CÂMPUS

Bagé

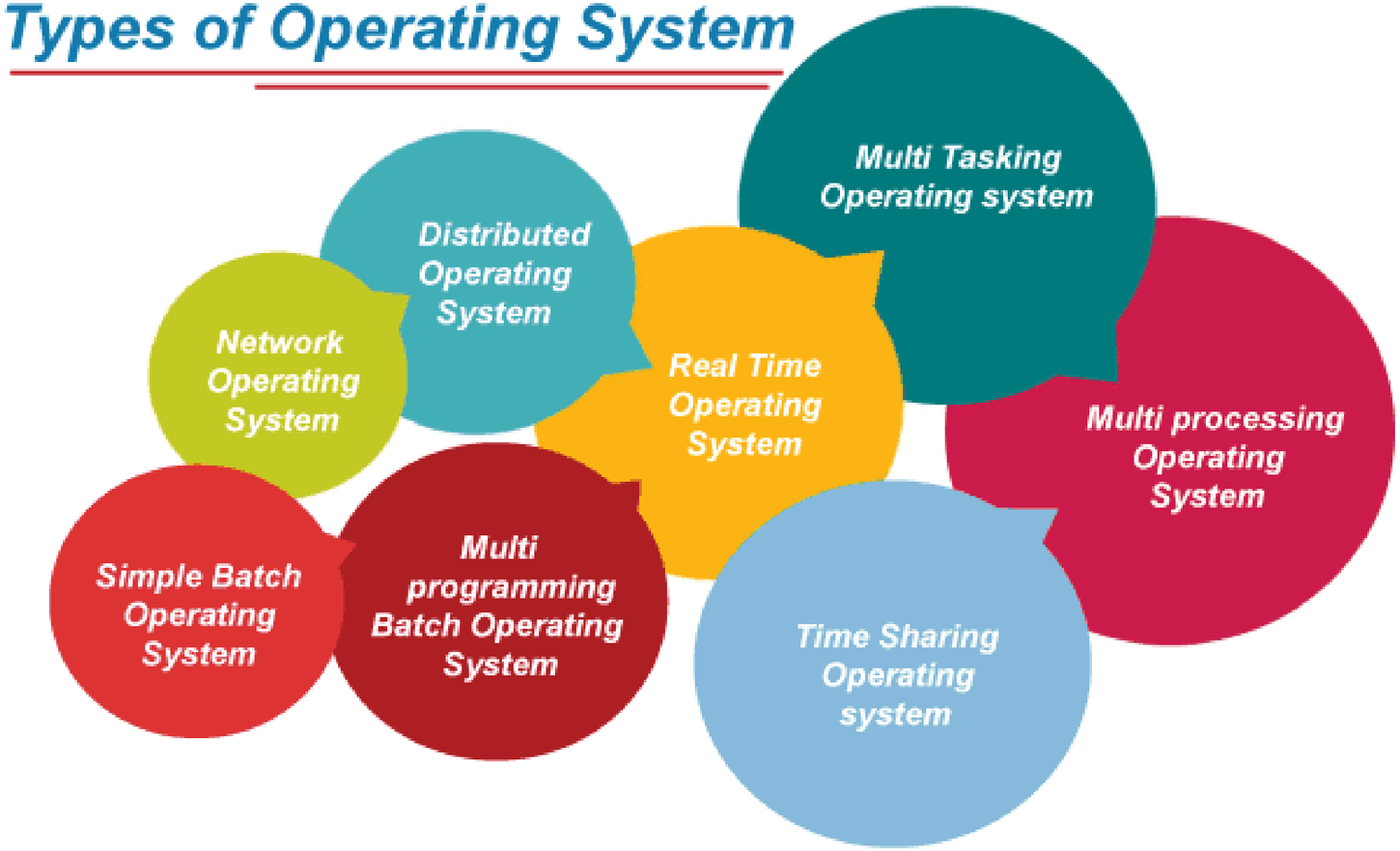
SISTEMAS OPERACIONAIS

INSTITUTO FEDERAL
Sul-rio-grandense

muito a descobrir

RODRIGO R SILVA

Sistemas Operacionais



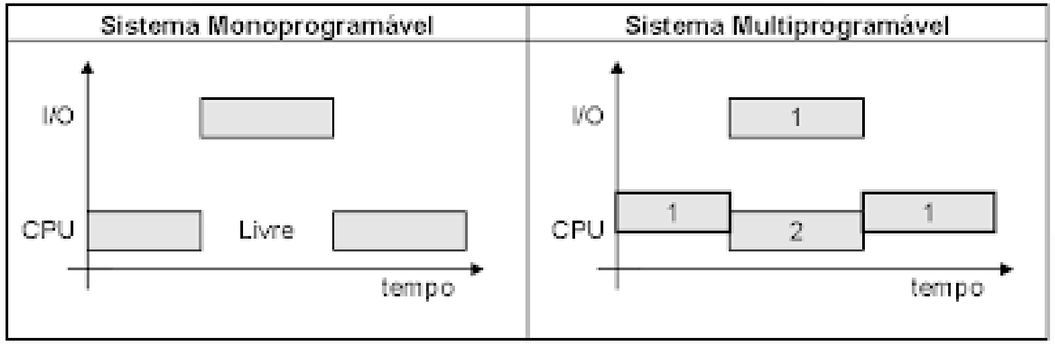
Multiprogramação

Antes da multiprogramação, os primeiros computadores utilizavam monoprogramação, onde apenas um programa era carregado na memória principal e executado de cada vez.

Enquanto esse programa solicitava operações de entrada/saída (E/S), a CPU ficava ociosa, desperdiçando poder computacional.

