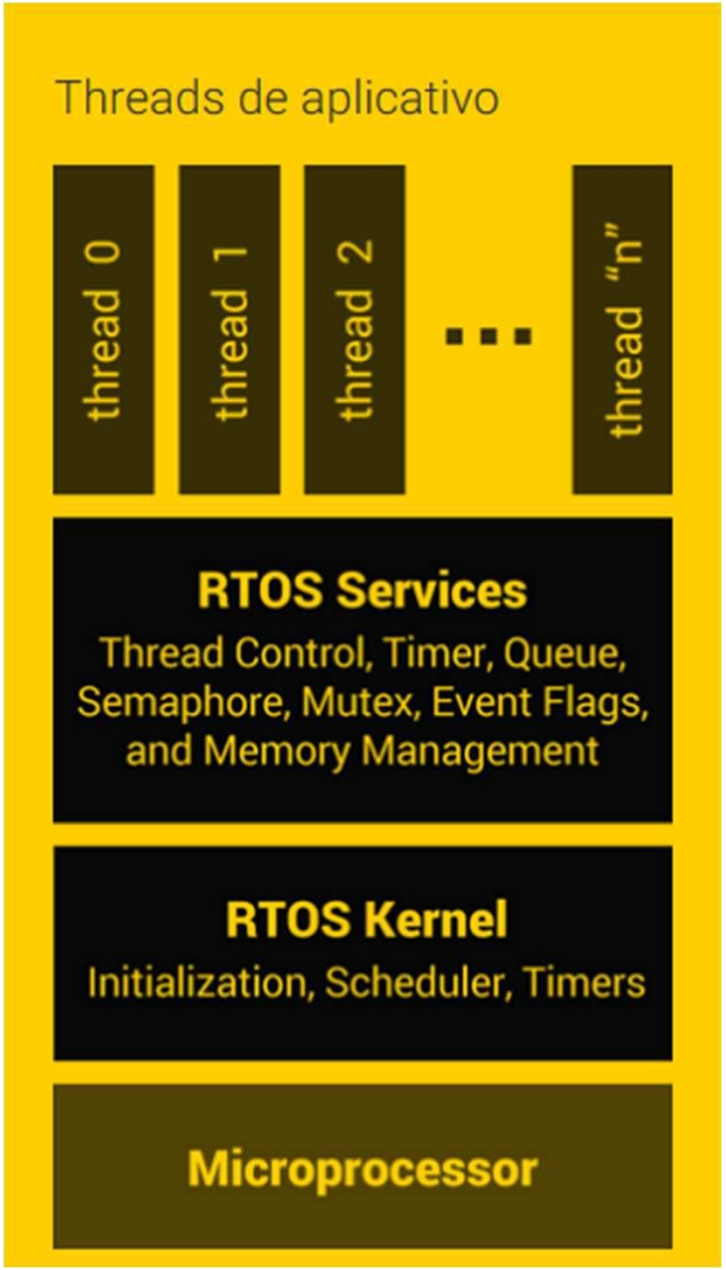
O que exatamente é um sistema operacional em tempo real?

- Um software de sistema que fornece serviços e gerencia recursos de processador para aplicativos
- Alocar o tempo de processamento entre várias tarefas.
- Preempção de tarefas.



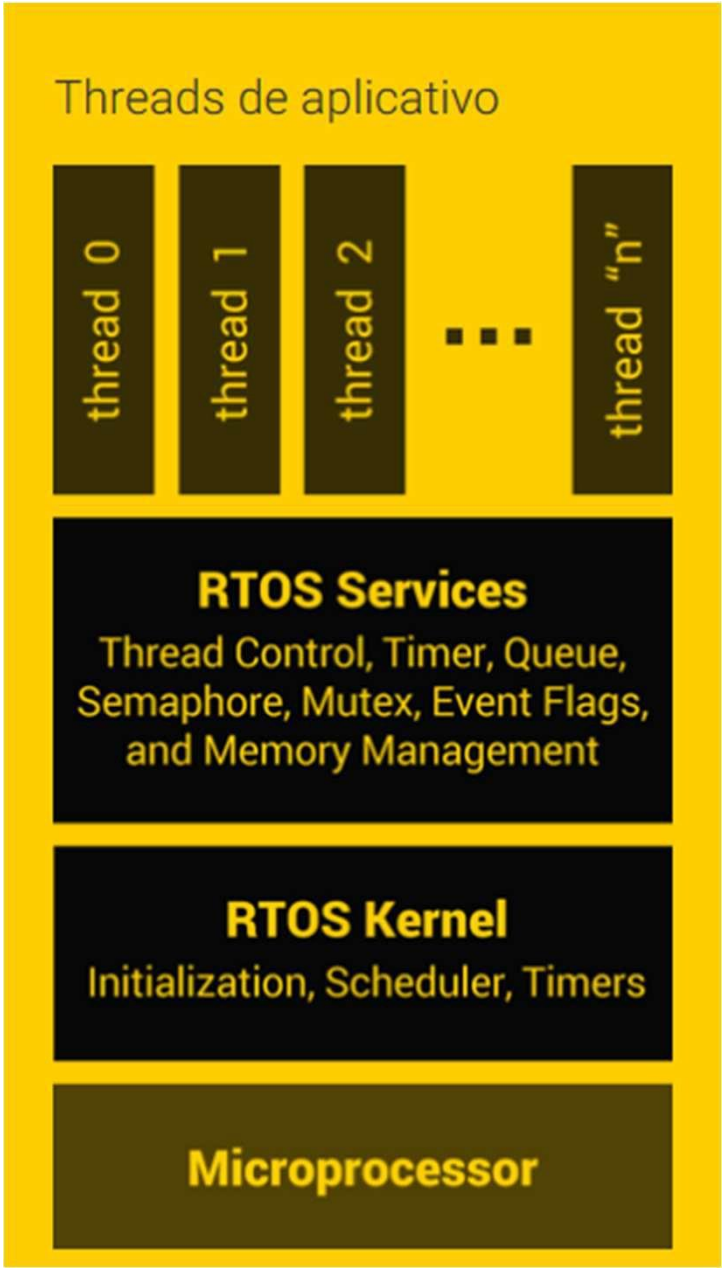
Quando você precisa de um sistema operacional em tempo real?

