Implementação de Tarefas em Kernel Completo

Fundamentos dos Sistemas de Tempo Real

2ª Edição

Rômulo Silva de Oliveira

Edição do Autor, 2020

www.romulosilvadeoliveira.eng.br/livrotemporeal

Como tarefas são implementadas em sistemas operacionais tradicionais ?

Introdução

- Cada vez mais aplicações complexas são incluídas em equipamentos e veículos
- Aplicações que incluem um grande número de tarefas de tempo real

- Muitas aplicações de tempo real, mesmo em equipamentos, precisam
  - Interface gráfica de usuário
  - Sistema de arquivos
  - Pilhas de protocolos de comunicação
  - Controle de acesso
  - Serviços providos por um sistema operacional

- Aplicações deste tipo requerem um kernel completo como sistema operacional multitarefa
  - Exemplo clássico neste caso é o Linux

O Sistema Operacional Tradicional 1/8

- O sistema operacional atua como um mediador entre as tarefas da aplicação e os recursos do sistema
  - Especialmente os periféricos
- Torna o uso do computador mais conveniente e mais eficiente

- O sistema operacional torna mais simples o trabalho ao esconder as complexidades do hardware
  - Tanto de usuários finais como dos programadores das aplicações

- Ele cria **abstrações mais convenientes** do que os recursos do hardware
  - Arquivos e diretórios são abstrações criadas a partir de espaço em disco
  - Processos e threads são criados a partir da divisão do tempo do processador
  - Portas de comunicação são criadas a partir do controlador ethernet
  - Espaços de endereçamento são criados a partir dos circuitos de memória
  - Etc

O Sistema Operacional Tradicional 2/8

- O desenvolvimento é facilitado
- O acesso aos periféricos não é mais feito acessando diretamente os registradores dos controladores de periféricos
  - Mas solicitando este acesso ao kernel
  - Que se encarrega dos detalhes operacionais do periférico em questão
- O código da aplicação pode solicitar ao kernel o envio de um pacote de dados pela porta serial
  - Sem precisar detalhar a série de comandos necessários
- Os comandos detalhados são enviados pelo kernel
  - Módulo do kernel de device-driver da porta serial

O Sistema Operacional Tradicional 3/8

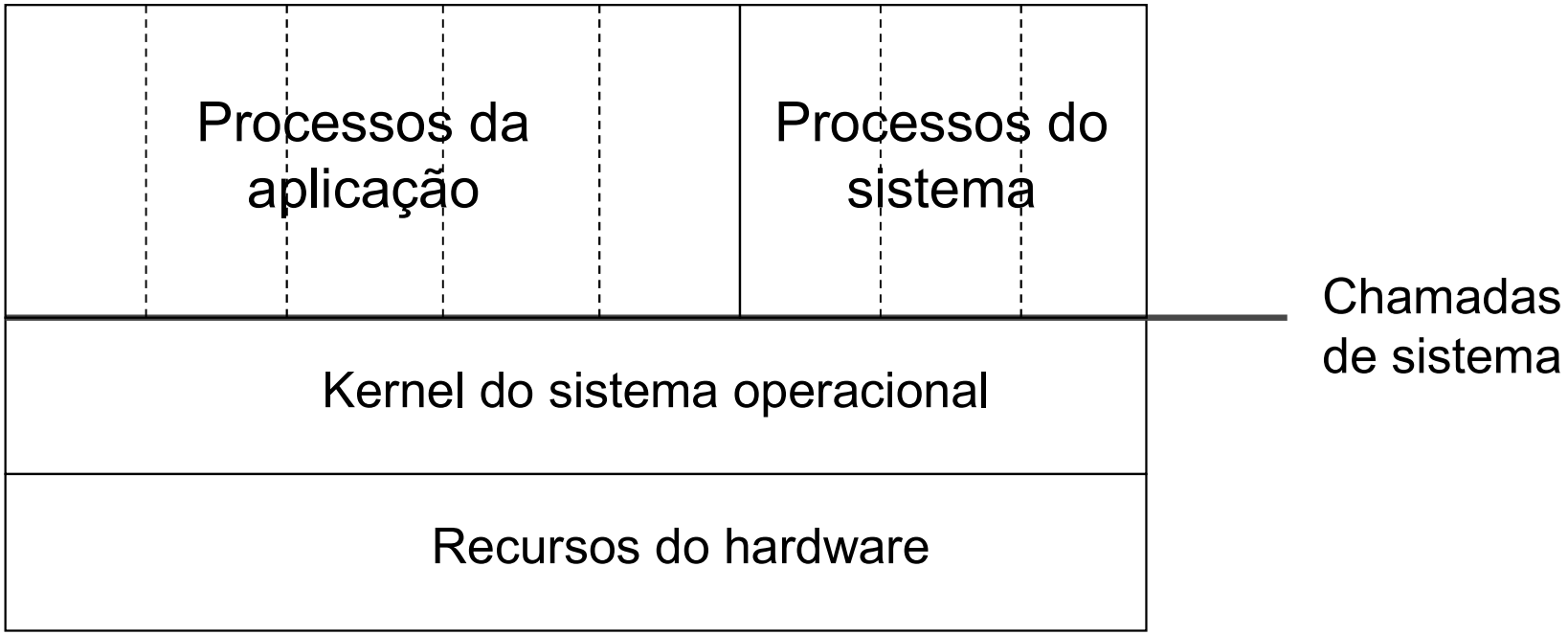
- O sistema operacional fornece uma gama de serviços
- Pode ser mais ou menos extensa, dependendo do seu tamanho
- Serviços típicos:
  - Execução de programas
  - Gerência do processador
  - Gerência de memória sofisticada
  - Sistemas de arquivos
  - Gerência de periféricos
  - Proteção entre processos
  - Pilhas de protocolo de comunicação
  - Controle de acesso
  - Contabilizações