Esse modelo era pouco eficiente, pois as operações de E/S eram muito mais lentas do que a velocidade de processamento.

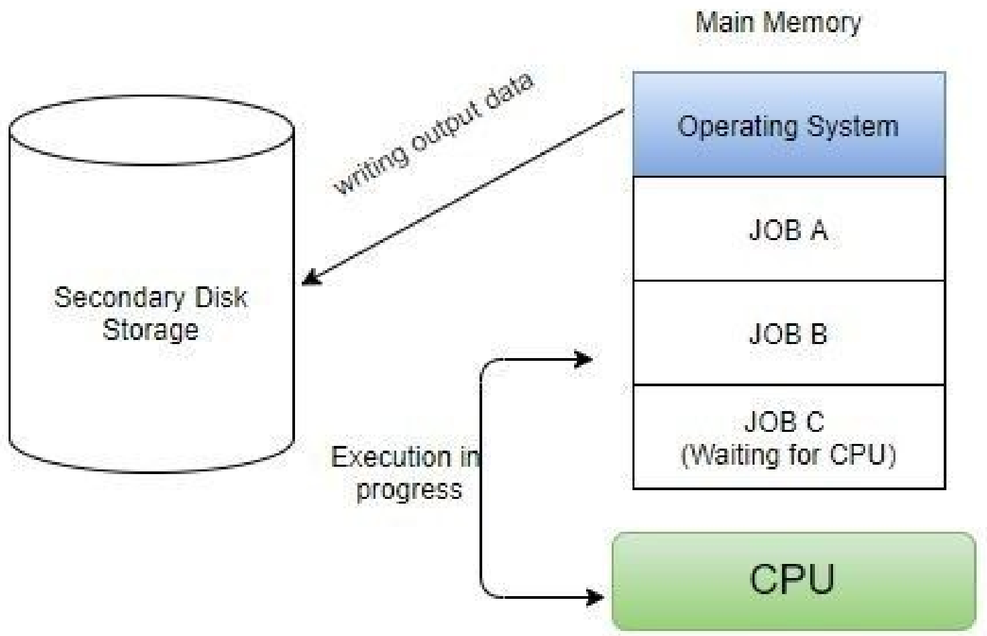
A multiprogramação surgiu nos anos 1960 como resposta a esse problema, aproveitando o tempo de espera de uns programas para executar outros, otimizando o uso do processador.



Multiprogramação

A multiprogramação é a técnica que permite manter vários programas na memória principal ao mesmo tempo, de forma que quando um programa não puder usar a CPU (ex.: espera de E/S), outro programa seja escalonado para execução.