- Cabe a cada equipe de desenvolvimento avaliar o peso de cada lado, para que a melhor escolha para o seu projeto possa ser feita. Mesmo se você não precisar de um RTOS, os benefícios podem superar o custo e torná-lo uma escolha inteligente.



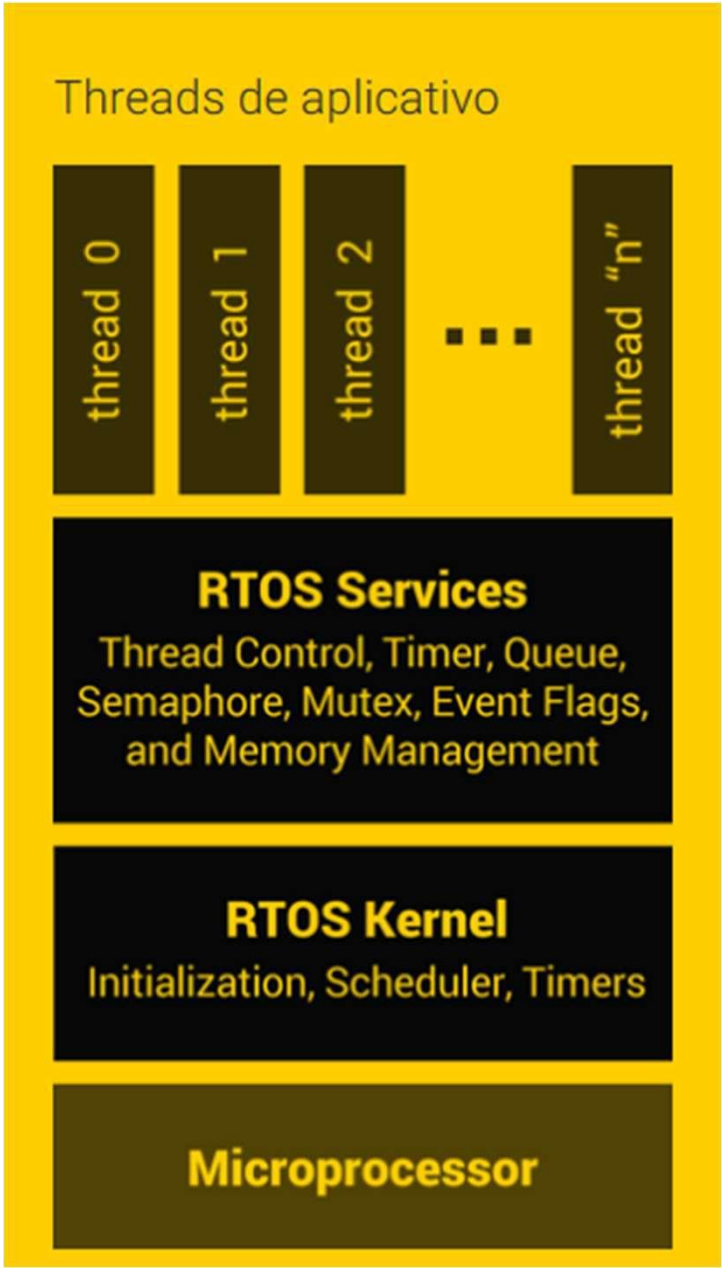
Quais são os benefícios de um RTOS?

