Sistemas de Tempo Real

Disciplina de Sistemas Embarcados

Prof. Renato Sampaio

Sistemas de Tempo Real

O que é tempo real?

"Um sistema de tempo-real é aquele em que a corretude das computações não depende somente da corretude lógica mas também no tempo em que o resultado é produzido. Se as restrições de tempo não são atendidas, é considerada uma falha do sistema."

(http://www.faqs.org/faqs/realtime-computing/faq/)

Tempo Real

- Conceitos de tempo-real
  - Não é ser rápido
  - De fato, muitas aplicações que exigem tempo-real não necessitam de sistemas de alto desempenho (performance).
  - O tempo-real tem relação com garantias de execução e não velocidade de execução em si.
  - É o ambiente que impõe a necessidade de processamento (Interrupções).

Tempo Real

- **Conceito de resposta em tempo-real:**
  - Tempo de resposta fixo;
  - A resposta é determinística em relação ao estímulo;
- **Conceito de resposta instantânea:**
  - Tempo de resposta nulo;
  - A resposta é imediata em relação ao estímulo;

tempo real ≠ instantaneidade.

Tempo Real

- **Sistema computacional em tempo real:**
  - A resposta não depende somente de lógica mas também do instante em que os resultados são produzidos;
  - Restrição temporal deve ser fortemente considerada no projeto do sistema computacional;
  - O não cumprimento das restrições temporais, em geral, implica em falhas catastróficas;

Tempo Real

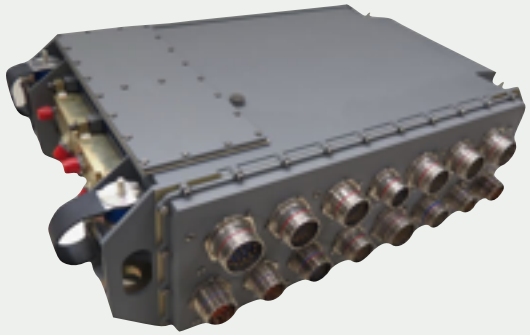
- **Sistema computacional em tempo real:**
  - Deve reagir a estímulos do ambiente controlado com um intervalo ditado pelo ambiente;
  - Instante de tempo em que um resultado deve ser produzido constitui o ***deadline***;
    - ***Soft real-time:*** tolerância no deadline;
    - ***Hard real-time***: não há tolerância para o atraso.

Hard real-time

- Fortemente baseado em deadlines;
- Tolerância “zero” a atrasos;
- Não cumprimento de deadlines implica em falhas catastróficas;
- Tempo de resposta é determinístico inclusive sob condição de sobrecarga;

Hard real-time

- Exemplos:
  - Sistema ABS;
  - Controlador de Motor - Full Authority Digital Engine Controller (FADEC)
  - Fly-by-wire Flight Control Computer (FCC)
  - Sistema de Defesa;
  - Controlador de robô manipulador, bípede, etc.



Soft real-time

- Também classificado como ***firm real-time***
- Também baseados em *deadlines*;
- Apresentam certa tolerância a atrasos;
- O não cumprimento de *deadlines* não implica em falhas catastróficas;
- Desempenho é degradado sob condição de sobrecarga;

Soft real-time

- Exemplos:
  - Tela do celular/tablet (rastreamento do movimento do dedo)
  - TV Digital
  - Tocador de música

Latência

- ***Interrupt latency .*** Tempo entre o disparo de um **evento** (interrupção) e o momento em que a aplicação o processa. (**Pior caso**)
- O que gera este atraso no tratamento da interrupção?
    - comunicação do barramento
    - operações de DMA
    - perdas de leitura na cache (*cache misses*)
    - mascaramento da interrupção