O Sistema Operacional Tradicional 4/8

- O componente mais importante de um sistema operacional é o seu **kernel**
  - Núcleo ou miolo em português, mas o termo kernel está estabelecido
- Trata-se da camada de software que controla diretamente os recursos do hardware
- Usualmente os sistemas de arquivos, pilhas de protocolos de comunicação e device-drivers são implementados dentro do kernel
- Acima do kernel executa o código da aplicação
  - Também podem ser executadas atividades do sistema
  - Servidores de impressão ou interface com usuário humano
  - São chamados de programas de sistema (system programs)
- Quando o kernel oferece apenas serviços básicos de gerência do processador ele é chamado de microkernel

O Sistema Operacional Tradicional 5/8

- A **multiprogramação (*multiprogramming*)** é usada para realizar a execução aparentemente simultânea de vários programas em um único processador
  - A partir da divisão do tempo do processador e de outros recursos



O Sistema Operacional Tradicional 6/8

- O emprego de multiprogramação torna o uso do computador muito mais eficiente

- Considere que o programa em execução precise ler informações de um arquivo as quais estão em um certo setor do disco magnético
  - O tempo aproximado para ler um setor do disco fica em torno de 10ms

- Suponha que neste computador a frequência do clock seja 2GHz
  - Neste caso o período do clock é de 0,5ns
  - Se uma instrução de máquina demora 4 clocks em média para executar, teremos 2ns

- Durante um acesso ao disco é possível executar 5 milhões de instruções de máquina
- Graças à multiprogramação,
  enquanto um programa espera pelo acesso ao disco,
  outro programa utiliza o processador

O Sistema Operacional Tradicional 7/8

- Podemos entender um **processo (*process*)** como uma abstração criada pelo kernel
- A qual possui uma série de atributos
  - Espaço de endereçamento, descritores de arquivos abertos, permissões de acesso, contabilizações, etc

- Todo processo também possui pelo menos um fluxo de execução
  - Este fluxo de execução é chamado de **thread** (thread of execution)
  - Composta basicamente pelo conteúdo dos registradores do processador

- Nos sistemas operacionais mais antigos, processos possuíam apenas uma thread

- Sistemas operacionais modernos: um processo pode possuir várias threads

- Na verdade threads existem dentro dos processos, usando a memória, os arquivos e demais recursos do seu processo
  - Todas as threads de um processo compartilham os recursos deste processo

O Sistema Operacional Tradicional 8/8

- Existem bibliotecas que executam fora do kernel e são capazes de implementar o conceito de
  thread a nível de usuário (user-level thread)
  - Implementam threads no programa de aplicação sem o conhecimento do kernel

- Para o kernel trata-se de um processo normal com apenas uma thread
  - Mas os recursos são na verdade divididos entre threads
  - Que somente existem a nível de código de usuário

- Neste livro iremos considerar apenas as chamadas
  threads a nível de kernel (kernel-level threads)
  - Threads implementadas e reconhecidas pelo kernel do sistema operacional

