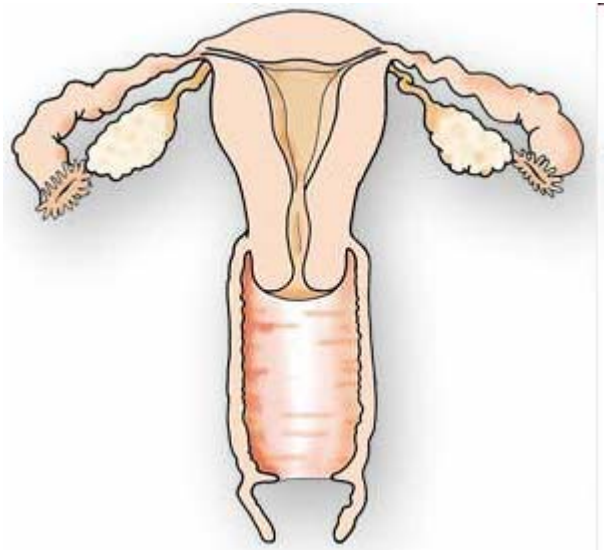
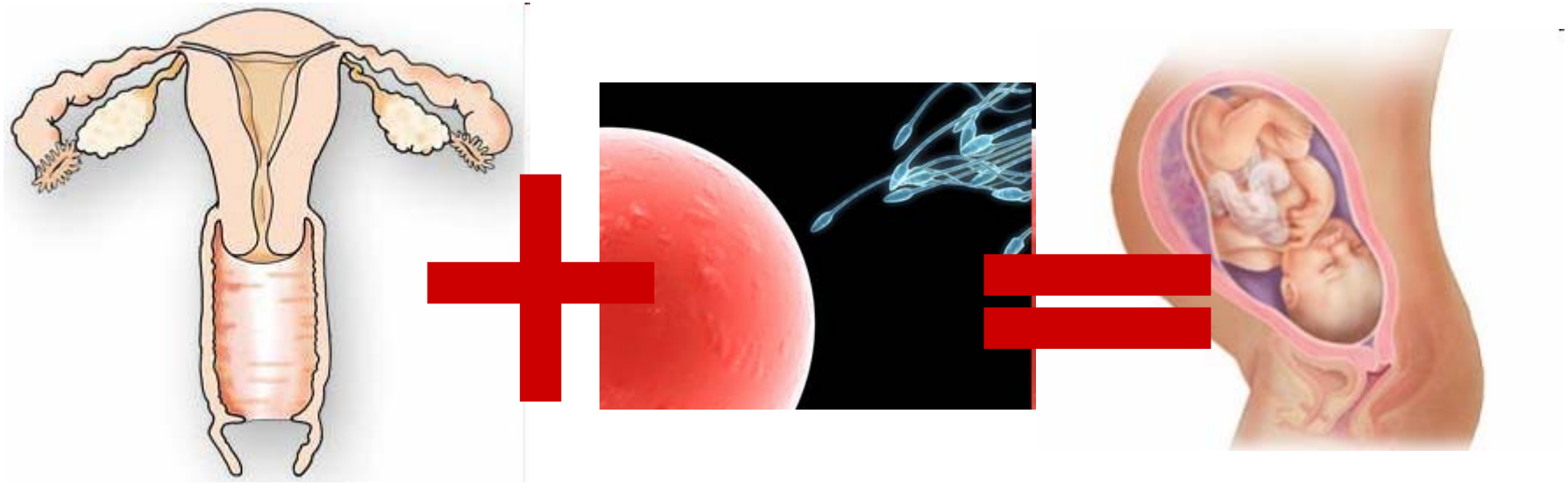
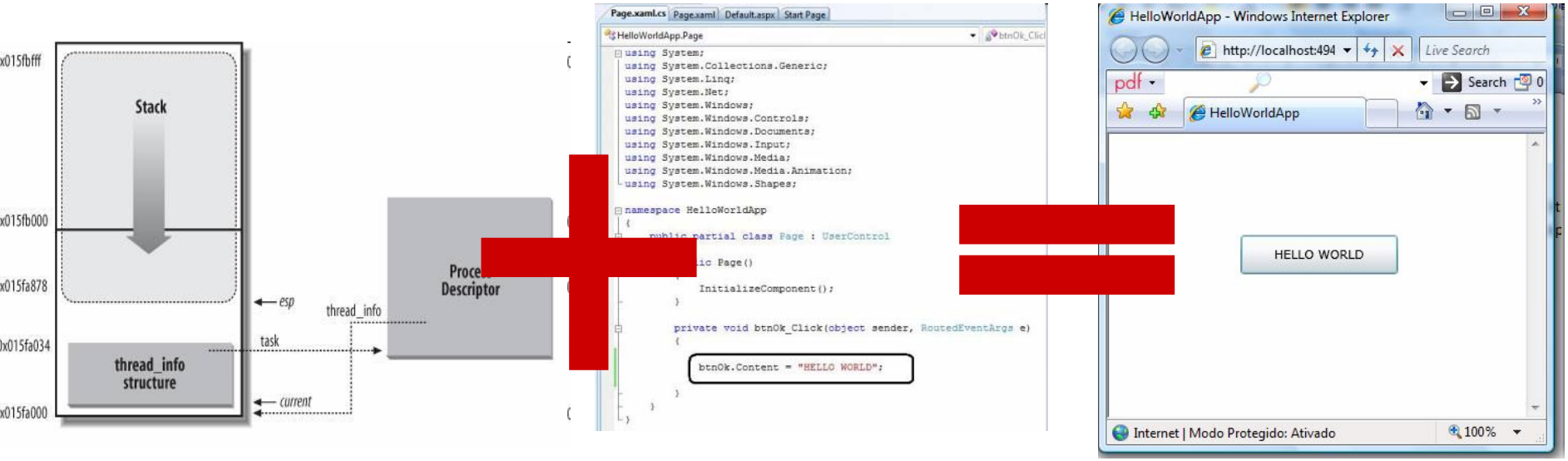
Projeto de Sistemas Distribuídos