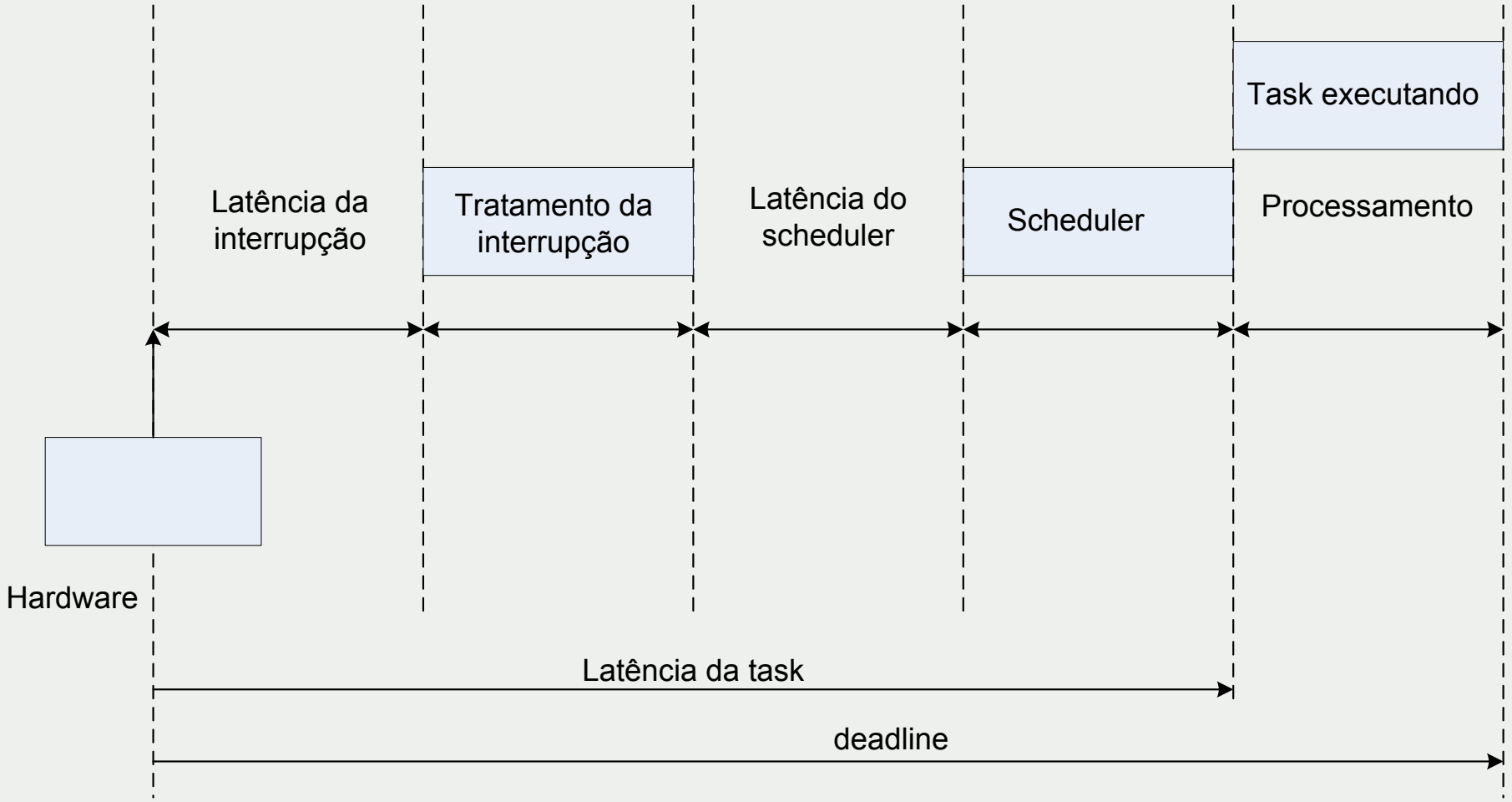
Latência

- Além disso, o gerenciador de interrupções pode colocar uma tarefa em fila para ser tratada posteriormente (outra interrupção), porém quando o processo for suspenso, não se sabe quando irá retornar à execução na CPU.
- Este é o chamado ***dispatch latency*** ou ***scheduling latency***.
- Neste caso o kernel acaba de trocar uma tarefa de alta prioridade por uma outra de baixa prioridade e não pode parar imediatamente até que chegue a um estado em que possa retornar de maneira segura para o escalonador

Deadline

- Intervalo de tempo entre a observação de um estímulo e a reação ao mesmo;
- Cumprimento fortemente associado ao estado do sistema computacional;
- *Hard real-time* deve garantir cumprimento de *deadline* ainda que esteja sob sobrecarga;

Tratamento de Interrupções



Requisitos do Sistema de Tempo-Real

- Previsibilidade;
- Confiabilidade;
- Segurança;
- Manutenção;
- Disponibilidade;