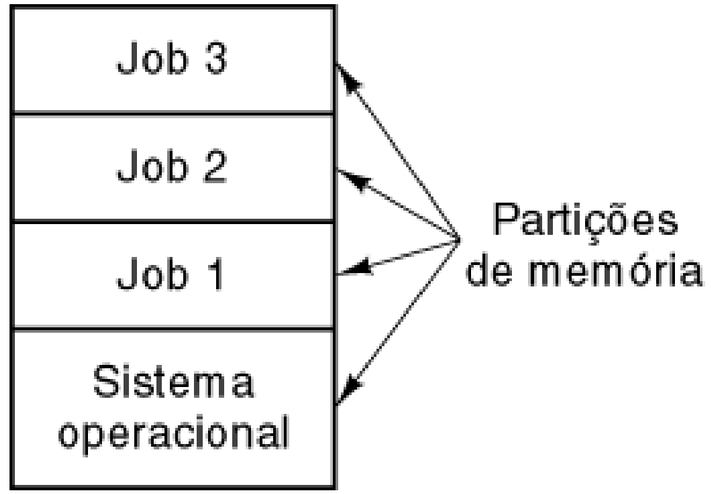
➡ A ideia central: maximizar a utilização da CPU, reduzindo o tempo ocioso.



Multiprogramação

Como Funciona

- Carregamento: o sistema operacional carrega vários programas na memória principal.
- Gerenciamento da CPU:
  - → Quando um processo está em estado de espera (aguardando E/S), outro é colocado para usar a CPU.
  - → O escalonador de processos define qual processo será escolhido.
- Gerenciamento de Memória: a memória deve ser dividida entre processos, garantindo isolamento e proteção.
- Troca de Contexto: o SO salva o estado do processo atual e carrega o estado de outro processo quando ocorre uma troca.
- Execução Concorrente: embora apenas um processo use a CPU por vez, vários progridem de forma alternada, dando a sensação de paralelismo.



Multiprogramação

Componentes Essenciais da Multiprogramação

- **Gerenciamento de Processos:** responsável por criar, suspender, retomar e encerrar processos.
- **Gerenciamento de Memória:** garante que múltiplos programas coexistam na memória sem corromper dados uns dos outros.
- **Escalonamento de CPU:** define critérios para escolher qual processo será executado a seguir.
- **Troca de Contexto:** operação que permite alternar entre processos em execução.
- **Proteção e Segurança:** impede que processos interfiram na memória e recursos de outros.

Multiprogramação

Benefícios da Multiprogramação

- Melhor aproveitamento da CPU (mínimo tempo ocioso).
- Aumento do throughput (número de jobs concluídos em determinado tempo).
- Uso eficiente de recursos de hardware (CPU, disco, impressora, etc.).
- Base para evolução dos sistemas interativos (multitarefa e time-sharing).

Desafios e Limitações

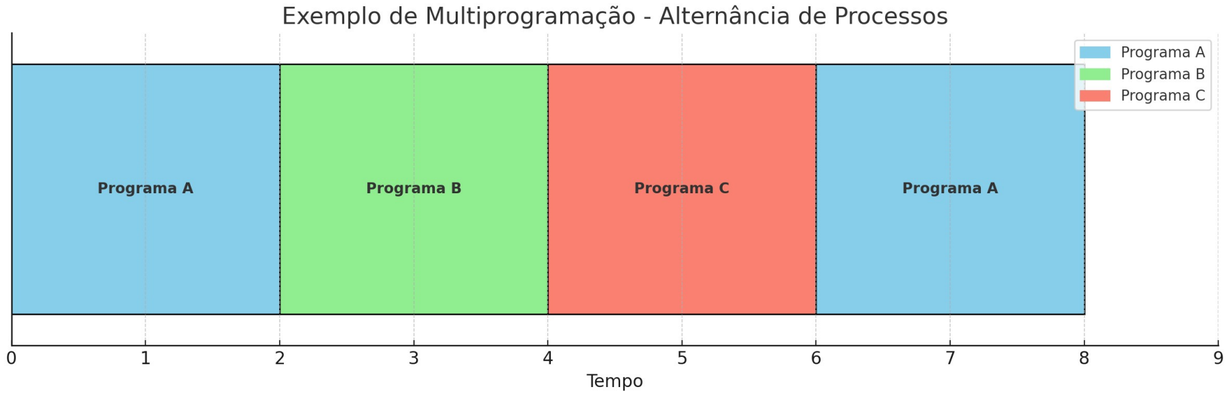
- Gerenciamento complexo: exige algoritmos sofisticados de escalonamento.
- Necessidade de maior memória principal para suportar múltiplos programas.
- Sobrecarga de troca de contexto (quanto mais frequente, maior o consumo de CPU com operações administrativas).
- Ausência de foco no usuário: multiprogramação clássica prioriza eficiência da máquina, não tempo de resposta (isso foi resolvido com sistemas de tempo compartilhado).