Processos

Prof. Msc. Marcelo Iury de Sousa Oliveira
marceloiury@gmail.com
http://sites.google.com/site/marceloiury/






Stack
thread_info structure
esp
thread_info
task
current
Descriptor
Page.xaml.cs
Page.xaml
Default.aspx
Start Page
HelloWorldApp.Page
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Animation;
using System.Windows.Shapes;
namespace HelloWorldApp
public partial class Page : UserControl
InitializeComponent();
private void btnOk_Click(object sender, RoutedEventArgs e)
btnOk.Content = "HELLO WORLD";
HelloWorldApp - Windows Internet Explorer
http://localhost:494
Live Search
Search
HelloWorldApp
HELLO WORLD
Internet | Modo Protegido: Ativado
100%

Processos

- Processo engloba duas características:
  - **Representação** - a um processo é alocado um espaço de endereços virtual para representar a própria imagem (imagem do processo)
  - **Execução** - processo é um fluxo de execução ao longo de um ou mais programas
    - intercalável com outros processos

Conceito de Processos

- Um sistema operacional executa uma variedade de programas:
  - Sistemas Batch – jobs,
  - Sistemas TimeShared - programas de usuários e tarefas.
- Um processo possui:
  - Registradores,
  - Pilhas e Filas,
  - Seção de dados,
  - Seção de instruções.

Estados do Processo

- Um processo pode assumir vários estados durante o seu ciclo de vida:
  - novo: o processo está sendo criado.
  - executando: instruções estão sendo executadas.
  - bloqueado: o processo está aguardando algum evento ou resposta de uma operação de I/O.
  - pronto: o processo está aguardando para ser processado pela CPU.
  - encerrado: o processo finalizou suas execuções.