Terminologia: Processos, Threads e Tarefas 1/3

- **Processo** é uma abstração criada pelo kernel a qual possui um conjunto de atributos
  - Espaço de endereçamento, descritores de arquivos abertos, direitos de acesso, contabilizações, etc

- Todo processo possui obrigatoriamente pelo menos uma thread
  - Podendo possuir mais de uma thread caso o kernel permita

- Um pequeno microkernel tipicamente implementa apenas threads:
  Todos os fluxos de execução da aplicação são threads
  que compartilham todos os recursos do sistema
  - Não existe sentido em pensar em processos

- Um kernel antigo não implementa threads explicitamente, apenas processos
  - Cada processo contem um único fluxo de execução (uma única thread implícita)

- Um kernel moderno implementa processos
  os quais podem incluir uma ou mais threads
  - Dentro do kernel também existem threads que executam código do kernel

Terminologia: Processos, Threads e Tarefas 2/3

- Na literatura em geral existe uma certa confusão sobre como são empregados os termos processos, threads e tarefas (tasks)

- O termo **processo** será usado para denotar uma abstração criada pelo kernel a partir da gerência dos recursos do sistema

- O termo **thread** representa um fluxo de execução, caracterizado principalmente pelo conteúdo dos registradores a cada momento
  - Todo processo tem pelo menos uma thread

- Uma **tarefa** corresponde à execução de um trecho de código que deve respeitar certos requisitos temporais
  - Como é implementada depende do sistema operacional em questão

Terminologia: Processos, Threads e Tarefas 3/3

- Na aplicação de tempo real uma tarefa pode aparecer como um conjunto de funções, uma única função, ou mesmo uma sequência de linhas de código
  - A definição do que é uma tarefa é feita no escopo da aplicação

- Tarefa: Ler o sensor de temperatura e colocar este valor na tela pode ser uma tarefa da aplicação

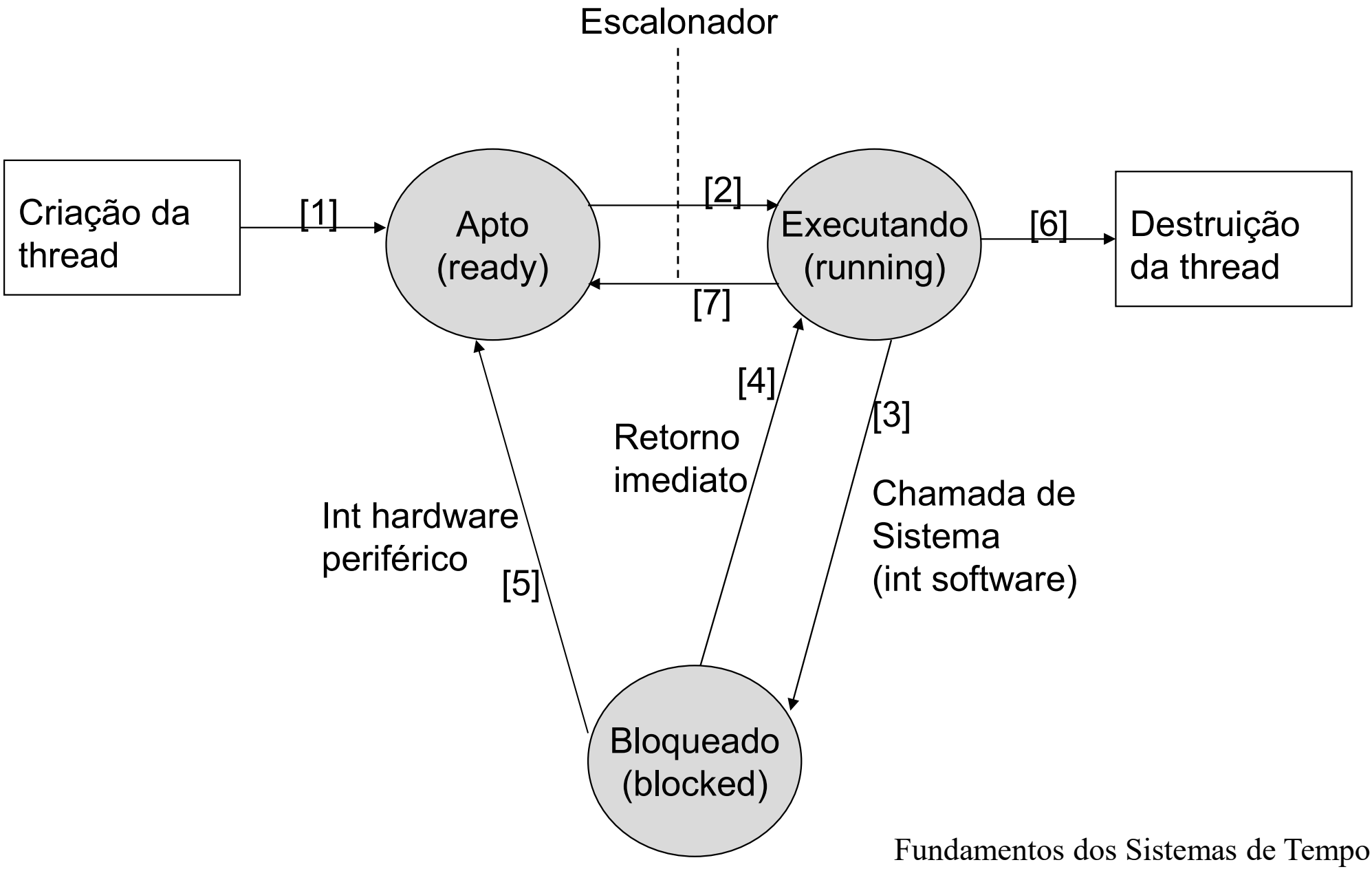
- Tarefa: Receber uma mensagem pela rede de comunicação e fechar a válvula de combustível