- RTOS fornece desempenho rápido e garantido em tempo real
- Um RTOS pode reduzir a sobrecarga
- Um RTOS facilita o desenvolvimento
- Um RTOS facilita a adição de novos recursos
- Portabilidade mais fácil dos aplicativos
- Certificação de Segurança



Quais são os custos de um RTOS?

- Um bom RTOS requer compra
  - *O ponto de partida para a maioria das licenças RTOS comerciais é da ordem de US $ 10.000 para uma licença livre de royalties com código fonte completo e suporte completo.*
- Um RTOS requer memória
- Aprendendo a usar um RTOS
- Compra x Construção



Introdução 2/4

- Qualquer programa de computador, seja de tempo real ou não, será mais facilmente construído se puder aproveitar os serviços de um sistema operacional
- O sistema operacional permite que o programador da aplicação utilize abstrações de mais alto nível
  - Processos
  - Threads
  - Arquivos
  - Segmentos de memória
  - Sockets de comunicação
- Sem precisar lidar com a gerência dos recursos básicos do hardware

Introdução 4/4

- **Definição mais adequada para SOTR**

“um sistema operacional apropriado para aplicações de tempo real”

Aspectos Funcionais dos SOTR 1/5

- Como qualquer sistema operacional, um SOTR procura tornar a utilização do computador mais eficiente e mais conveniente
  - Facilidades providas por SOPG são bem vindas em um SOTR
- Com respeito à funcionalidade de um SOTR
  - É razoável supor que o mesmo deveria suprir os mesmos serviços que um SOPG
  - A aplicação de tempo real precisa dos mesmos serviços que as demais aplicações
  - Ela apenas coloca requisitos adicionais, de natureza temporal

Aspectos Funcionais dos SOTR 2/5

- Em sistemas embutidos (embedded systems) com sérias limitações de memória, energia e outras
  - SO deveria prover apenas os serviços realmente usados pela aplicação
  - No sentido de não gastar recursos do hardware para implementar serviços que jamais serão usados pela aplicação

Aspectos Funcionais dos SOTR 3/5

- Existem alguns serviços adicionais e/ou melhorados que um SOTR deve prover
- Serviço de relógio (gettime)
  - Permite que o processo obtenha a hora e data correntes, preferencialmente conforme a UTC ou algo relacionado
- Temporização de intervalo (sleep)
  - Permite que o processo fique suspenso durante um determinado intervalo de tempo, especificado como parâmetro
- Temporização de instante (wake-up)
  - Permite que o processo fique suspenso até um determinado instante de tempo, especificado como parâmetro
- Monitoração temporal das tarefas (watch-dog)
  - Dispara a execução de um tratador de exceção caso o processo não sinalize que terminou determinado conjunto de tarefas até um instante de tempo especificado como parâmetro

Diferenças Construtivas entre SOPG e SOTR 1/14

- **Algoritmo de Escalonamento Apropriado**

- Ofereça métodos de análise e testes de escalonabilidade
- O escalonamento mais usado em sistemas de tempo real é aquele baseado em prioridades fixas preemptivas
- Este é o algoritmo implementado pela maioria dos sistemas operacionais de tempo real

Diferenças Construtivas entre SOPG e SOTR 2/14

- **Níveis de Prioridade Suficientes**

- Escalonamento baseado em prioridades é suportado pela maioria dos sistemas operacionais
  - Número de diferentes níveis de prioridade varia bastante
- Quando o número de níveis de prioridade disponíveis é menor do que o número de tarefas, e passa a ser necessário agrupar várias tarefas no mesmo nível, isto reduz a escalonabilidade do sistema
- O desejado é que o número de níveis de prioridade seja igual ou maior do que o número de tarefas no sistema

Diferenças Construtivas entre SOPG e SOTR 3/14

- **Sistema Operacional não Altera Prioridades das Tarefas**

- Muitos sistemas operacionais de propósito geral também incluem mecanismos que reduzem automaticamente a prioridade de uma thread na medida que ela consome tempo de processador
  - Diminui o tempo médio de resposta no sistema
- Em um SOTR esses mecanismos
  - Não contribuem para a qualidade temporal
  - Tornam mais complexa a verificação do cumprimento dos requisitos temporais
  - Devem ser evitados

Diferenças Construtivas entre SOPG e SOTR 4/14

- **Tratadores de Interrupções com Execução Rápida**

- O tratador de interrupção deve realizar apenas aquele processamento mínimo que é necessário imediatamente na ocorrência da interrupção

Diferenças Construtivas entre SOPG e SOTR 5/14

- **Desabilitar Interrupções ao Mínimo**

- Por vezes, trechos de código do sistema operacional precisam executar com as interrupções desabilitadas
  - Deve ser minimizado