Hard vs Soft real-time

Característica	*Hard real-time*	*Soft real-time*
Tempo de resposta	Determinístico	Estocástico
Desempenho em carregamento de pico	Previsível	Degradado
Segurança	Geralmente crítico	Não crítico
Integridade de dados	Curto intervalo	Longo intervalo
Detecção de erros	Autônoma	Assistida pelo usuário

Estímulo

- Classificação do método de observação do ambiente:
  - ***Time-triggered***
  - ***Event-triggered***

Estímulo

- ***Time-triggered***
  - Observação ocorre de forma **síncrona**;
  - Realizada sempre em que há uma notificação do *real-time clock*;
  - Observação é feita independente da mudança de estado do ambiente observado;

Estímulo

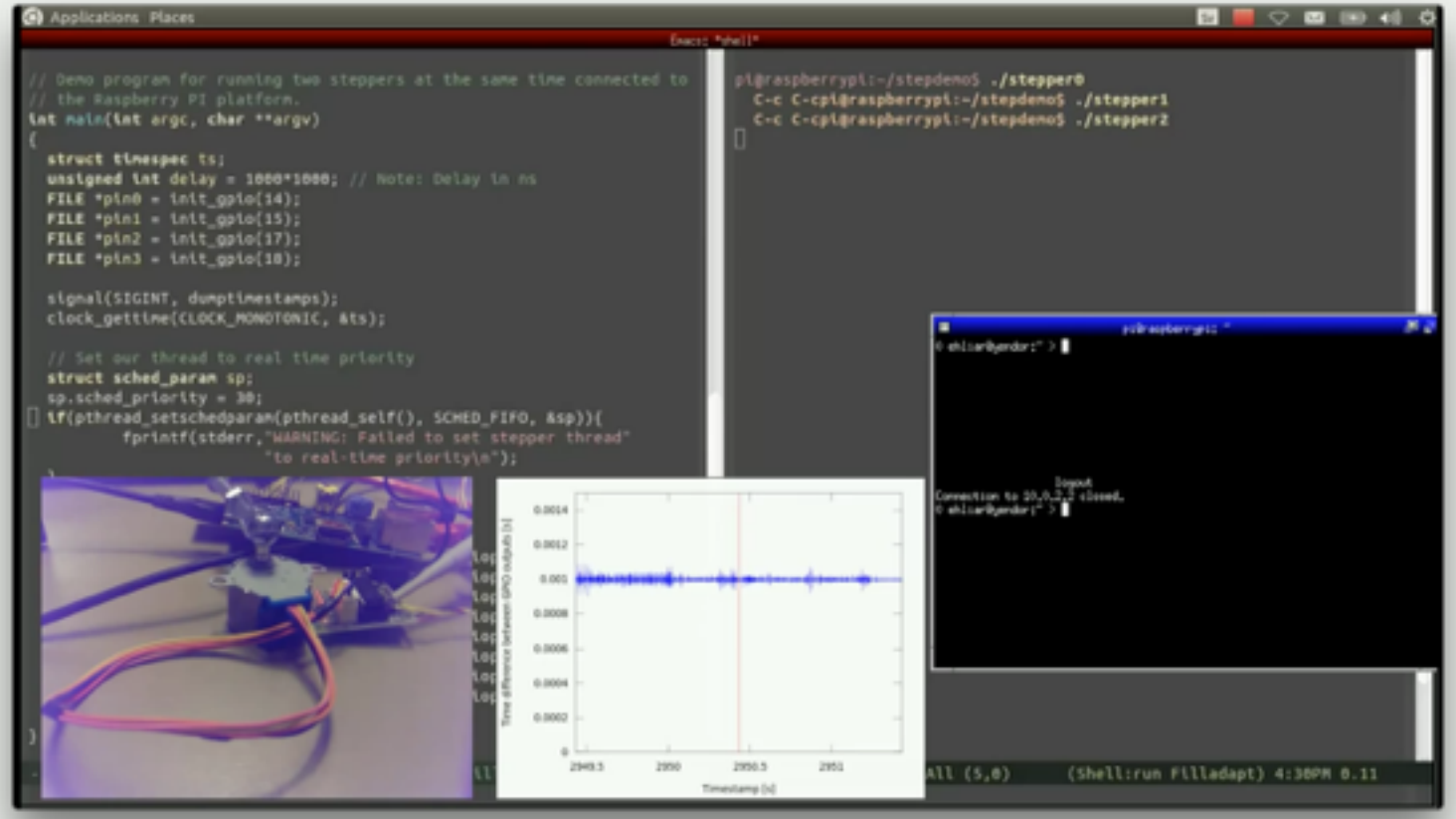
- ***Event-triggered***
  - Ambiente controlado é observado de forma **assíncrona**;
  - O início da observação depende somente do ambiente observado;
  - Observação é feita somente com a mudança de estado do ambiente observado;