- Muitas vezes a tarefa da aplicação de tempo real é implementada como um processo ou thread no âmbito do sistema operacional
- Por vezes uma tarefa da aplicação de tempo real é implementada diretamente como um tratador de interrupção

Chamadas de Sistema 1/3

- Processos solicitam serviços ao kernel através de **chamadas de sistema (*system calls*)**
- Chamadas de sistema são implementadas como interrupções de software cujo tratador faz parte do kernel
- Um processo desejando ler um arquivo deve colocar em um registrador específico o código numérico da operação e gerar a interrupção de software que aciona o kernel
- Programas escritos em linguagens de alto nível usam as bibliotecas da linguagem de programação
  - As funções da biblioteca que fazem as chamadas de sistema
- Caso um processo possua 5 threads
  - Possível que cada uma delas faça uma chamada de sistema
  - Este processo teria então 5 chamadas de sistema sendo atendidas simultaneamente

Estados da Thread de um Processo 2/5



Gerência de Memória 1/5

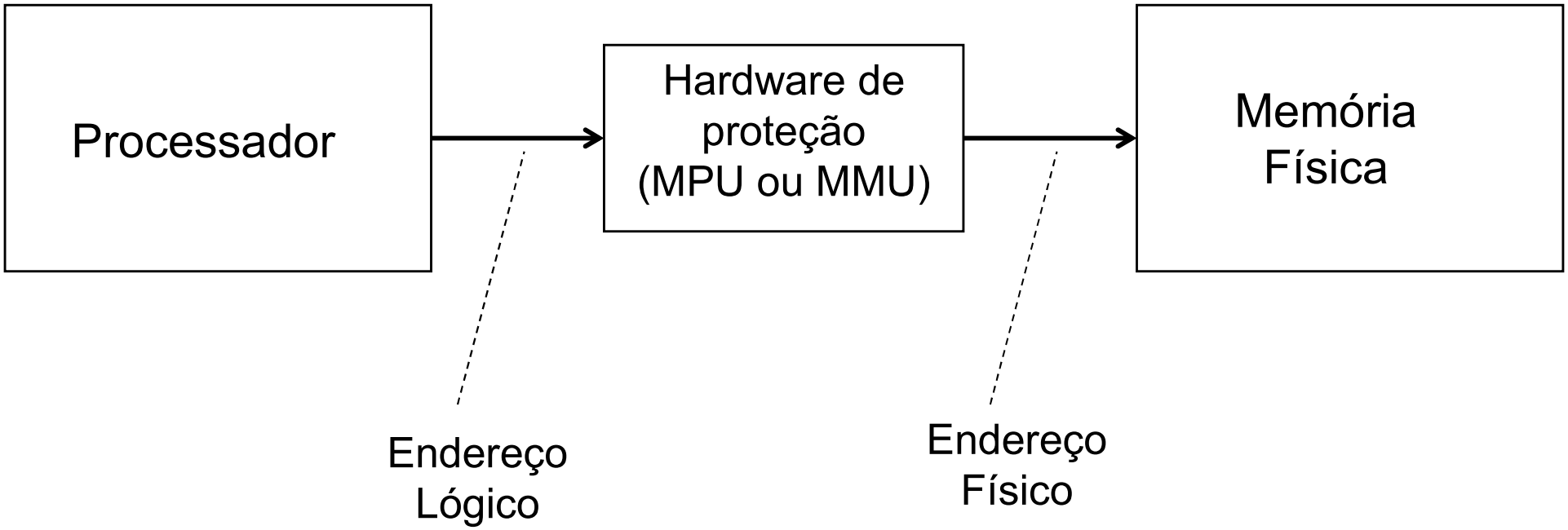
- Em um kernel de sistema operacional que implementa proteção entre processos, para que a proteção entre eles seja completa também é necessário proteger a memória