Diferenças Construtivas entre SOPG e SOTR 6/14

- **Emprego de Threads de Kernel**

- Em um SOTR é importante que todas as atividades do kernel ou microkernel sejam feitas ou por tratadores de interrupção rápidos
- Ou por threads de kernel as quais são liberadas pelos tratadores de interrupção
- Estas threads de kernel devem executar conforme a sua prioridade
  - Podendo ter prioridade menor que algumas tarefas de tempo real da aplicação
  - Conforme a atribuição de prioridades escolhida pelo desenvolvedor da aplicação
  - Por exemplo, driver do teclado versus tarefa de controle realimentado

Diferenças Construtivas entre SOPG e SOTR 7/14

- **Tempo de Chaveamento entre Tarefas Pequeno**

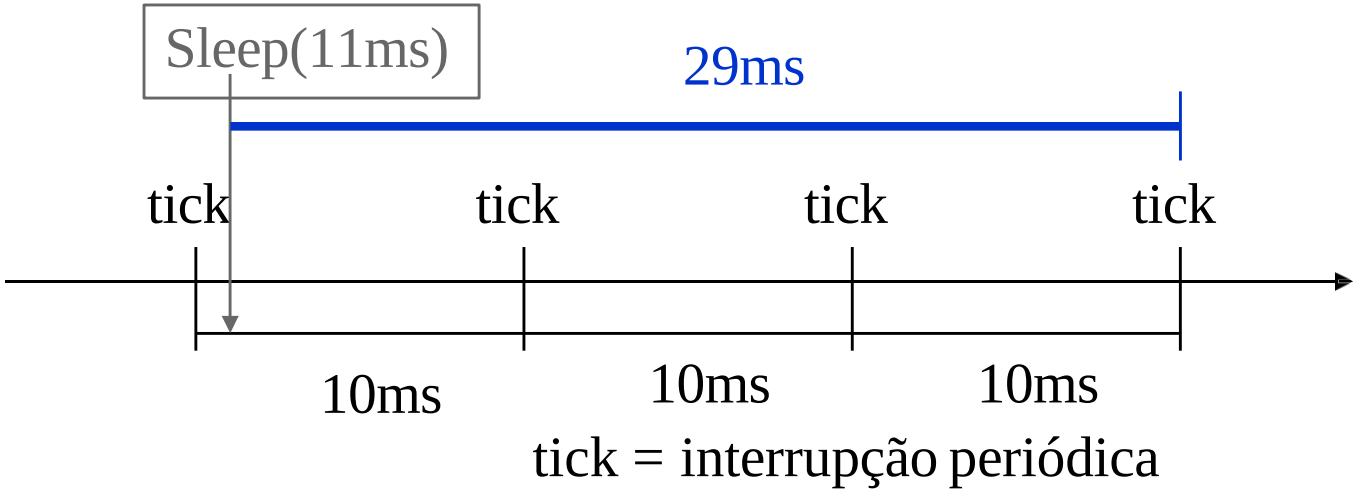
- Uma métrica muito citada no mercado de sistemas operacionais é o tempo para chaveamento de contexto entre duas tarefas
- Inclui salvar os registradores da tarefa que está executando e carregar os registradores com os valores da nova tarefa
- Em geral, esta métrica não inclui o tempo necessário para decidir qual tarefa vai executar
  - Depende do algoritmo de escalonamento utilizado
- O tempo de chaveamento de contexto soma-se ao tempo máximo de execução de cada tarefa, em um cenário de pior caso
  - Quanto menor, melhor

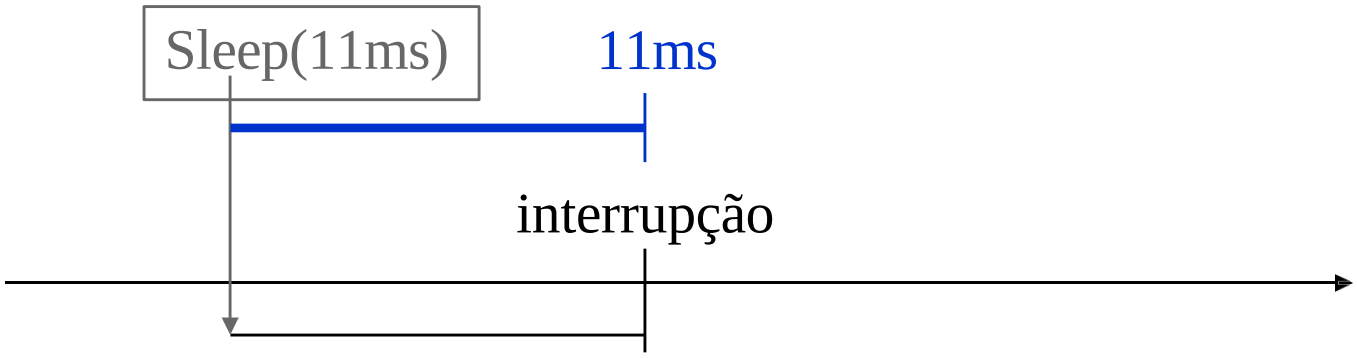
Diferenças Construtivas entre SOPG e SOTR 8/14