Preciso de um Sistema de Tempo-Real?

- Meus requisitos incluem *deadlines*? Minha aplicação seria válida caso complete as tarefas de maneira atrasada? Estes atrasos causariam algum risco de segurança a alguém? Haveria alguma degradação no serviço?
- Algum dispositivo externo que interage com minha aplicação exige algum requisito relacionado ao tempo de resposta?
- Há alguma parte da minha aplicação que deveria ter prioridade de tarefas acima dos serviços padrão do próprio SO (rede, sistema de arquivos, etc)?
- Preciso de uma granularidade mais refinada para controlar os *delays* e os *timeouts*?

Exemplo

Real Time Programming in Linux - Controlling a stepper connected to the Raspberry Pi



https://www.youtube.com/watch?v=uIXkvz1-weQ

Sistemas Operacionais de Tempo Real

- Em geral, existem dois tipos de Sistemas Operacionais:
  - Sistemas Operacionais de Propósito Geral (**GPOS**);
  - Sistemas Operacionais em Tempo Real (**RTOS**);

Sistemas Operacionais de Tempo Real

- **Sistemas Operacionais de Propósito Geral (GPOS):**
  - Orientado a execução de tarefas (não RT);
  - Orientado ao desempenho médio;
  - Não garante o cumprimento de *deadlines*;
  - Cumprimento de *deadlines* é fortemente degradado com sobrecarga sobre o sistema;

Sistemas Operacionais de Tempo Real

- **Sistemas Operacionais em Tempo Real (RTOS):**
  - Orientado a execução de tarefas (em RT);
  - Orientado ao desempenho particular;
  - Garante o cumprimento de *deadlines*;
  - O cumprimento de *deadlines* é garantido mesmo sob condição de sobrecarga (*hard real-time*);

Sistemas Operacionais de Tempo Real

- **Similaridades entre RTOS e GPOS:**
  - Devem fornecer suporte a multitarefas;
  - Deve gerenciar recursos de hardware/software;
  - Fornece serviços fundamentais para a aplicação;
  - Suporte a abstração de hardware para aplicações desenvolvidas;

Sistemas Operacionais de Tempo Real

- **Diferenças de um RTOS em relação a um GPOS:**
  - Maior confiabilidade em aplicações embarcadas;
  - Capacidade de modificar taxa de processamento para cumprir necessidades;
  - Melhor desempenho de tarefa;
  - Consumo de memória otimizado;
  - Política de escalonamento otimizadas;
  - Melhor portabilidade para diferentes plataformas de hardware;

Sistemas Operacionais de Tempo Real

- **Características de um GPOS:**
  - Predominantemente em PCs, Workstation e Mainframe;
  - Em geral, requerem uma grande quantidade de memória;
  - Suporte a extensão de memória (swap);
  - Não são tão adequados aos sistemas embarcados (restrição de memória e alto desempenho);