Multiprogramação

Exemplo Prático (Ilustração)

Imagine três programas:

- Programa A: executa cálculos (uso de CPU).
- Programa B: solicita leitura de disco (E/S lenta).
- Programa C: imprime relatório (E/S intermediária).



Ciclo:

1. A utiliza a CPU → solicita disco → entra em espera.
2. Enquanto B espera o disco, C assume a CPU.
3. Quando C pede impressora, a CPU retorna ao A (que já recebeu dados do disco).

➡ Resultado: a **CPU nunca fica ociosa**, pois sempre há um processo pronto para executar.

Multiprogramação

Exemplos:

- **Servidores Web :** Considere o ambiente movimentado de um servidor web processando muitas solicitações recebidas. Um Sistema Multiprogramável permite que o servidor alterne eficientemente entre o processamento dessas solicitações, otimizando a capacidade de resposta e garantindo que nenhuma solicitação fique sem resposta.
- **Gerenciamento de Banco de Dados :** No âmbito dos bancos de dados, o Sistema Multiprogramação entra em ação quando vários usuários consultam o banco de dados simultaneamente. A multitarefa do sistema garante que essas consultas sejam processadas simultaneamente, aumentando significativamente a eficiência da recuperação de dados.
- **Edição de Vídeo :** Um software de edição de vídeo executando vários processos simultaneamente – renderização, aplicação de efeitos e exportação de arquivos. O Sistema Multiprogramação orquestra essas tarefas, reduzindo drasticamente o tempo necessário para finalizar projetos.

Compartilhamento de Tempo

Definição Técnica

O time-sharing é uma técnica de gerenciamento de CPU onde múltiplos processos ativos compartilham o processador de forma preemptiva.

- A CPU é dividida em intervalos fixos ou variáveis de tempo chamados quantum (time slice).
- Cada processo recebe um quantum de tempo para executar; ao final desse intervalo, o sistema operacional pode:
  - → Interromper o processo (preempção) e colocar outro na CPU, ou
  - → Mantê-lo em execução, caso ainda não tenha expirado o tempo.
- Essa alternância é gerenciada pelo escalonador de processos.

➡ O objetivo principal é proporcionar interatividade e resposta rápida ao usuário, garantindo que nenhum processo monopolize a CPU.

Compartilhamento de Tempo

Mecanismos Fundamentais

- **Preempção**

→O SO utiliza temporizadores de hardware (timer interrupts) para interromper um processo quando seu quantum expira.

→Isso garante que nenhum processo permaneça indefinidamente na CPU, diferentemente da multiprogramação não-preemptiva.

- **Quantum de Tempo**

→Se o quantum for muito curto → o sistema terá boa responsividade, mas o overhead de troca de contexto será alto.

→Se o quantum for muito longo → reduz-se o overhead, mas piora o tempo de resposta percebido pelos usuários.

→Valores típicos em sistemas modernos: 10 ms a 100 ms.

Compartilhamento de Tempo

Mecanismos Fundamentais

- **Escalonamento de Processos**

O algoritmo mais utilizado em time-sharing é o Round-Robin (RR):

→Os processos prontos são colocados em uma fila circular (fila de prontos).