Diagrama de Estados do Processo

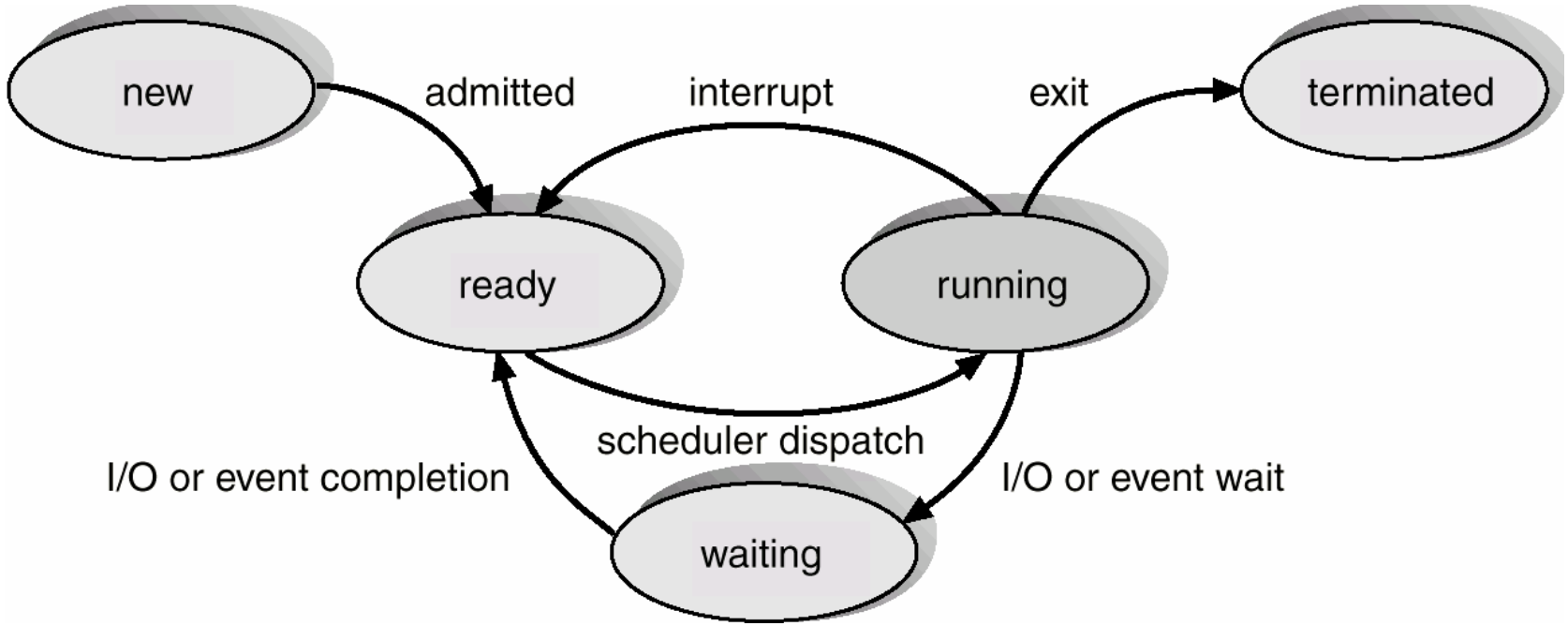
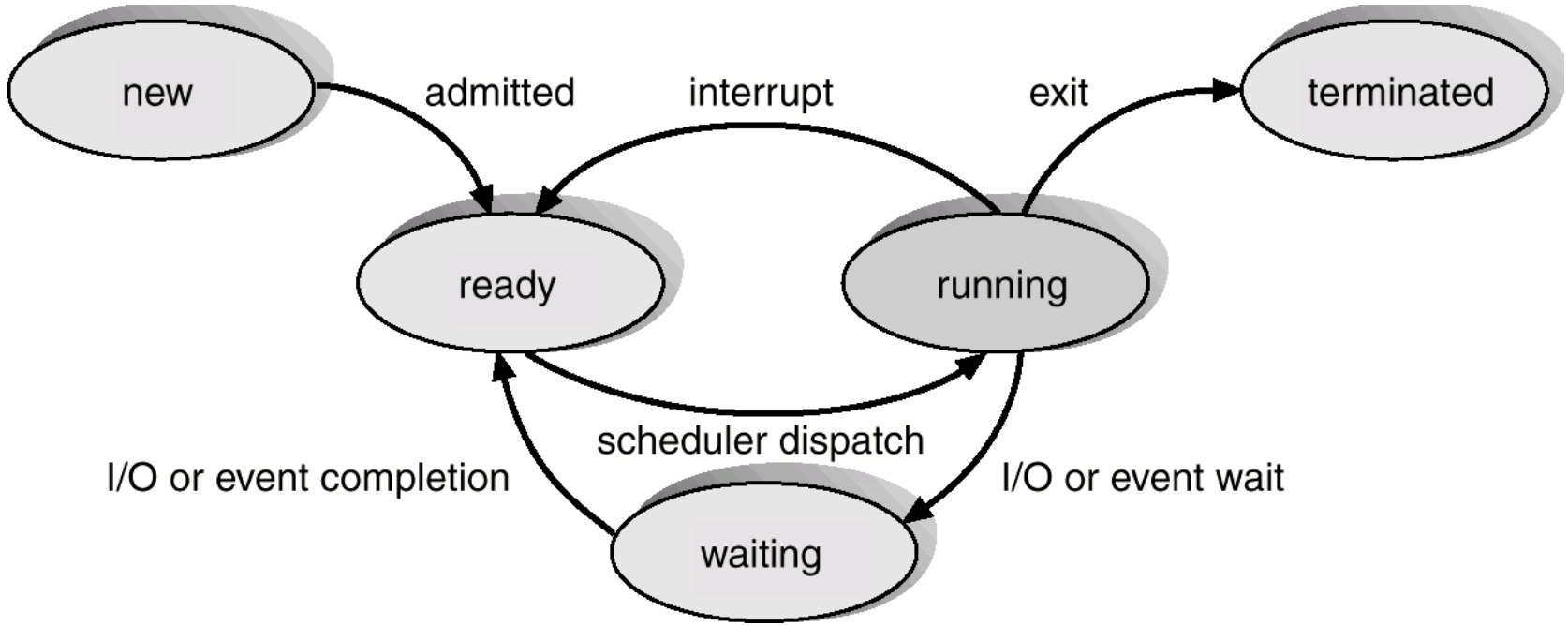


Diagrama de Estados do Processo



Pergunta que não quer calar: em um PC comum, temos vários processos e alguns poucos processadores e recursos de entrada e saída. Como fazemos para esses processos coexistirem?

Diagrama de Estados do Processo

Escalonamento!!!

Pergunta que não quer calar: em um PC comum, temos vários processos e alguns poucos processadores e recursos de entrada e saída. Como fazemos para esses processos coexistirem?

