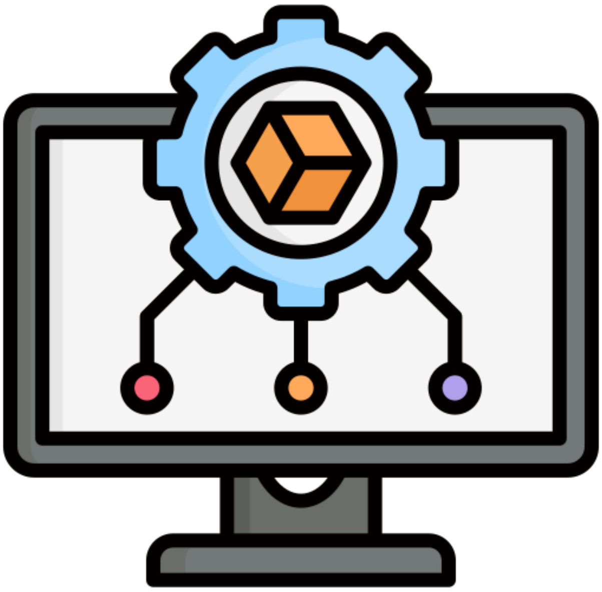
=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=

Escalonador de Processos

=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=

===

Disciplina: Arquitetura de computadores - ICP246



RELATÓRIO

SIMULADOR DE ESCALONAMENTO DE PROCESSOS
ESTRATÉGIA DE SELEÇÃO ROUND ROBIN (OU CIRCULAR)
COM FEEDBACK

SUMÁRIO

1. Introdução

Um simulador de escalonamento de processos tem como objetivo reproduzir, de maneira simplificada, as técnicas utilizadas por um sistema operacional real para gerenciar a execução de múltiplos processos concorrentes. Esses processos competem por tempo de CPU (Unidade Central de Processamento) e por recursos como os dispositivos de Entrada e Saída (E/S ou I/O), fundamentais para a interação do computador com o ambiente externo — recebendo dados (entrada) e produzindo saídas com base nas instruções processadas.

No simulador desenvolvido, foi implementada uma combinação entre as estratégias de escalonamento Round Robin e Feedback, que juntas proporcionam uma abordagem dinâmica e justa para a execução dos processos. O Round Robin garante a alternância entre os processos de forma cíclica e preemptiva, atribuindo a cada um uma fatia de tempo fixa (quantum) para execução. Quando esse tempo se esgota e o processo ainda não finalizou, ele é preemptado e retorna para a fila de prontos. Já a estratégia de Feedback adiciona uma camada de priorização adaptativa, penalizando processos que consomem mais tempo de CPU, rebaixando-os para filas de menor prioridade, enquanto prioriza processos novos ou que retornam de operações de I/O.

Durante o desenvolvimento, diversos conceitos da disciplina de Arquitetura de Computadores e Sistemas Operacionais foram empregados e reforçados. Entre eles, destacam-se: gerenciamento de processos, filas de prioridade, troca de contexto, chamadas de sistema (System Calls), tempo de fatia (Time Slice), estados dos processos, interrupções, operações de entrada e saída, alocação e compartilhamento de recursos, segurança e integridade de acesso, além dos conceitos de multiprogramação e sistemas multitarefa. Ainda que o simulador opere com um conjunto fixo de processos (sem entrada dinâmica durante a execução), ele ilustra aspectos importantes tanto do escalonamento de curto prazo quanto de decisões estratégicas adotadas em sistemas reais.

Este trabalho, portanto, não apenas simula o funcionamento interno de um escalonador, como também consolida o aprendizado teórico por meio da prática e da experimentação.

2. Objetivos do trabalho

Este trabalho tem como principal finalidade o desenvolvimento de um simulador capaz de implementar um algoritmo de escalonamento de processos. A estratégia adotada foi o escalonamento Round Robin

com Feedback, amplamente reconhecida por promover uma alocação equitativa do tempo de CPU entre os processos, ao mesmo tempo em que considera suas prioridades e interações com operações de Entrada e Saída (I/O).

