CÂMPUS

Bagé

SISTEMAS OPERACIONAIS

INSTITUTO FEDERAL
Sul-rio-grandense

muito a descobrir

RODRIGO R SILVA

Tipos de S.O

Os sistemas operacionais (SO) podem ser classificados de diferentes formas, dependendo do critério utilizado (estrutura, finalidade ou forma de uso). Abaixo estão os principais tipos:

- Quanto ao número de usuários

→ Monousuário: permite apenas um usuário por vez.

Exemplo: MS-DOS.

→ Multiusuário: vários usuários acessam recursos ao mesmo tempo.

Exemplo: Unix, Linux, Windows Server.

Tipos de S.O

- Quanto ao número de tarefas

→Monotarefa: executa apenas um programa de cada vez.

Exemplo: MS-DOS.

→Multitarefa: permite rodar diversos programas simultaneamente.

Exemplo: Windows, Linux, macOS.

- Quanto ao acesso a processadores

→Monoprocessado: utiliza apenas um processador.

→Multiprocessado: suporta múltiplos processadores trabalhando em paralelo.

Exemplo: versões modernas de Windows, Linux e Unix.

Tipos de S.O

- Quanto ao tempo de resposta

→Batch (em lote): processa tarefas em sequência, sem interação com o usuário durante a execução.

→Tempo compartilhado (Time-sharing): divide o tempo da CPU entre os usuários/programas.

→Tempo real: responde imediatamente a eventos críticos, usado em sistemas industriais, automotivos e médicos.

Exemplo: QNX, VxWorks.

- Quanto ao dispositivo/uso principal

→Desktop: usados em computadores pessoais.

Exemplo: Windows, Linux, macOS.

→Servidor: voltados a redes, gerenciamento de dados e serviços.

Exemplo: Windows Server, Red Hat Enterprise Linux.

→Móveis/Embarcados: otimizados para dispositivos móveis e sistemas embarcados.

Exemplo: Android, iOS, sistemas de carros e eletrodomésticos.

Tipos de S.O

Critério	Tipos principais	Tipos principais
Usuários	Monousuário / Multiusuário	MS-DOS / Unix
Tarefas	Monotarefa / Multitarefa	MS-DOS / Windows
Processadores	Monoprocessado / Multiprocessado	SO antigos / Linux moderno
Tempo de resposta	Batch / Time-sharing / Tempo real	Mainframes / Unix / QNX
Uso/Dispositivo	Desktop / Servidor / Móvel	Windows / Linux Server / Android

Gerência do Processador

O que é Gerência do Processador

A gerência do processador é a parte do Sistema Operacional responsável por controlar a utilização da CPU.

Como geralmente existem mais processos prontos para execução do que processadores disponíveis, o SO precisa decidir qual processo será executado em determinado momento.

Essa decisão é tomada pelo escalonador (scheduler), que utiliza algoritmos de escalonamento.

Escalonador (Scheduler)

O escalonador é o módulo do SO responsável por escolher, de acordo com critérios definidos, qual processo ocupará a CPU.

Tipos de escalonadores:

- Escalonador de Longo Prazo (Job Scheduler)

  Decide quais jobs entram no sistema para serem processados.

  Controla o grau de multiprogramação (quantidade de processos em memória).

  Ex.: em sistemas batch, escolhe quais jobs da fila serão carregados.

- Escalonador de Médio Prazo

  Atua em sistemas de swap.

  Pode suspender (colocar em disco) ou retomar processos da memória para balancear a carga.

  Ex.: liberar memória quando há sobrecarga.

- Escalonador de Curto Prazo (CPU Scheduler)

  O mais importante. Escolhe qual processo pronto vai usar a CPU.

  Atua com altíssima frequência (milissegundos). Tem impacto direto no desempenho percebido pelo usuário.

Escalonador (Scheduler)

Critérios de Escalonamento

Para decidir, o escalonador usa métricas como:

→Tempo de resposta (tempo até o processo começar a executar).

→Tempo de espera (quanto o processo ficou na fila).

→Tempo de retorno (tempo total do processo no sistema).

→Throughput (número de processos concluídos por unidade de tempo).

→Uso eficiente da CPU.

Escalonador (Scheduler)

Em S.O, os algoritmos de escalonamento podem ser agrupados em dois tipos principais:

- **Algoritmos Não-Preemptivos**

Uma vez que o processo começa a executar na CPU, ele não pode ser interrompido até terminar ou bloquear (ex.: esperar E/S).