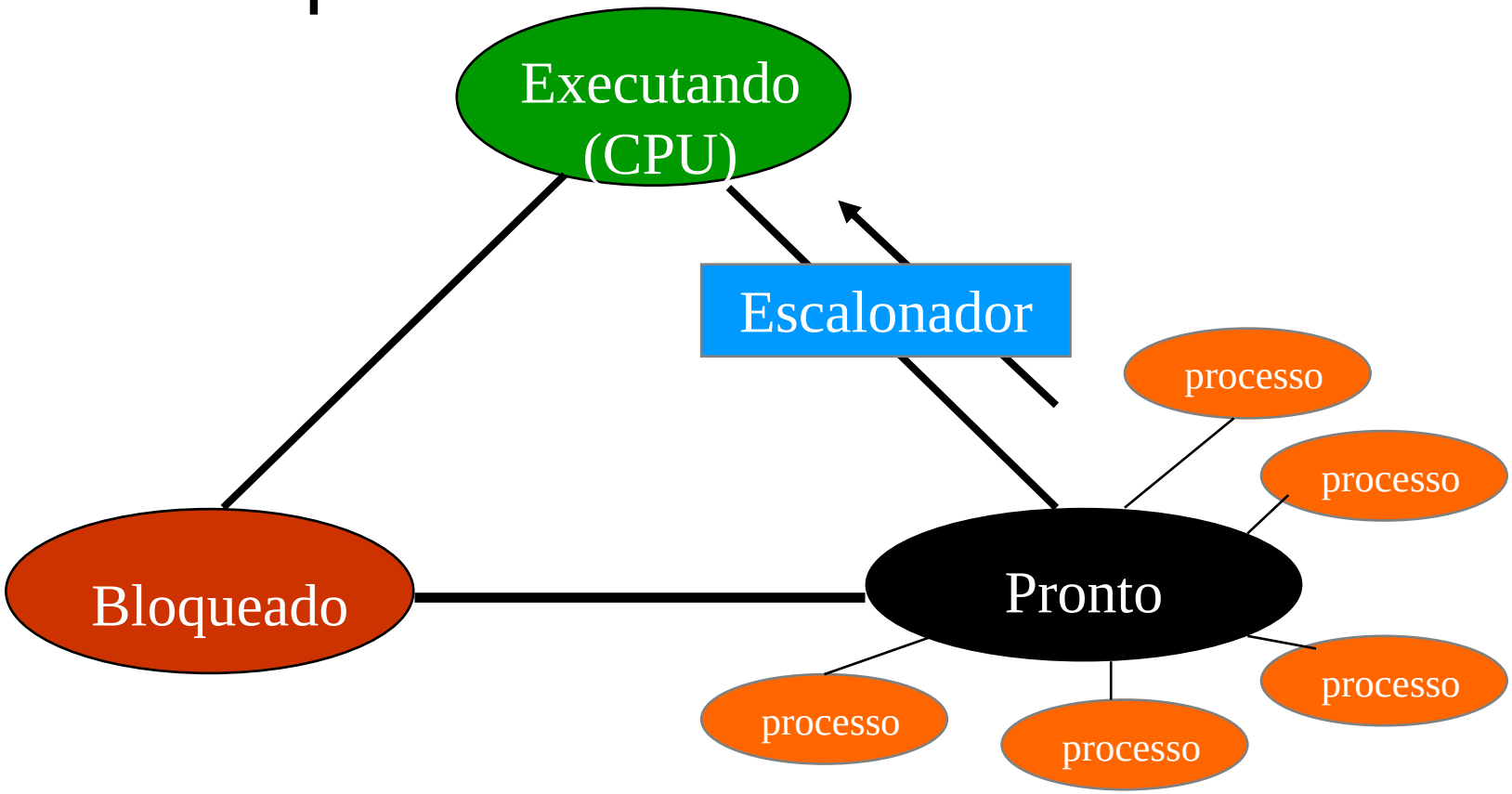
Como que escalonamos processos?

Filas de Escalonamento de Processos

- Fila de Jobs - conjunto de todos os processos do sistema.
- Fila de Prontos - conjunto de todos os processos residentes na memória principal, prontos e esperando para serem executados.
- Fila de Dispositivos - conjunto de processos aguardando por um dispositivo de I/O.
- Processos migram entre as várias filas.