Além de oferecer uma visão prática sobre o gerenciamento de processos realizado pelo sistema operacional, o projeto visa aprofundar a compreensão teórica adquirida ao longo da disciplina de Arquitetura de Computadores e Sistemas Operacionais. Durante a implementação, diversas dúvidas e situações práticas surgiram, o que exigiu da equipe uma análise crítica do comportamento de sistemas reais, com foco na otimização do desempenho global. Esse processo levou à compreensão de que o desempenho não depende apenas da CPU ou dos dispositivos de I/O isoladamente, mas da interação eficaz entre todos os componentes envolvidos.

Dado o grau de complexidade envolvido no desenvolvimento de um simulador funcional, a organização do projeto foi essencial. Para isso, o grupo estruturou um planejamento colaborativo com reuniões, divisão de responsabilidades entre os membros e modularização do código-fonte, facilitando sua manutenção e evolução. O trabalho em equipe também foi um elemento enriquecedor, permitindo troca de experiências, discussões técnicas e maior entendimento sobre os desafios reais enfrentados por escalonadores em sistemas operacionais.

3. Premissas estabelecidas para o Escalonador

Durante o desenvolvimento do simulador de escalonamento de processos, foram definidas diversas premissas para modelar o comportamento de um sistema operacional real, respeitando os requisitos do trabalho proposto e os fundamentos teóricos estudados na disciplina.

3.1. Gerência de Processos

O simulador utiliza quatro filas principais para o controle e organização dos processos:

- ➔ **highQueue: fila de alta prioridade;**
- ➔ **lowQueue: fila de baixa prioridade;**

- **ioQueue: fila única para todos os processos em operação de I/O;**

- **arrivalQueue: fila de processos ainda não prontos para execução, com tempo de chegada definido.**

Cada processo é representado por uma estrutura contendo as seguintes informações:

- **Identificador único (PID), prioridade, ciclos totais e restantes, tempo de chegada, nome, status ativo e, caso possua, uma estrutura com os dados da requisição de I/O.**

A ordem de execução prioriza a Fila de alta prioridade. Se esta estiver vazia, a execução passa para a Fila de baixa prioridade. Se ambas estiverem vazias, a CPU entra em estado de ociosidade.

Todos os processos, tanto prontos quanto bloqueados, são organizados em filas, enquanto os processos em execução (CPU ou I/O) são manipulados em variáveis auxiliares.

3.2. Escalonamento Round Robin com Feedback

- O algoritmo adotado é Round Robin com Feedback.

- A fatia de tempo (quantum) definida foi de 4 unidades de tempo (u.t.), como constante global do simulador.

- O quantum não é reaproveitado entre processos. Se um processo finaliza ou é enviado para I/O antes de consumir todo seu quantum, o restante é desconsiderado, garantindo isonomia no escalonamento.

- Estratégia de realocação dos processos:

  - Processos preemptados (aqueles que não completam sua execução dentro do quantum) são movidos para a fila de baixa prioridade;

  - Processos que retornam de I/O são movidos para a fila de alta ou baixa prioridade, dependendo do tipo de I/O realizado (ver seção 3.4).

➔ O feedback é implementado de forma simples e eficaz, por meio de movimentação entre as filas, penalizando ou recompensando o processo conforme seu comportamento.

3.3. Geração e Chegada de Processos

➔ O simulador permite duas formas distintas de geração de processos: geração aleatória e geração manual via código. Ambas respeitam as mesmas estruturas e regras definidas para entrada no sistema e execução.

3.3.1 Geração Aleatória

➔ **Ao iniciar a simulação, o usuário define o número de processos (até o limite de 50), que serão gerados aleatoriamente.**

➔ **Cada processo é criado com os seguintes atributos randomizados:**

- **Total de ciclos: entre 2 e 20;**
- **Tempo de chegada: entre 0 e 20 ciclos;**
- **Prioridade inicial: Alta;**
- **Possui I/O: definido aleatoriamente;**
- **Tipo de I/O (se existir): aleatório entre Disco, Fita Magnética ou Impressora;**
- **Ciclos antes do I/O: Entre 1 e Total de ciclos -1.**

3.3.2 Geração Manual via Código

➔ **O simulador também permite que processos sejam criados diretamente no código-fonte, com parâmetros definidos manualmente pelo programador.**

➔ **Essa funcionalidade é útil para testes dirigidos, simulações específicas e validação de casos extremos.**

➔ **Os processos manuais seguem o mesmo modelo de estrutura e podem ser adicionados diretamente às filas Alta, Baixa ou Chegada, conforme a lógica da simulação e os valores de**

Tempo de chegada.