O escalonador só atua quando o processo voluntariamente libera a CPU.

Mais simples de implementar, mas pode causar longos tempos de espera para outros processos.

📌 Exemplos de algoritmos não-preemptivos:

FCFS (First Come, First Served / FIFO) → o primeiro a chegar é o primeiro a executar.

SJF (Shortest Job First – versão não-preemptiva) → escolhe sempre o processo com menor tempo estimado de execução.

Escalonamento por Prioridades (não-preemptivo) → processo de maior prioridade entra na CPU, mas não pode ser retirado até terminar.

Escalonador (Scheduler)

- **Algoritmos Preemptivos**

O processo em execução pode ser interrompido pelo SO e devolvido à fila de prontos.

O escalonador pode forçar a troca de contexto (context switch) para dar chance a outro processo.

Melhora a responsividade do sistema, sendo essencial em ambientes interativos e de tempo real.

📌 Exemplos de algoritmos preemptivos:

Round Robin (RR) → cada processo recebe um quantum de tempo e é interrompido quando ele acaba.

SJF (versão preemptiva, também chamada SRTF – Shortest Remaining Time First) → escolhe sempre o processo com menor tempo restante.

Escalonamento por Prioridades (preemptivo) → se chega um processo com prioridade maior, ele interrompe o atual.

Multifilas com realimentação (Feedback Queue) → dinâmica, permitindo mover processos entre filas.

Escalonador (Scheduler)

Critérios de Escalonamento

Para decidir, o escalonador usa métricas como:

→Tempo de resposta (tempo até o processo começar a executar).

→Tempo de espera (quanto o processo ficou na fila).

→Tempo de retorno (tempo total do processo no sistema).

→Throughput (número de processos concluídos por unidade de tempo).

→Uso eficiente da CPU.

S.O em Lote (Batch System)

Contexto Histórico

Os sistemas batch surgiram na década de 1950 e 1960, quando os computadores eram máquinas de grande porte (mainframes) extremamente caros e de uso compartilhado.

Os usuários não interagiam diretamente com a máquina.

O acesso era feito por meio de cartões perfurados, fitas magnéticas ou discos.

Um operador humano recebia os programas, os agrupava em lotes (batches) e entregava para o sistema processar.

Esse modelo foi fundamental no início da computação, pois permitiu que vários programas fossem executados sem a necessidade de presença contínua do programador.

S.O em Lote (Batch System)

Conceito Geral

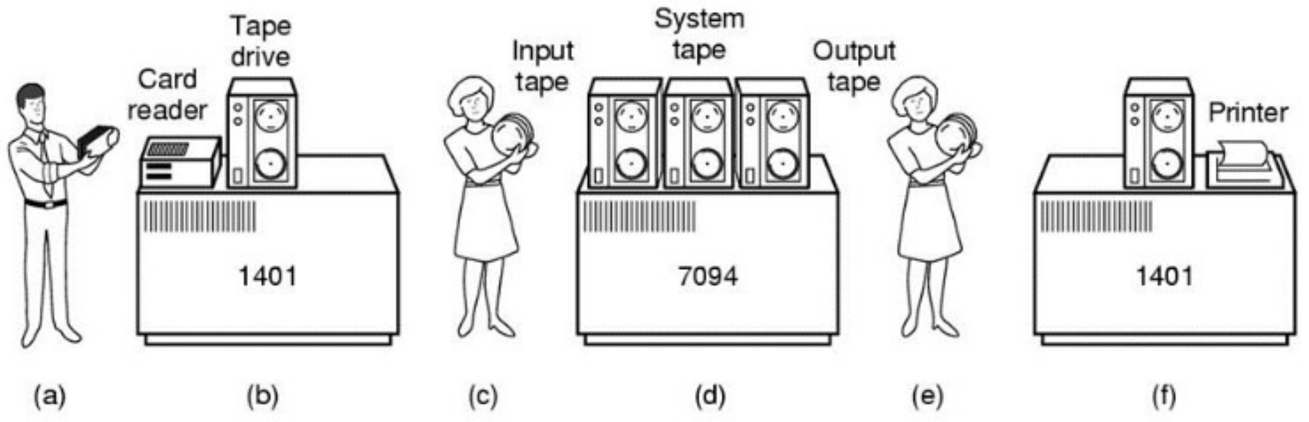
Um sistema batch é um ambiente em que diversos jobs (programas + dados) são enfileirados para execução.