Escalonador

- Parte do sistema operacional que escolhe quem deve utilizar a CPU



Escalonador

- Tem que escolher o processo “certo”
- Tem que se preocupar em fazer uso eficiente da CPU
  - Alternar processos é muito caro!
    - Modo usuário → modo supervisor
    - Estado atual do processo e seus registradores devem ser salvos
    - O mapa de memória deve ser salvo
    - Um novo processo deve ser escolhido
    - O novo processo precisa ser iniciado
    - Recarregar memória cache
  - Pode comprometer uma grande quantidade de tempo da CPU

Objetivos Gerais do Escalonador

- Justiça
  - Processos semelhantes → serviços semelhantes
  - Categorias diferentes podem ser tratados diferentemente
- Cumprimento das políticas do sistema
- Equilíbrio
  - Manter ocupadas todas as partes do sistema
  - Misturar processos CPU-bound e I/O-bound na memória

Tipos de Algoritmos de Escalonamento

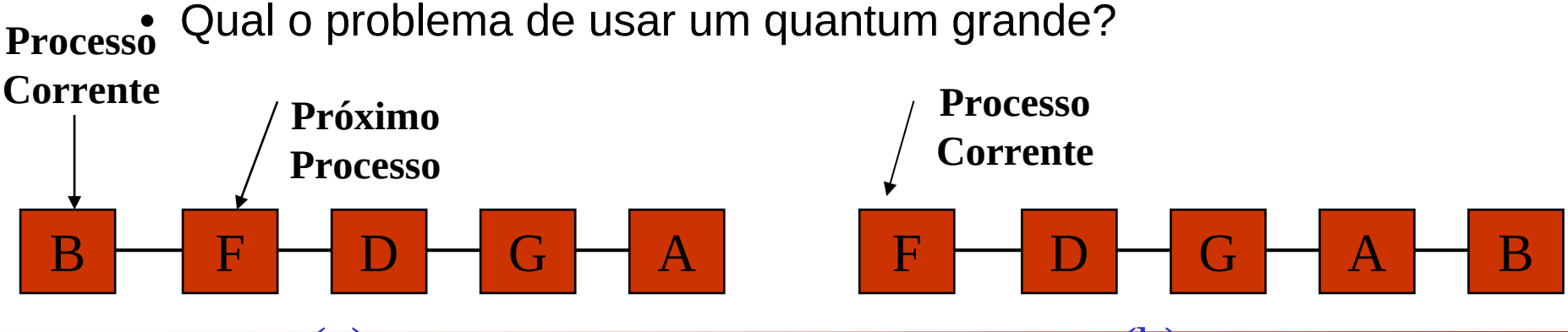
- Algoritmo de escalonamento não-preemptivo
  - Não toma decisões quando ocorrem interrupções de relógio
  - Processos não são compulsoriamente suspensos
- Algoritmo de escalonamento preemptivo
  - Toma decisões quando ocorrem interrupções de relógio
  - Processos tem fatias de tempo (quantum) de uso da CPU
  - Quando esse tempo expira, escolhe um outro processo para executar
  - Proporciona melhores tempos de resposta em sistemas de tempo compartilhado

Escalonadores não-preemptivos

- FIFO
  - Há uma fila única de processos prontos
  - Novos jobs são encaminhados para o fim da fila
  - Quando um processo bloqueia, o próximo da fila é selecionado
  - Processos que passaram do estado bloqueado para pronto são colocados no fim da fila
- Job mais curto primeiro
  - Pressupõe conhecimento prévio dos tempos de execução de todos os processos
  - Privilegia processos de tamanho menor
  - Reduz o tempo médio de espera dos processos

Escalonadores preemptivos

- Alternância circular (Round-Robin)
  - Há uma lista circular de processos prontos
  - Cada processo tem um quantum no qual ele é permitido executar
  - Se o quantum não for suficiente para o processo terminar, ele vai para o fim da fila e aguarda a próxima rodada
  - Qual o tamanho do quantum?
    - Qual o problema de usar um quantum pequeno?
    - Qual o problema de usar um quantum grande?