→O escalonador aloca a CPU para o primeiro processo da fila por um quantum.

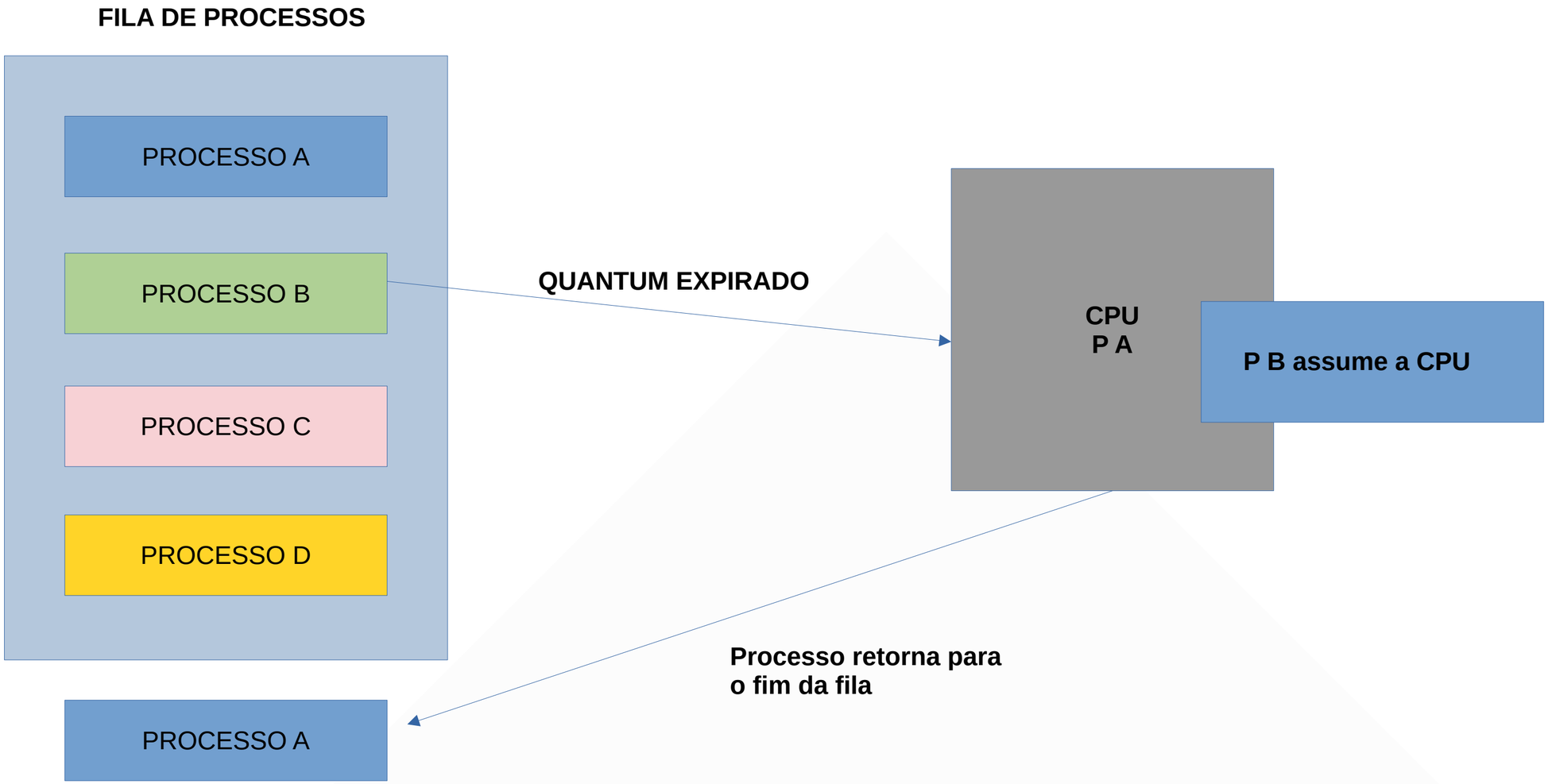
→Quando o quantum expira (ou o processo termina antes), ocorre uma interrupção de timer.