Sistemas Operacionais de Tempo Real

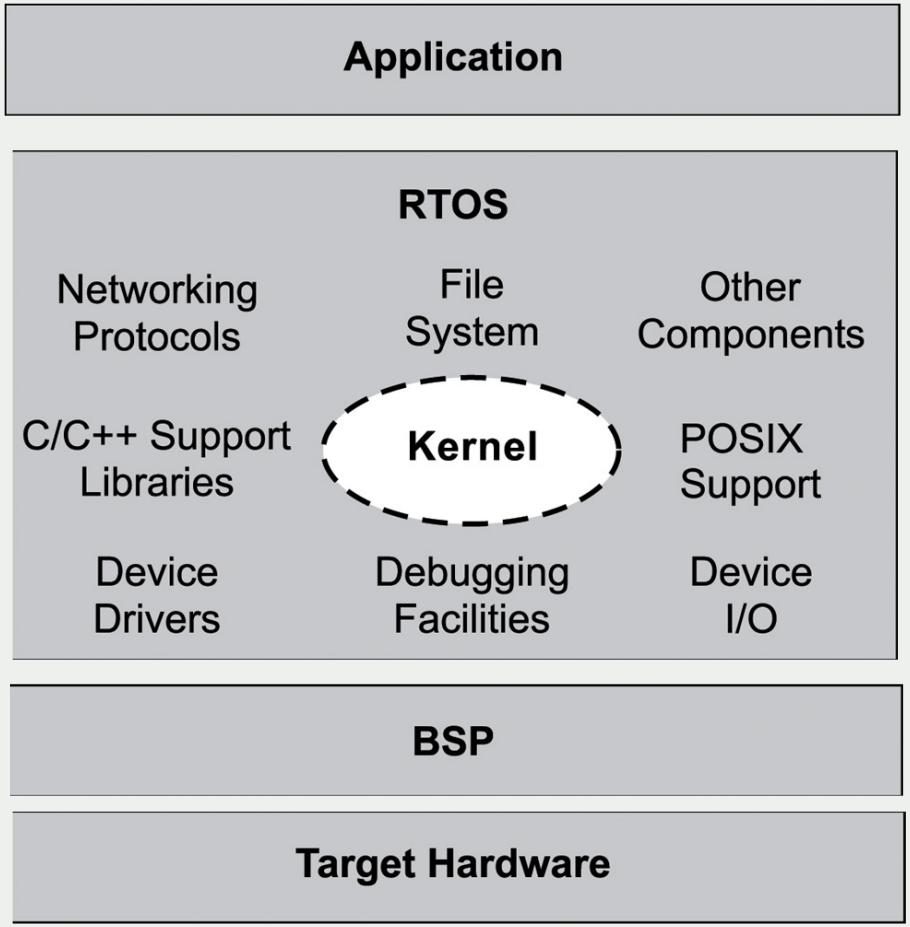
- **Características de um RTOS:**
  - Predominantemente em sistemas embarcados;
  - Compactos, confiáveis e escalonáveis;
  - Possuem melhor desempenho em sistemas em tempo real;

Sistemas Operacionais de Tempo Real

- **Funcionalidades básicas em um RTOS:**
  - Escalona a execução em um tempo hábil;
  - Suporte ao gerenciamento de recurso de sistemas;
  - Suporte ao desenvolvimento de aplicações;
- **Componentes básicos de um RTOS:**
  - Kernel RT;
  - Módulos (device drivers, sistemas de arquivo, protocolo de rede, dentre outros);