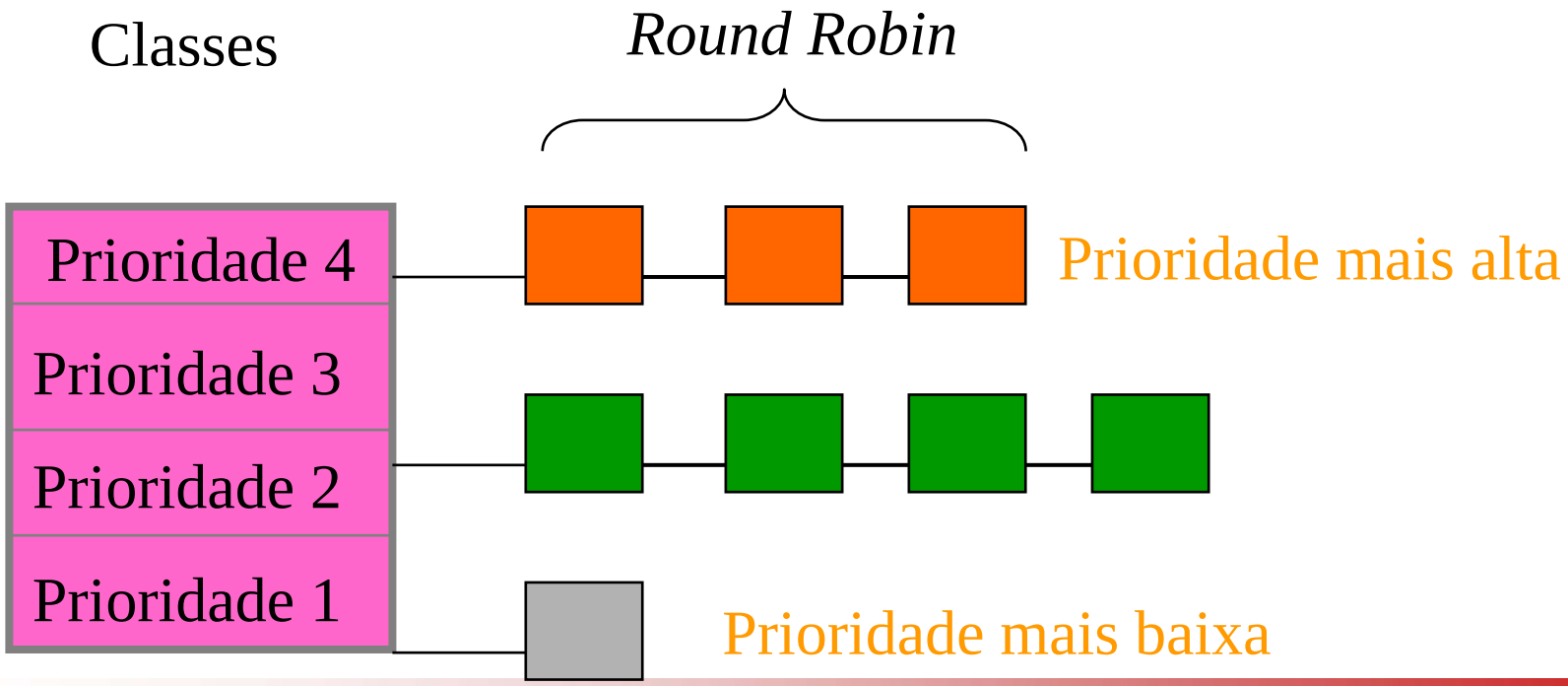


Escalonadores preemptivos

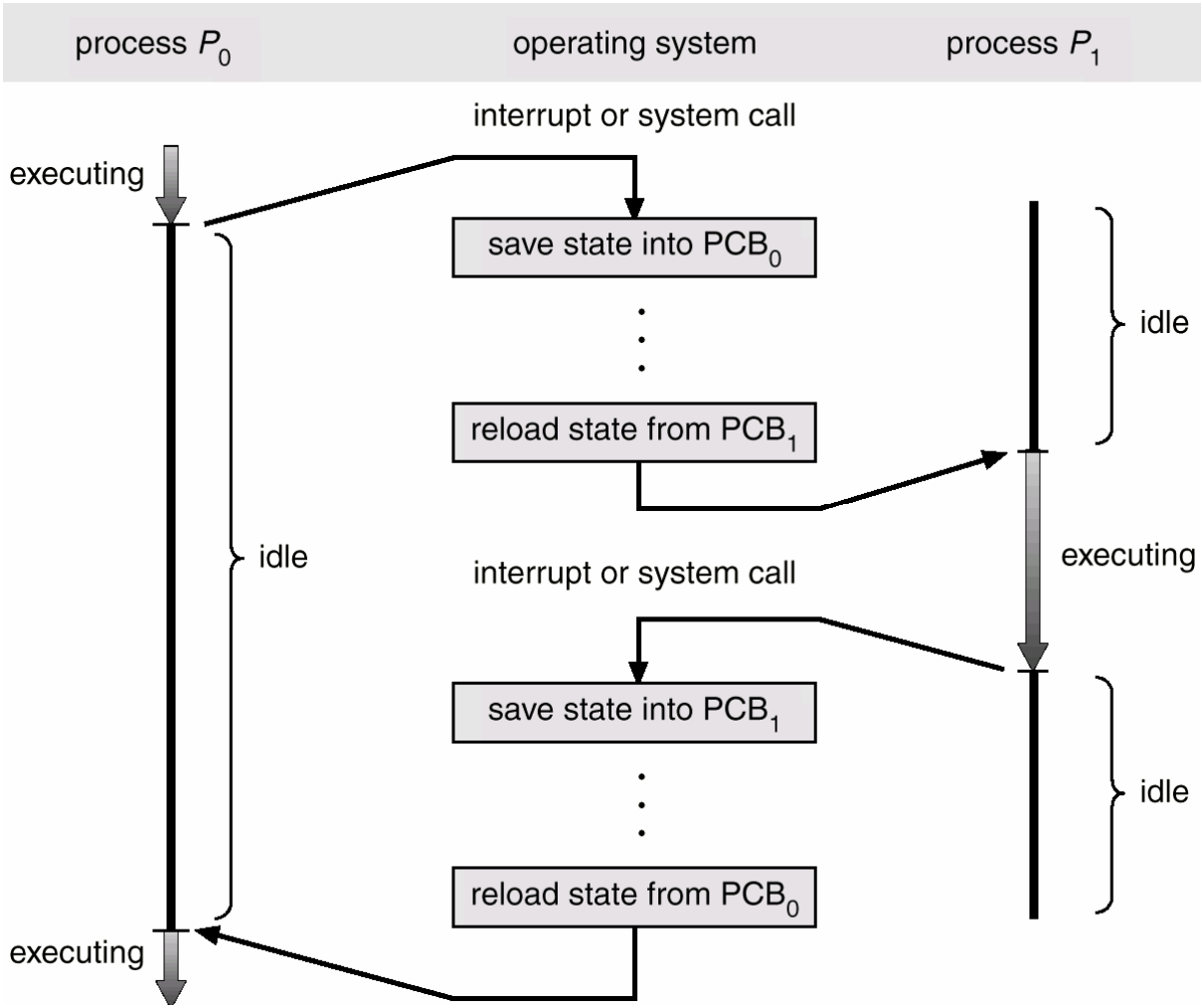
- Escalonamento por prioridades
  - Cada processo tem uma prioridade
  - O processo pronto com prioridade mais alta é escolhido para utilizar a CPU
  - O que fazer para que o processo com prioridade mais alta não monopolize a CPU?
    - Redução de prioridade
    - Quantum máximo