→O processo interrompido vai para o final da fila de prontos, e o próximo processo é escalonado.

- **Troca de Contexto**

- O SO deve salvar o estado atual da CPU (registradores, contador de programa, pilha, etc.) do processo em execução.
- Carregar o estado do próximo processo da fila.
- Essa operação é chamada de context switch e gera overhead, pois consome ciclos da CPU sem executar instruções de usuário.

Compartilhamento de Tempo

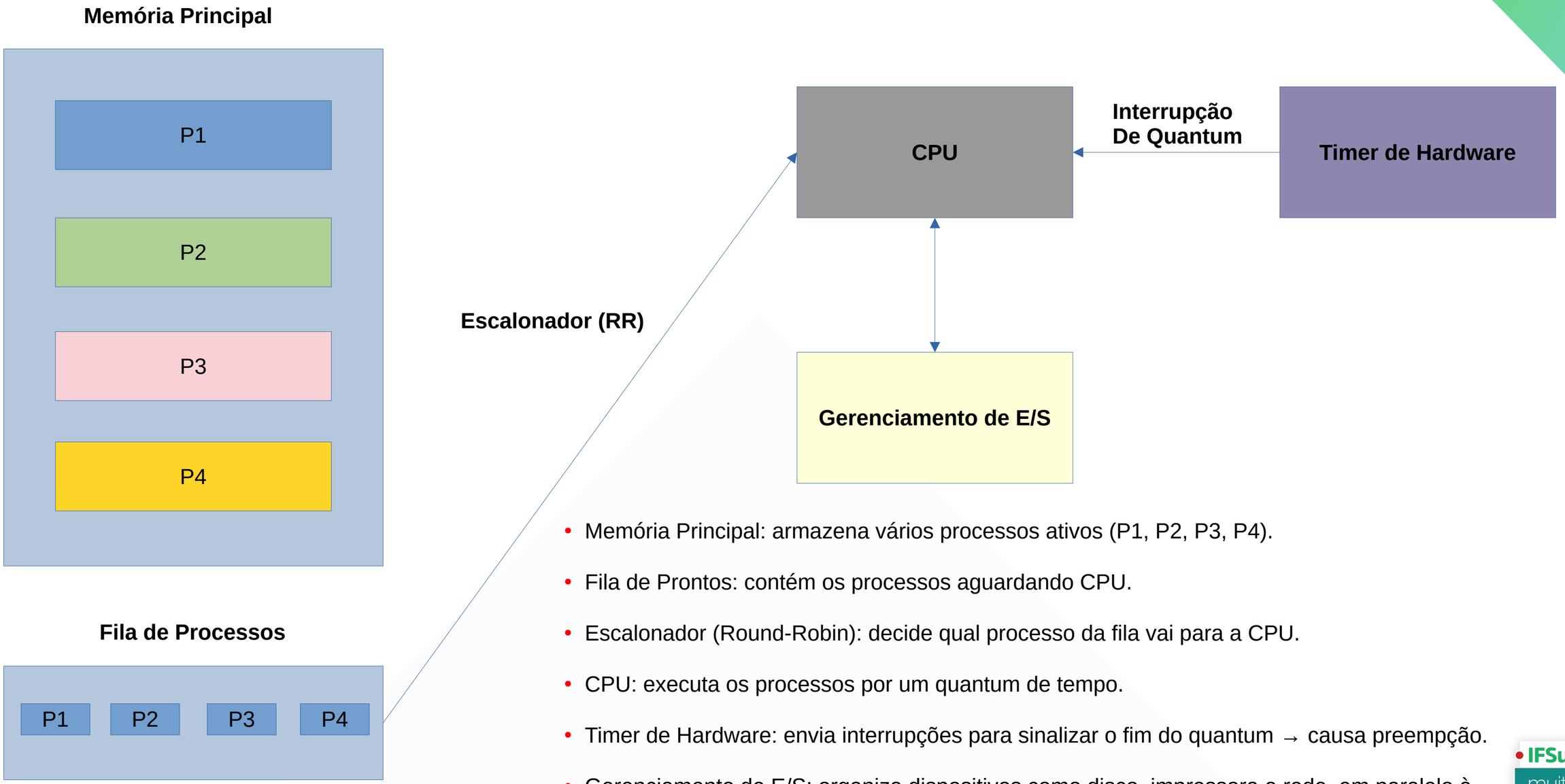


Compartilhamento de Tempo

Estrutura do Sistema em Time-Sharing

- CPU Scheduler: decide qual processo receberá a CPU.
- Dispatcher: executa a troca de contexto entre processos.
- Timer Interrupt: garante que a CPU seja devolvida ao sistema quando o quantum expira.
- Gerenciamento de Memória: todos os processos ativos devem estar (parcial ou totalmente) na memória principal → técnicas como paginação e memória virtual foram introduzidas para viabilizar isso.
- Gerenciamento de Entrada/Saída: processos de E/S não devem bloquear a CPU; filas de espera e mecanismos de spooling são empregados.

Compartilhamento de Tempo



- Memória Principal: armazena vários processos ativos (P1, P2, P3, P4).
- Fila de Prontos: contém os processos aguardando CPU.
- Escalonador (Round-Robin): decide qual processo da fila vai para a CPU.
- CPU: executa os processos por um quantum de tempo.
- Timer de Hardware: envia interrupções para sinalizar o fim do quantum → causa preempção.
- Gerenciamento de E/S: organiza dispositivos como disco, impressora e rede, em paralelo à CPU.

Compartilhamento de Tempo

Métricas de Desempenho em Time-Sharing

Para avaliar a eficiência de um sistema de time-sharing, utilizam-se métricas como:

- Tempo de resposta: intervalo entre a requisição do usuário e a primeira resposta do sistema.
- Tempo de retorno (turnaround time): tempo total para a conclusão de um processo.