- **Emprego de Temporizadores com Alta Resolução**

- Tarefas da aplicação armam temporizações ao final das quais uma ação ocorre
- Essa ação pode ser assíncrona (por exemplo, o envio de um sinal Unix)
- Ou síncrona (por exemplo, a liberação da tarefa após uma chamada "sleep()")
- Temporizadores são utilizados na implementação de mecanismos de *time-out*, *watch-dog*, e também tarefas periódicas, entre outros usos
- Um SOTR utiliza temporizadores de alta resolução, baseados em interrupções aperiódicas
  - Relógio de hardware não gera interrupções periódicas mas é programado para gerar uma interrupção no próximo instante de interesse
- Supondo ser o final do "sleep()" o próximo instante de interesse, o temporizador em hardware seria programado para gerar uma interrupção exatamente no momento esperado pela tarefa em questão

Diferenças Construtivas entre SOPG e SOTR 9/14





Diferenças Construtivas entre SOPG e SOTR 10/14

- **Comportamento das Chamadas de Sistema no Pior Caso**

- Aplicações de tempo real são beneficiadas quando o código que implementa as chamadas de sistema apresenta bom comportamento também no pior caso
- Na construção de um SOTR devem ser evitados algoritmos que apresentam excelente comportamento médio
  - Porém um péssimo comportamento de pior caso

Diferenças Construtivas entre SOPG e SOTR 11/14

- **Preempção de Tarefa Executando Código do Kernel**

- Um SOTR deve ser preemptivo
  - Ainda que em determinados momentos, interrupções precisem ser desabilitadas e a preempção desligada por pequenos intervalos de tempo

Diferenças Construtivas entre SOPG e SOTR 12/14

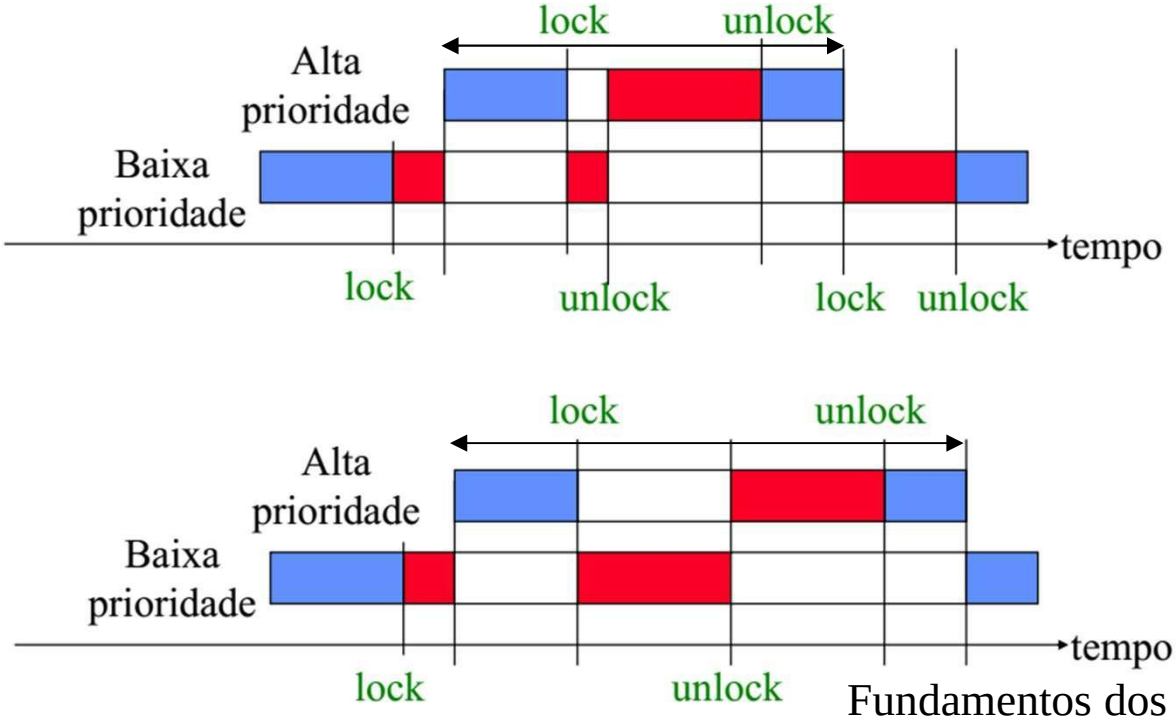
- **Mecanismos de Sincronização Apropriados**