Escalonamento por Prioridades

- Também é comum agrupar processos em classes de prioridades
  - Escalonamento por prioridade entre as classes
  - Round-robin dentro de cada classe



Mudança de Contexto



Mudança de Contexto

- Quando a CPU altera para um outro processo, o sistema deve armazenar o estado do processo antigo e carregar o estado armazenado do novo processo.
- O tempo gasto para a mudança de contexto não é útil aos processos (overhead).
- Este tempo gasto é dependente das características do hardware.

O que podemos fazer para melhorar isso?

O Modelo de Thread

- Diferença entre *thread* e processo
  - Processos são usados para agrupar recursos
  - *Threads* são entidades escalonadas para a execução sobre a CPU
- *Threads* permitem que **múltiplas execuções** ocorram no mesmo ambiente do processo **de forma independente**

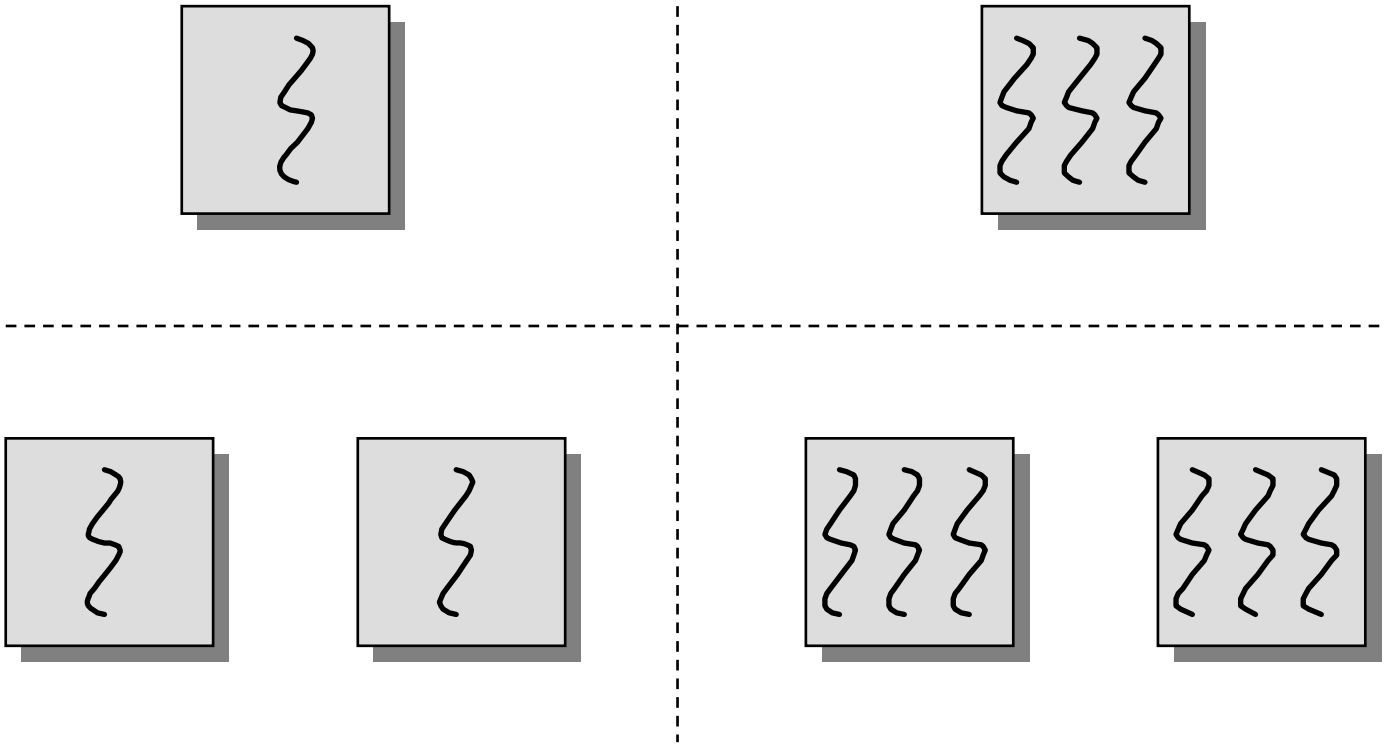
O Modelo de Thread

- Múltiplas *threads* executando em paralelo é semelhante a múltiplos processos executando em um computador
- *Threads* compartilham um mesmo espaço de endereçamento
- Processos compartilham um espaço físico de memória, discos e impressoras e recursos semelhantes
- Exemplo: fazer um bolo

O Modelo de Thread

- *Threads* também são chamados de processos leves (*lightweight process*)
  - Pois possuem somente algumas propriedades dos processos
- *Multithread*
  - Sistemas com a possibilidade de execução de vários *threads* para um mesmo processo
  - Neste ambiente, um processo é definido como a unidade de proteção e a unidade de alocação de recursos

Processos e Threads



Características das Threads

- É mais rápido criar uma thread que um processo
- É mais rápido terminar uma thread que um processo
- É mais rápido chavear entre threads de um mesmo processo
- Threads podem se comunicar sem invocar o núcleo já que compartilham memória e arquivos