3.3.3 Entrada no Sistema

- ➔ **Processos com Tempo de chegada == 0 são imediatamente inseridos em sua fila correspondente (alta prioridade);**
- ➔ **Processos com Tempo de chegada > 0 são colocados na Fila de chegada, onde aguardam decremento ciclo a ciclo até sua entrada no sistema.**

Essa flexibilidade de geração permite tanto a simulação de cargas de trabalho realistas, quanto a construção de cenários controlados para análise de desempenho e comportamento do escalonador.

3.4. Gerência de I/O

- ➔ **Foi definida uma única fila de I/O para todos os tipos de dispositivos, e o simulador considera recursos infinitos: múltiplos processos podem estar em operação de I/O simultaneamente.**
- ➔ **A cada ciclo:**
  - ◆ **Todos os processos em I/O têm seus ciclos restantes decrementados.**
  - ◆ **Ao completar a operação de I/O, o processo retorna automaticamente à fila de prontos de acordo com o tipo:**
    - **Disco → volta à fila de baixa prioridade;**
    - **Fita magnética e Impressora → voltam à fila de alta prioridade.**

3.5. Estado de Ociosidade da CPU

- ➔ A CPU entra em estado de ociosidade se ambas as filas de prontos (Alta e Baixa) estiverem vazias, mesmo que haja processos nas filas de chegada ou I/O.
- ➔ **Durante a ociosidade:**
  - ◆ **A contagem de ciclos continua normalmente.**
  - ◆ **O quantum é decrementado até o próximo reinício.**

- Processos em I/O ou chegada continuam sendo atualizados normalmente.

➔ Ao fim do quantum ocioso, o sistema verifica novamente se há processos prontos. Caso positivo, o ciclo de execução retorna.

3.6. Limitações e Parâmetros Definidos

Parâmetro	Valor
Número máximo de processos	50
Quantum de execução	4 ciclos
Ciclos de I/O (por tipo)	Disco: 4 u.t. Fita: 7 u.t. Impressora: 5 u.t.
Ciclos de chegada	Definido ou aleatório
Ciclos por processo	Definido ou aleatório

As decisões de implementação descritas acima foram baseadas em simulações realistas e estudos de comportamento de sistemas operacionais multitarefa. Todas as premissas foram aplicadas fielmente no código-fonte, com estrutura modularizada, e os resultados puderam ser observados por meio da geração de logs detalhados e estatísticas finais, que serão apresentados nas seções seguintes.

4. Funcionamento do Simulador

O simulador tem dois modos de funcionamento.

Modo automático: Faz uma simulação completa instantânea.

Modo Passo a Passo: Aguarda interação do usuário antes de realizar cada ciclo (Ideal para testes).

4.1. Fluxo Geral da Simulação

A simulação é baseada em ciclos sucessivos, onde a cada ciclo global:

- ➔ O processo atual (se houver) consome uma unidade de seu quantum.
- ➔ Todos os processos em espera na fila de I/O têm seus ciclos de bloqueio decrementados.
- ➔ A fila de chegada tem seus tempos de chegada reduzidos, liberando processos para as filas de pronto no momento adequado.

O quantum global é fixo em 4 unidades, reiniciado exclusivamente a cada esgotamento.

4.2. Chegada de Processos

Os processos podem ser gerados com um valor aleatório de tempo de chegada entre 0 e 20. Aqueles com tempo 0 são imediatamente direcionados à sua fila de prioridade inicial. Os demais são mantidos em uma fila de chegada (`arrivalQueue`) e transferidos para as filas de execução quando seu tempo de chegada chega a zero.

4.3. Estratégia Round Robin com Feedback

O escalonamento principal segue a política Round Robin, na qual cada processo recebe uma fatia de tempo fixa para execução. O feedback ocorre através de duas filas de prioridade:

- ➔ Fila de Alta Prioridade (`highQueue`): recebe processos recém-chegados, e processos que retornam do I/O do tipo fita magnética ou impressora.
- ➔ Fila de Baixa Prioridade (`lowQueue`): recebe processos preemptados e processos que retornam do I/O do tipo disco.

A CPU sempre executa o primeiro processo da fila de alta prioridade; se estiver vazia, seleciona da fila de baixa prioridade.

Se um processo for finalizado antes de consumir totalmente seu quantum, o quantum restante é descartado até seu reinicio. Isso pode não ser tão otimizado, mas é justo e segue as normas do algorítmo Round Robin.