Sistemas Operacionais de Tempo Real

Camadas de software encontrada em sistemas embarcados

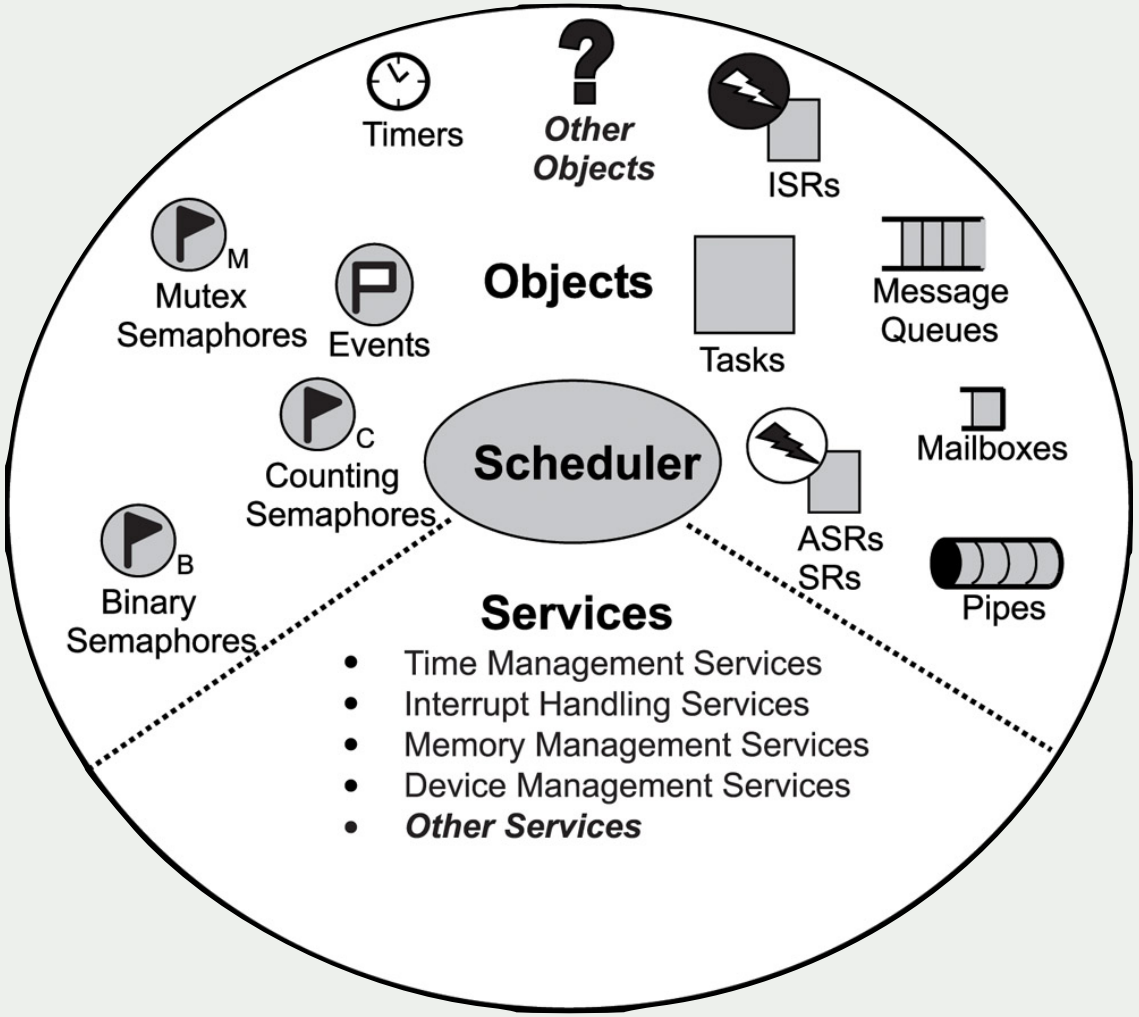


Sistemas Operacionais de Tempo Real

- **Principais componentes de um RTOS:**
  - Escalonador (Scheduler);
  - Serviços de gerenciamento de tarefas (*tasks*);
  - Serviços em geral;

Sistemas Operacionais de Tempo Real

Componentes de um kernel RTOS



Sistemas Operacionais de Tempo Real

- **Escalonador (Scheduler):**
  - Principal componente do kernel;
  - Determina a seqüência de execução das tarefas;
  - Seqüência de execução determinada com base em uma política de escalonamento;
- **Quem pode ser escalonado?**
  - Entidades que podem competir por um tempo de execução (Ex.: tasks, threads, processos);

Sistemas Operacionais de Tempo Real

- **Políticas de escalonamento:**
  - FIFO (First In First Out);
  - LIFO (Last In First Out);
  - Round-Robin;
  - Dentre outras;

Sistemas Operacionais de Tempo Real

- **Serviço de gerenciamento de tasks:**
  - Capacidade do OS em manipular múltiplas atividades considerando um conjunto de *deadlines*;
  - Conjunto de serviços que permitem a criação, manipulação e execução de *tasks*;
  - Gerencia a execução de *tasks* para evitar *deadlocks*