Cada job possui instruções específicas (como o programa a ser rodado, arquivos de entrada e saída).

O SO (Sistema Operacional) organiza a fila de jobs e os executa um após o outro, de maneira automática, sem intervenção humana durante o processamento.

📌 Job = Unidade de trabalho enviada ao sistema, composta por:

- Código a ser executado.
- Dados de entrada.
- Parâmetros de execução.
- Destino da saída (arquivos, impressora, etc.)

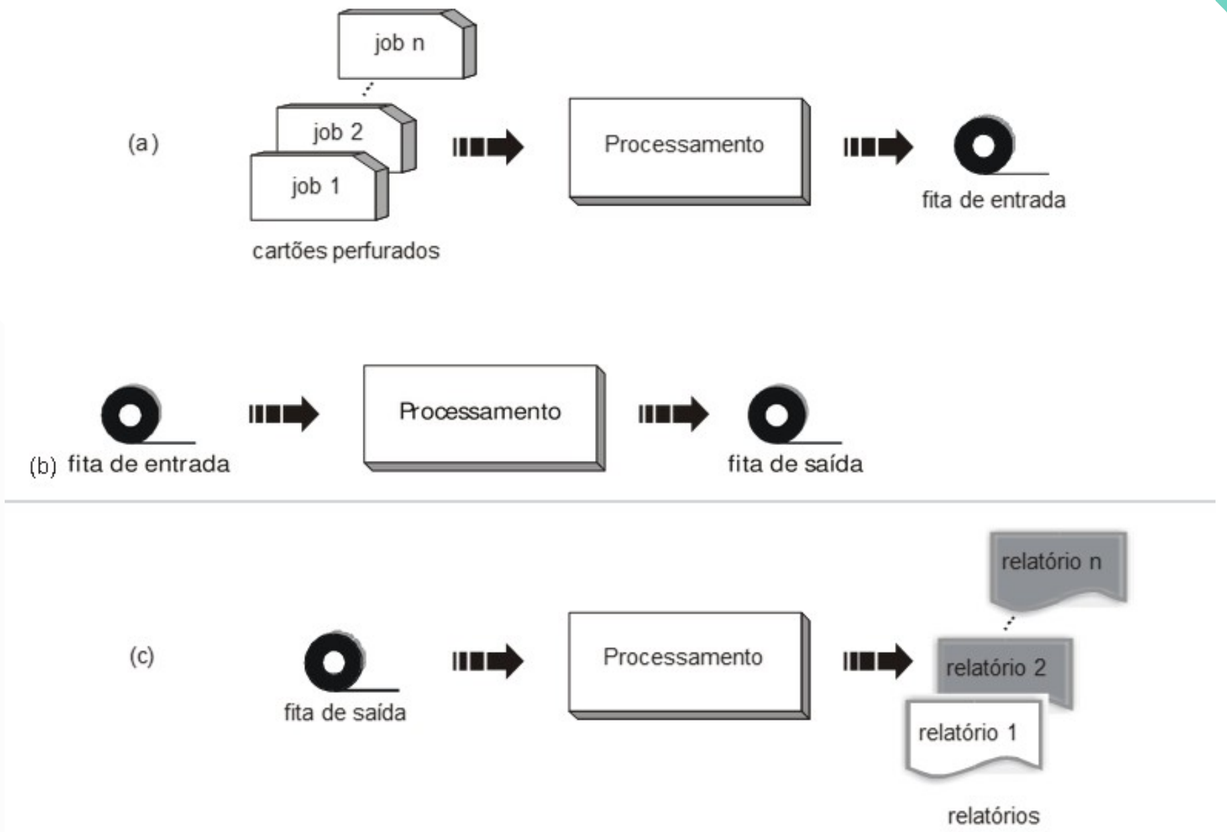


S.O em Lote (Batch System)

Funcionamento Técnico

O processo típico em um sistema batch funciona assim:

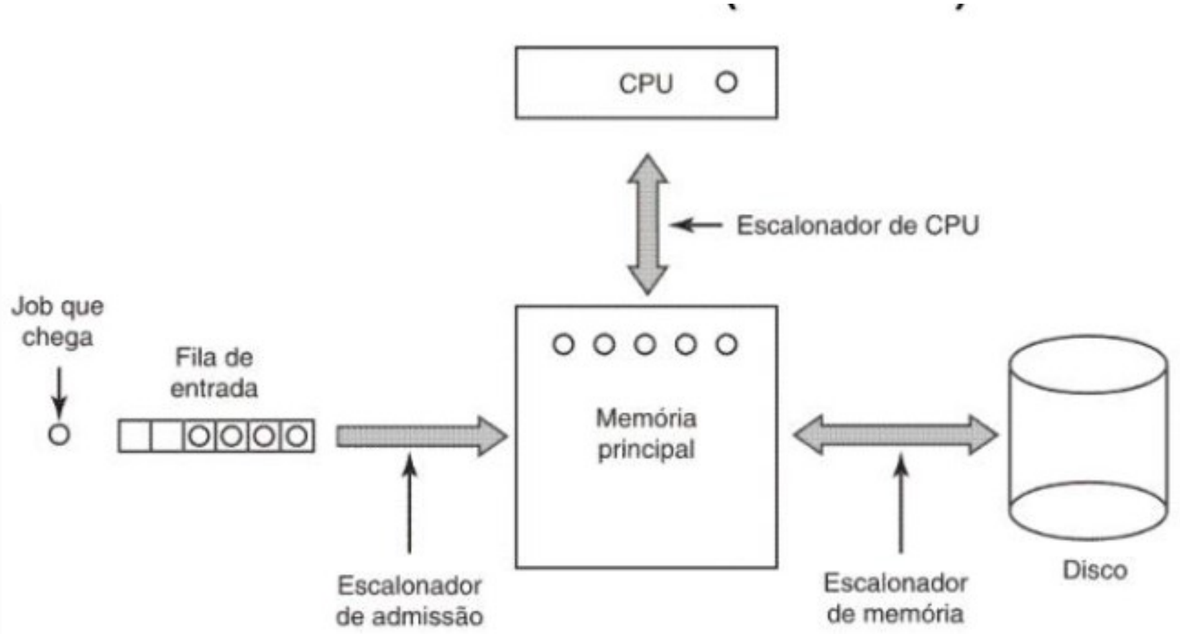
1. Usuário submete seus programas e dados.
2. O operador organiza os jobs em lotes, geralmente de natureza semelhante (ex.: cálculos financeiros, estatísticas, relatórios).
3. O Monitor de Controle de Jobs (job control language – JCL em mainframes IBM, por exemplo) define como o sistema deve tratar cada job.
4. O sistema executa os jobs sequencialmente, controlando entrada/saída (E/S) e uso de CPU.
5. O resultado é devolvido (impresso ou gravado em arquivos) apenas no final da execução.



S.O em Lote (Batch System)

Características Importantes

- **Ausência de Interatividade**: usuários não podem acompanhar nem modificar a execução em tempo real.
- **Execução Sequencial**: jobs são processados na ordem de entrada ou de acordo com critérios de priorização pré-definidos.
- **Alta utilização da CPU**: enquanto um job aguarda E/S, outro pode ser executado (nos sistemas batch mais evoluídos).
- **Simplicidade**: o modelo é linear, fácil de entender e organizar.



S.O em Lote (Batch System)

Vantagens

- Maximização do uso da CPU: aproveita melhor o tempo de processamento.
- Automação: reduz necessidade de intervenção humana.
- Adequado para tarefas repetitivas: como relatórios, cálculos de grande volume ou estatísticas.
- Estabilidade: execução previsível e organizada.

Desvantagens

- Tempo de Espera Elevado (Turnaround Time): o usuário só recebe os resultados depois que todo o lote termina.
- Pouca Flexibilidade: se um job falhar, pode comprometer o lote inteiro.
- Sem interação imediata: impossível corrigir dados de entrada ou erros durante a execução.
- Requer boa organização: depende de operadores treinados e de lotes bem preparados.

S.O de Tempo Real (Real-Time System)

Definição

Um sistema operacional de tempo real (RTOS – Real-Time Operating System) é um SO projetado para fornecer respostas imediatas e previsíveis a eventos externos.