4.4. Gerência da Fila de I/O

Todos os processos que solicitam operações de entrada/saída são movidos para uma fila única de I/O (**ioQueue**) após os Ciclos até I/O chegarem a 0. Cada processo possui um tempo de espera correspondente ao tipo de I/O requisitado:

- ➔ Disco: retorna à fila de baixa prioridade.
- ➔ Fita magnética: retorna à fila de alta prioridade.
- ➔ Impressora: retorna à fila de alta prioridade.

Durante a ociosidade da CPU ou mesmo em execução normal, todos os processos em I/O têm seus tempos decrementados simultaneamente (Recursos infinitos). Quando esse tempo atinge zero, o processo é reenfileirado conforme sua prioridade de retorno.

4.5. Estado de Ociosidade da CPU

A CPU entra em modo ocioso quando não há processos nem na fila de alta, nem na de baixa prioridade. Durante essa fase, os ciclos globais continuam passando, mantendo o funcionamento da fila de I/O e da chegada de novos processos.

Ao retornar algum processo à fila de execução (seja pela chegada ou finalização do I/O), a CPU se mantém ociosa até o próximo Quantum.

4.6. Finalização da simulação

Quando todas as filas estão vazias (**Alta, Baixa, I/O, Chegada**), e a CPU não está ociosa, o programa finaliza e exibe as estatísticas da simulação.

5. Saída do Simulador

A saída do simulador foi projetada para apresentar de forma detalhada o comportamento dos processos durante a execução do escalonador. Durante cada ciclo do clock, o programa exibe:

- **O número do ciclo atual e o quantum restante.**
- **O estado das filas:**
  - **Fila de Alta Prioridade**
  - **Fila de Baixa Prioridade**
  - **Fila de I/O**
- **As informações do processo em execução, como:**
  - **ID**
  - **Ciclos restantes**
  - **Tipo de I/O (caso aplicável)**
  - **Situação (executando, paralisado, preemptado ou finalizado)**

Além disso, são mostradas mensagens quando ocorrem eventos importantes, como:

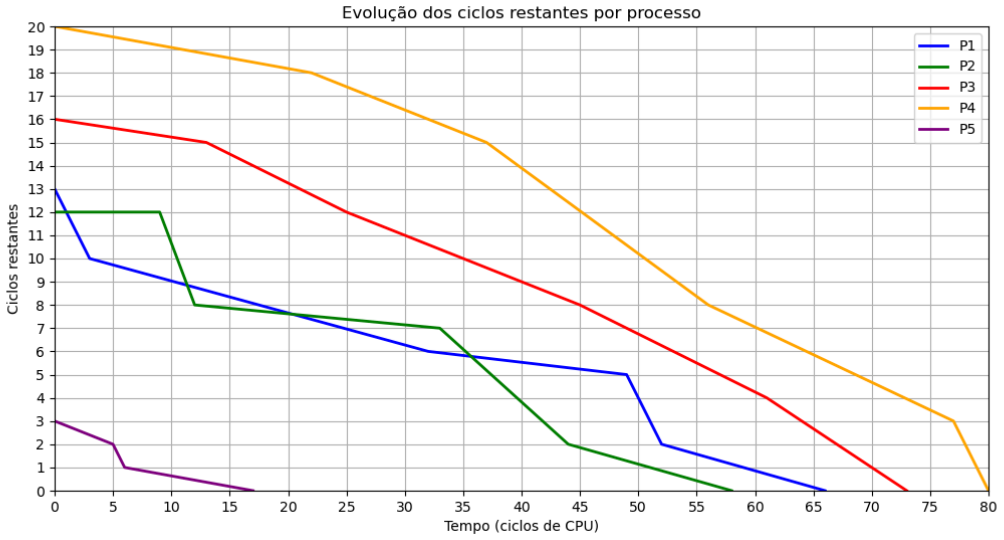
- **Preempções**
- **Envio de processos para a fila de I/O**
- **Retorno de processos da I/O para suas respectivas filas**
- **Períodos em que a CPU está ociosa, aguardando novos processos ou liberação da I/O.**

5.1. Testes com o simulador

Segue a tabela de dados dos processos inseridos manualmente para testes, os processos escolhidos foram selecionados para demonstrar diferentes comportamentos do simulador.