  - no caso de comunicação entre processos, a intervenção do núcleo é necessária para proteção e sincronização
- Suspender um processo implica em suspender todas as threads deste processo já que compartilham o mesmo espaço de endereçamento
- O término de um processo implica no término de todas as threads desse processo

Threads em Sistemas Distribuídos

- Proporcionam um meio conveniente para permitir chamadas bloqueadoras de sistema sem bloquear o processo inteiro no qual o thread está executando;
- Imagine um processo monothread: O que aconteceria com o processo quando a interrupção da placa de rede é feita para envio/recebimento de dados?

Clientes Multithread

- Usados para ocultar latências de comunicação, separando threads de envio/recebimento de dados com threads de processamento da interface.
  - Torna possível recebimento de vários arquivos de uma página WEB ao mesmo tempo;
  - Torna possível acesso a vários servidores (redundantes), que servirão os dados independentemente, gerando maior velocidade.

Servidores Multithread

- Além de simplificar o código do servidor, explora paralelismo para obter alto desempenho, mesmo em sistemas monoprocessadores;
  - Um thread despachante cria a divisão de vários threads com tarefas distintas, como ler disco, receber dados de socket, enviar dados para socket, atender N usuários simultaneamente;
  - O thread despachante atribui a requisição a um thread operário ocioso (bloqueado).
- Servidores Monothread não poderiam atender a um segundo usuário enquanto lê disco!

Virtualização e seu papel em S.D.s

- É a capacidade de uma única CPU em fingir que há mais delas, assim como essa extensão a outros recursos.
  - Estende ou substitui uma interface existente para imitar o comportamento de outro sistema

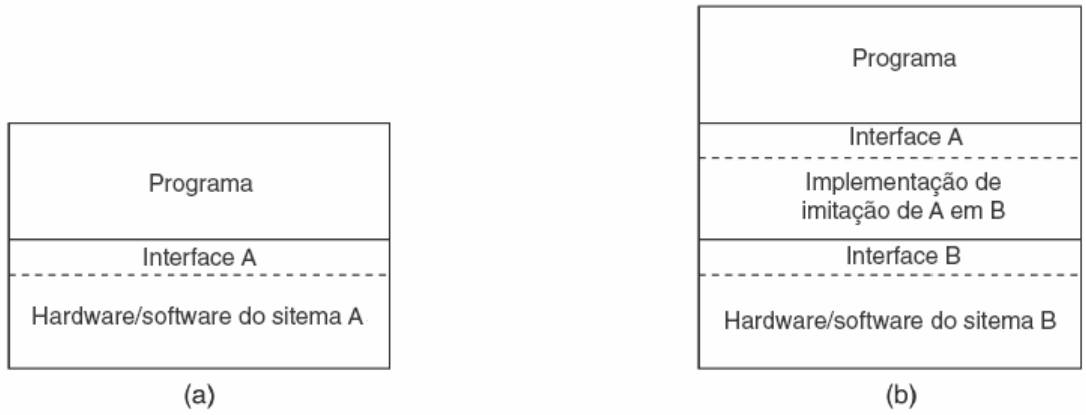


Figura 3.4 (a) Organização geral entre programa, interface e sistema. (b) Organização geral da virtualização do sistema A sobre o sistema B.