- O foco principal não é apenas rapidez, mas sim determinismo (garantia de que uma tarefa será concluída dentro de um prazo específico).
- Em tempo real, o tempo de resposta é parte da correção do sistema: se a tarefa não for concluída no prazo, o sistema pode falhar em sua função.

S.O de Tempo Real (Real-Time System)

Funcionamento

- O sistema reage a eventos externos (como sensores, sinais ou interrupções).
- Os processos são escalonados de acordo com prioridades rígidas e prazos de execução (deadlines).
- Tarefas críticas têm prioridade absoluta sobre tarefas menos importantes.
- Usa frequentemente escalonamento preemptivo (interrompe uma tarefa em execução para atender outra mais urgente).

📌 Exemplo prático:

→Em um carro com airbag, o sistema deve inflar o airbag em milissegundos após o impacto.

→Se a resposta atrasar, mesmo que seja só 1 segundo, o sistema é considerado falho, mesmo que o código esteja correto.

S.O de Tempo Real (Real-Time System)

Tipos de Sistemas de Tempo Real

- Hard Real-Time (Tempo Real Rígido)

→Os prazos (deadlines) devem sempre ser cumpridos.

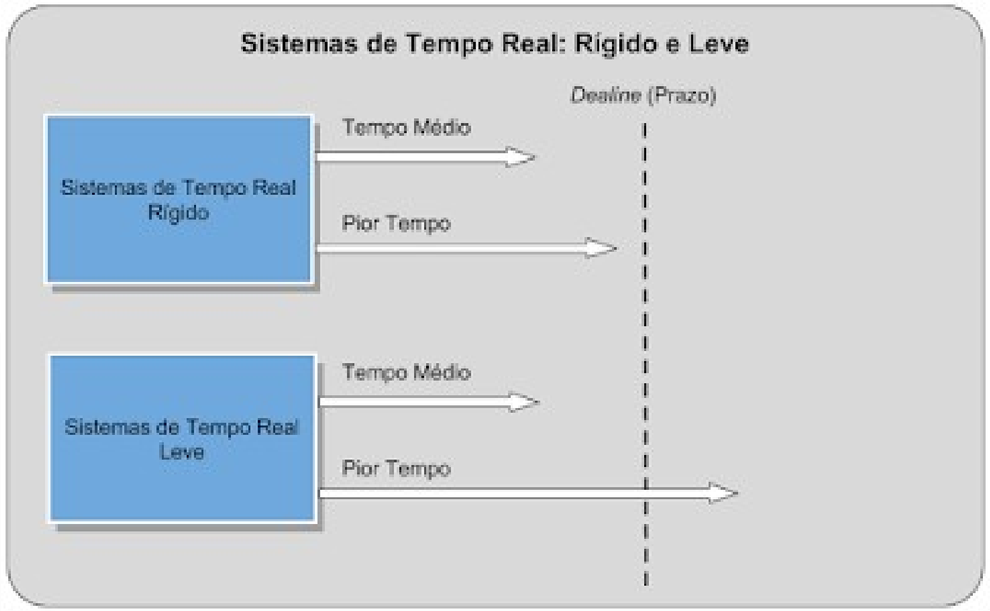
→Qualquer atraso resulta em falha catastrófica.

→Exemplo: controle de avião, airbag automotivo, equipamentos médicos (marcapasso).

- Soft Real-Time (Tempo Real Flexível)

→O cumprimento dos prazos é desejável, mas pequenos atrasos são tolerados.

→Exemplo: streaming de vídeo, jogos online, chamadas VoIP.



S.O de Tempo Real (Real-Time System)

Características

- Determinismo: previsibilidade na execução.
- Alta prioridade para eventos críticos.
- Uso intensivo de interrupções.
- Escalonamento preemptivo é regra, com algoritmos específicos para deadlines.
- Eficiência e confiabilidade: o sistema deve ser pequeno, rápido e estável.

S.O de Tempo Real (Real-Time System)

Vantagens

- Resposta imediata a eventos críticos.
- Alta confiabilidade em aplicações de missão crítica.
- Garantia de prazos de execução (quando projetado corretamente).

Desvantagens

- Implementação complexa.
- Custo elevado (hardware e software dedicados).
- Flexibilidade limitada → o foco está em tarefas críticas, não em usabilidade geral.

S.O: Batch e Tempo Real

Conclusão

- Tipos de S.O
- Escalonador
- Batch
- 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.