- É necessário um componente específico de hardware junto ao processador para verificar a validade de cada acesso

- Dependendo da complexidade e dos serviços providos por tal componente, diferentes gerências de memória são empregadas

- Partições variáveis usando uma Unidade de Proteção de Memória
- Paginação usando uma Unidade de Gerência de Memória

Gerência de Memória 5/5



Partições Variáveis 1/9

- Uma forma muito comum de gerência de memória em pequenos computadores é através de **partições variáveis** (*variable partitioning*)
- A memória física é dividida em **partições** (*memory partitions*)
  - Tamanhos são definidos conforme as necessidades dos processos

- Tipicamente a memória lógica de um processo corresponde a uma área contígua de memória
  - Associada com uma partição específica da memória física

- Em alguns sistemas um processo pode ter sua memória lógica dividida em várias algumas áreas contíguas
  - Número pequeno de partições
  - Por exemplo: código, pilha e variáveis globais

Partições Variáveis 7/9

- Gerência de memória precisa ser analisada com respeito às limitações
  - Resultam em subutilização da memória
- **Fragmentação interna (*internal fragmentation*)**
  - Quando mais memória do que é necessário é alocada para o processo

- Memória pode ser gerenciada em pedaços chamados parágrafos, ao invés de bytes
- Neste caso a menor região de memória física livre tem o tamanho de um parágrafo

Partições Variáveis 8/9

- Quando a memória livre total é maior do que o solicitado pelo processo
- Mas o pedido não pode ser atendido em função de como memória é gerenciada
- Ocorre **fragmentação externa (*external fragmentation*)**

- Por exemplo, memória livre total de 160 Kbytes
  - Duas regiões de 80Kbytes cada uma
  - Processo necessita uma região de memória de 110 Kbytes
  - O mesmo não poderá ser atendido
  - Temos fragmentação externa

- Dinâmica intensa de alocações e liberações de regiões de memória com tamanhos diferentes:
  fragmentação externa torna-se um grande problema

Partições Variáveis 9/9

- No caso de **sistemas embutidos ou embarcados (*embedded systems*)**
- Fragmentação externa pode não existir
- Neste tipo de sistema o computador encontra-se embutido dentro de algum equipamento ou máquina maior
- O software é responsável por controlar o equipamento maior
  - Processos são todos criados na inicialização do equipamento
  - Somente são destruídos no desligamento do equipamento
  - Como não existe destruição de processos, não existe liberação de memória
    - Não existe a formação de regiões de memória livre separadas
  - "malloc()" e "free()" usam a memória do próprio processo

Paginação 1/16

- Com partições variáveis a fragmentação externa pode tornar-se um grande problema
- Caso uma região de memória alocada precise ser aumentada
  - Somente será possível se depois dela na memória física vier uma região de memória livre
- A origem do problema está na necessidade de cada região ocupar uma área contígua de memória
- Se pudéssemos quebrar o processo em pequenos pedaços
  - Espalhar os pedaços pela memória física
  - E ainda assim o processo funcionar como se ocupasse uma área contígua
  - Então não existiria mais fragmentação externa
- A paginação faz exatamente isto

Resumo

- Introdução
- O Sistema Operacional Tradicional
- Terminologia: Processos, Threads e Tarefas
- Chamadas de Sistema
- Estados da Thread de um Processo
- Chaveamento de Contexto
- Gerência de Memória
- Partições Variáveis
- Paginação
- Outros Aspectos da Paginação
- Memória Virtual com Paginação por Demanda