- Aplicações de tempo real são majoritariamente construídas como programas concorrentes
- Threads precisam acessar variáveis compartilhadas
- Precisam de mecanismos de sincronização
- Um SOTR deve oferecer para as tarefas de aplicação mecanismos de sincronização apropriados para tempo real

Diferenças Construtivas entre SOPG e SOTR 13/14

- **Granularidade das Seções Críticas dentro do Kernel**

- Manter uma granularidade fina para as seções críticas dentro do kernel
  - Aumenta a complexidade do código
  - Mas reduz o tempo que uma tarefa de alta prioridade precisa esperar até que a tarefa de baixa prioridade libere a estrutura de dados compartilhada



Diferenças Construtivas entre SOPG e SOTR 14/14

- **Gerência de Recursos em Geral**

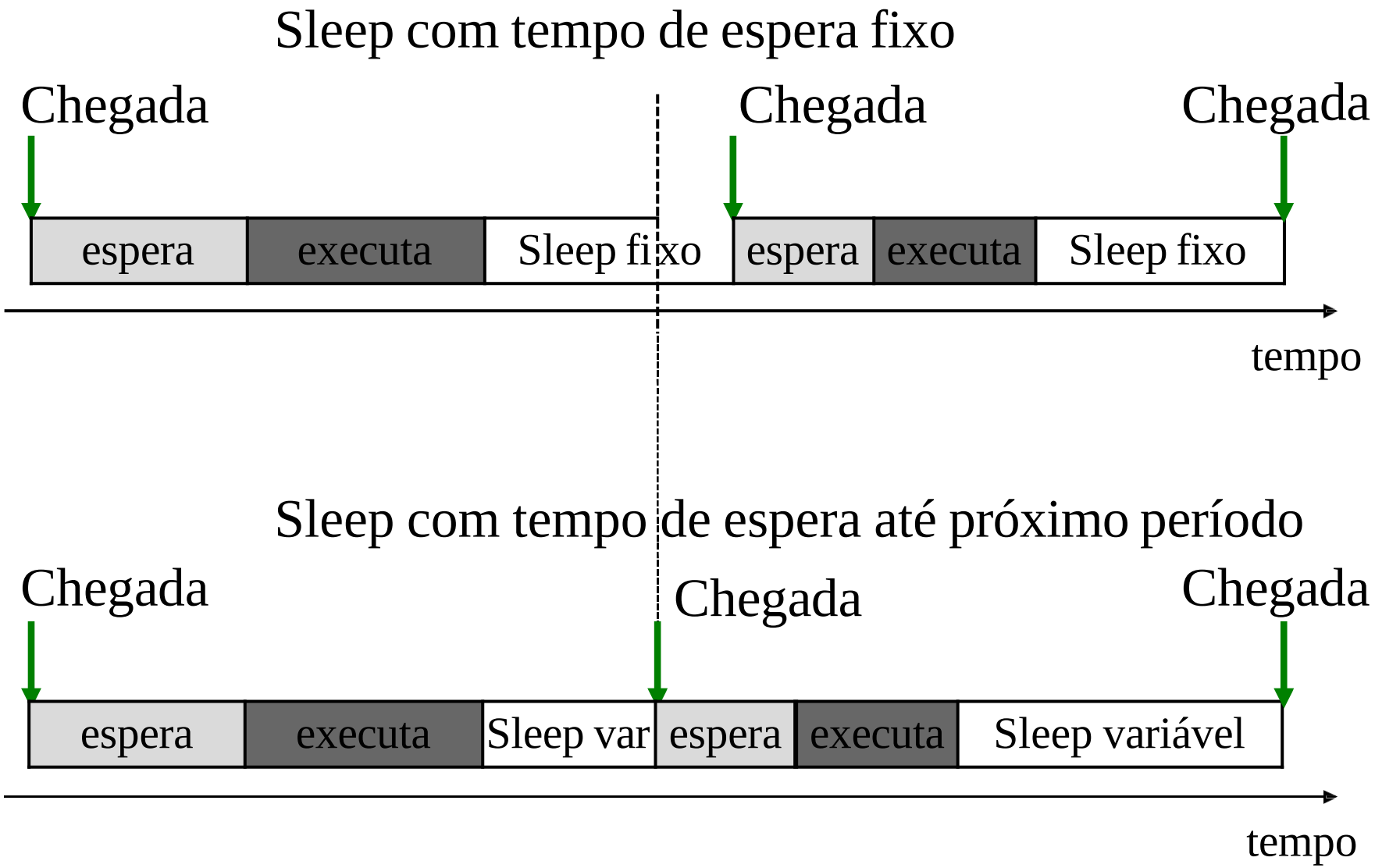
- Todos os sistemas operacionais desenvolvidos ou adaptados para tempo real mostram grande preocupação com a divisão do tempo do processador entre as tarefas
- Memória, periféricos, controladores, servidores também deveriam ser escalonados visando atender os requisitos temporais da aplicação
- Muitos sistemas ignoram isto
  - Tratam os demais recursos da mesma maneira empregada por um SOPG
- Todas as filas do sistema deveriam respeitar as prioridades das tarefas
  - Por exemplo, as requisições de ethernet deveriam ser ordenadas conforme a prioridade e não pela ordem de chegada