- Throughput: quantidade de processos concluídos por unidade de tempo.
- Overhead de troca de contexto: percentual de tempo gasto em operações administrativas (context switch).
- Equidade (fairness): se todos os processos recebem quanta proporcionais sem starvation.

Compartilhamento de Tempo

Benefícios Técnicos

- Suporte a multiusuários com sensação de simultaneidade.
- Melhor experiência em processos interativos (editores de texto, navegadores, IDEs).
- Base para os sistemas multitarefa modernos.
- Previne starvation (inanição) de processos, já que todos recebem tempo de CPU.

Limitações Técnicas

- Overhead elevado de context switch se o quantum for mal configurado.
- Dependência de hardware avançado (necessidade de temporizadores, memória virtual).
- Processos de tempo real podem não ser bem atendidos (por isso existem sistemas operacionais de tempo real – RTOS).
- Difícil balanceamento entre tempo de resposta vs. eficiência global.

Resumo

Conclusão

- Multiprogramação
- Compartilhamento de Tempo Real

Bibliografia Básica

CARISSIMI, A., S. Toscani: **Sistemas Operacionais**. 3. ed. Porto Alegre: Bookman, 2008.

SILBERSCHATZ, A. P.; GALBIN, B.; GAGNE, G. **Fundamentos de Sistemas Operacionais**. 8. ed. São Paulo: LTC, 2010.

TANENBAUM, A. S. **Sistemas Operacionais Modernos**. 3. ed. São Paulo: Pearson, 2010.

Bibliografia Complementar

DEITEL H. M.; DEITEL P. J.; CHOFFNES, D. R.; **Sistemas Operacionais**. 3. ed. São Paulo: Prentice-Hall, 2005.

MACHADO, F. B.; MAIA, L. P. **Arquitetura de Sistemas Operacionais**. 4. ed. São Paulo: LTC, 2007.

TANEMBAUM, A. **Organização Estruturada de Computadores**. Rio de Janeiro: 5. ed. São Paulo: LTC, 2006.

TOSCANI, S. S. **Sistemas Operacionais e Programação Concorrente**. 1. ed. Porto Alegre: Sagra Luzzato, 2003.

TORRES, G. **Hardware:** curso completo. 4. ed. Rio de Janeiro: Axcel Books, 2001.