Sistemas Operacionais de Tempo Real

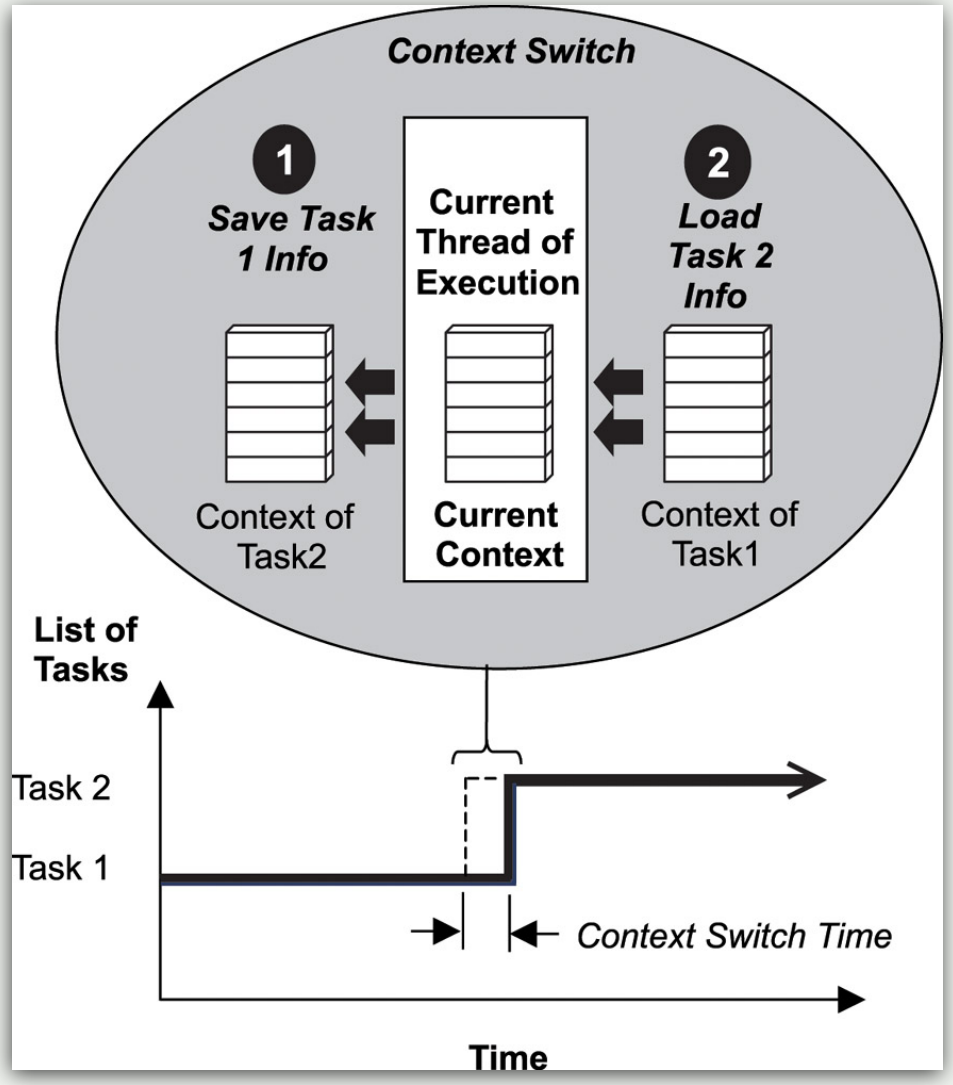
- **Serviços gerais:**
  - Operações que o kernel desempenha sobre um objeto;
  - Exemplo de serviços gerais:
  - Temporalização;
  - Tratamento de interrupções;
  - Gerenciamento de recursos;
  - System call;
  - Dentre outros;

Sistemas Operacionais de Tempo Real

- **Conceito de troca de contexto:**
  - Múltiplas *tarefas* disputam por tempo de CPU;
  - Entretanto, somente uma *tarefa* pode ser executada no tempo;
  - O escalonador deve exercer a troca de contexto para permitir que *deadlines* sejam cumpridos;
  - Cada *tarefa* tem seu próprio contexto, que é o estado dos registros da CPU requeridos cada vez que é escalonada para executar.;
  - A troca de contexto ocorre quando o escalonador troca uma tarefa em execução por uma outra;

Sistemas Operacionais de Tempo Real

- **Troca de contexto:**



Sistemas Operacionais de Tempo Real

- **Etapas da troca de contexto:**
  - O kernel salva as informações de *task1* em sua TCB;
  - Kernel carrega as informações TCB da *task2* que se torna a *task* corrente (em execução);
  - Kernel congela o contexto da *task1* enquanto a *task2* é executada
  - Kernel retorna a execução da *task1* onde ela parou (exatamente antes da troca de contexto);

Bibliografia

- Kopetz, Hermann; Real-time Systems: Design Principles for Distributed Embedded Applications, 2nd Ed. Springer, 2011.