Cuidados do Desenvolvedor da Aplicação 1/2

- Por melhor que seja o design do SOTR usado, cabe ainda ao desenvolvedor da aplicação uma série de cuidados
- O correto uso das facilidades do SOTR é necessário

- Talvez o principal aspecto seja a atribuição de prioridades
- Uma razão frequente para a perda de deadlines é a inclusão no código da aplicação de seções críticas longas
- Implementação de tarefa periódica
  - Usar uma chamada de sistema apropriada para isto, caso o sistema operacional em questão ofereça uma
  - Uma outra possibilidade é usar uma chamada de sistema do tipo “sleep()” onde o parâmetro não é o tempo fixo de espera mas sim o instante absoluto no futuro quando a tarefa será liberada
  - Por exemplo, a função clock_nanosleep com a opção TIMER_ABSTIME no Linux

Cuidados do Desenvolvedor da Aplicação 2/2



Microkernel Exemplo: FreeRTOS 1/5

- FreeRTOS(www.freertos.org) foi desenvolvido por Richard Barry em torno de 2003
- Mais tarde o desenvolvimento e manutenção continuaram através da empresa Real Time Engineers Ltd.
- Em 2017 a empresa Real Time Engineers Ltd. passou o controle do projeto FreeRTOS para a Amazon Web Services (AWS – aws.amazon.com)
- Existem portes do FreeRTOS para mais de 30 arquiteturas de processadores
- FreeRTOS foi criado para o mercado de aplicações embutidas ou embarcadas (*embedded*) de tempo real de pequeno porte
  - Podem incluir tarefas com diferentes níveis de criticalidade
- FreeRTOS é um microkernel que permite a execução de múltiplas threads (chamadas de tarefas na terminologia do FreeRTOS)
  - Escalonamento baseado em prioridades preemptivas
  - Diversos mecanismos de sincronização entre threads

Exemplo

Tarefa Periódica no FreeRTOS

Kernel Exemplo: Linux PREEMPT_RT 1/5

- Nos últimos 20 anos surgiram muitas variantes de tempo real do Linux
- Esta seção trata especificamente do PREEMPT_RT
- Patch aplicado no kernel do Linux
- Reduz os segmentos de código do kernel onde preempções não são possíveis
- Move grande parte do código dos tratadores de interrupções para threads do kernel
- Uma excelente fonte de informações atualizadas sobre o PREEMPT_RT é o **"Real-Time Summit"**, organizado pela Linux Foundation (www.linuxfoundation.org)
- Também existe uma página sobre a história do PREEMPT_RT no site da Linux Foundation

Kernel Exemplo: Linux PREEMPT_RT 5/5

- O patch PREEMPT_RT transforma o Linux em um kernel mais apropriado para tempo real
- A implementação das temporizações é alterada, permitindo que temporizadores Posix no espaço de usuário operem com alta resolução
- Conversão de parte do código dos tratadores de interrupção em threads de kernel (threaded interrupt handlers), as quais são preemptáveis
- Alterações nos mecanismos de sincronização dentro do kernel
- Tornou preemptáveis seções críticas dentro do kernel
- Spin-locks são usados para proteger seções críticas dentro do kernel
  - Quando a seção crítica é longa e/ou muito disputada, spin-lock gera atrasos
  - Maioria dos spin-locks convertidos em rt_mutex (sleeping spinlock)

Considerações Finais 1/6

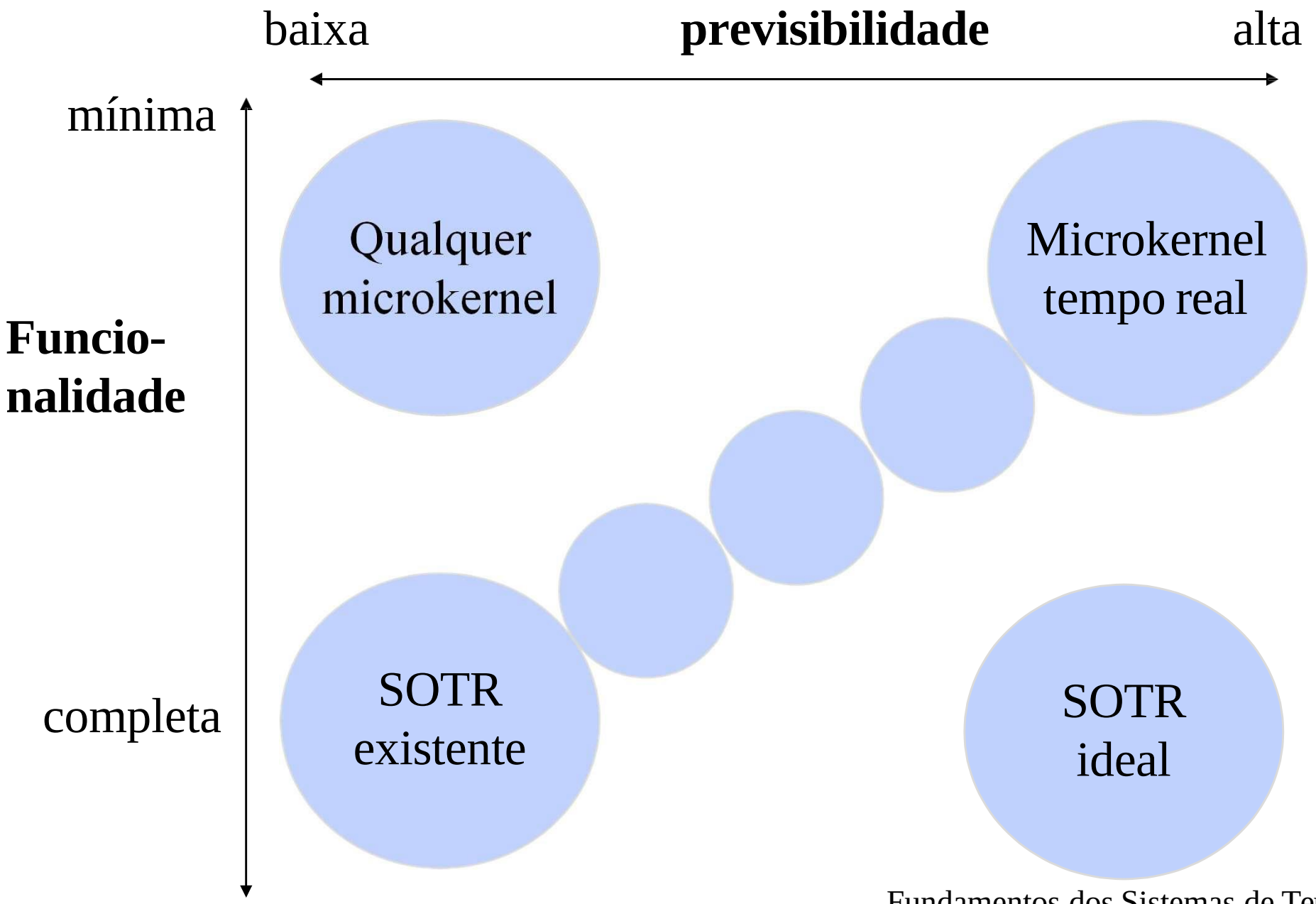
- A escolha de um SOTR não é trabalho simples
- Fatores a considerar:
  - Diferentes SOTR possuem diferentes abordagens de escalonamento
  - Desenvolvedores de SOTR publicam métricas diferentes
  - Desenvolvedores de SOTR não publicam todas as métricas e todos os dados
  - As métricas fornecidas foram obtidas em plataformas diferentes
  - O conjunto de ferramentas para desenvolvimento de aplicações que é suportado varia
  - Ferramentas para monitoração e depuração das aplicações variam
  - As linguagens de programação suportadas em cada SOTR são diferentes
  - O conjunto de periféricos suportados por cada SOTR varia
  - O conjunto de plataformas de hardware suportados varia em função do SOTR
  - Cada SOTR possui um esquema próprio para a incorporação de novos tratadores de dispositivos (device-drivers)
  - Diferentes SOTR possuem diferentes níveis de conformidade com os padrões
  - A política de licenciamento e o custo associadovariam

Considerações Finais 5/6

- Uma busca rápida na Internet mostrará centenas de SOTR
  - Como kernel ou microkernel
  - Comercial ou software livre
  - Para tempo real crítico ou não

- Em uma dimensão temos a funcionalidade oferecida pelo SOTR
- Na outra dimensão, temos o determinismo temporal
- Funcionalidade mínima com boa previsibilidade temporal é possível
  - Microkernel de tempo real
- Um kernel completo vai oferecer muito menos determinismo que um microkernel

Considerações Finais 6/6



Resumo

- Introdução
- Aspectos Funcionais dos SOTR
- Aspectos Temporais dos SOTR
- SOTR: O Ideal Impossível
- SOTR: O Ideal Possível porém Inexistente
- SOTR: ARealidade
- Diferenças Construtivas entre SOPG e SOTR
- Cuidados do Desenvolvedor da Aplicação
- Microkernel Exemplo: FreeRTOS
- Kernel Exemplo: Linux PREEMPT_RT
- Considerações Finais