	Ciclos até a chegada	Ciclos totais ativos	Tipo de I/O solicitado	Ciclos até I/O	Possui I/O
Manual-1	0	13	Disco	3	Sim
Manual-2	7	12	Fita magnética	6	Sim
Manual-3	12	16	Impressora	3	Sim
Manual-4	18	20	Nenhum	-1	Não
Manual-5	2	3	Fita magnética	2	Sim

O gráfico abaixo mostra o comportamento do processo durante os ciclos da simulação, o log completo da simulação acima estará num log.txt para ser analisado.



Legenda:

P1 = Manual-1
P2 = Manual-2
P3 = Manual-3
P4 = Manual-4
P5 = Manual-5

```
                    Exemplo de trecho de saída
=================================================================================

=+=+=+=+=+=++=+=+=+=+=++=+=+=+=+=++=+=+=+=+=++=+=+=+=+=+=+=+=+=+=+=+=+
                         CICLO ATUAL: 66 | QUANTUM: 3
*********************************************************************************
FILA DE ALTA PRIORIDADE: Fila vazia, nada a imprimir.
FILA DE BAIXA PRIORIDADE: [| ID: 22982 | ID: 8824 ]
FILA I/O: Fila vazia, nada a imprimir.
*********************************************************************************
#################################################################################
                         Executando Processo ID: 29972
#################################################################################
Ciclos totais: 13 | Ciclos restantes: (1 - 1) -> 0
Possui I/O: Verdadeiro
Tipo de I/O: Disco | Ciclos I/O: 0 | Ciclos restantes para I/O: 0
#################################################################################
Processo ID: 29972 | Situação: Finalizado | Removido | Quantum utilizado: 2
#=#=#=#=#=#=#=#=#=#=#=#=#=#=#=#=#=#=#=#=#=#=#=#=#=#=#=#=#=#=#=#=#=#=#=#=#=#=#=#=
```

```
                   Saída de estatísticas do código
=================================================================================
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                              Estatísticas Finais
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Total de ciclos executados: 80
Total de processos finalizados: 5
Preempções registradas: 11
Operações de I/O realizadas: 4
Ciclos com CPU ociosa: 16
Ciclos médios por processo: 16,00
Throughput / Processos por ciclo: 0,062500
Processos que retornaram da I/O: 4
Tipos de I/O utilizados:
  - Disco: 1
  - Fita magnética: 2
  - Impressora: 1
```

6. Conclusão

Desenvolver um simulador de escalonamento de processos foi uma experiência desafiadora e, ao mesmo tempo, enriquecedora. O projeto nos permitiu compreender de forma prática como o sistema operacional realiza a gerência de múltiplos processos concorrentes, tomando decisões constantes sobre uso de CPU, filas de prioridade e dispositivos de Entrada e Saída. A implementação do algoritmo Round Robin com Feedback evidenciou a importância da organização das filas e da estratégia de

preempção para garantir uma distribuição justa e eficiente dos recursos computacionais.

Durante o processo, foi possível observar o impacto direto de escolhas como o valor do quantum, a prioridade de retorno após operações de I/O e o controle de ociosidade da CPU no desempenho geral do sistema. Indicadores como o número de preempções, o tempo médio por processo e o throughput — definido como a razão entre processos finalizados e o tempo total de execução — serviram como métricas valiosas para avaliar a eficiência do nosso simulador. Em uma das execuções, por exemplo, foram finalizados 50 processos em 636 ciclos, resultando em um throughput de aproximadamente ~0,0786 processos por unidade de tempo, um valor aceitável considerando o comportamento randômico das entradas.

Mais do que codificar, o trabalho exigiu tomada de decisões coerentes, análise crítica dos resultados e depuração de casos especiais que poderiam comprometer a execução do simulador. A necessidade de modularização do código, gerenciamento de múltiplas filas e validação de eventos concorrentes simulou de forma reduzida os desafios enfrentados por sistemas reais.

Ao final, o conhecimento adquirido ultrapassa o conteúdo técnico: aprendemos a trabalhar em equipe, dividir responsabilidades, iterar soluções e discutir diferentes abordagens. Sem dúvida, essa experiência ampliará nosso entendimento em disciplinas futuras e nos prepara melhor para os desafios da área de sistemas e desenvolvimento de software.

7. Referências

Conteúdo da disciplina ICP246 Arquitetura de Computadores

Slides de Escalonamento de processos:

➔ unid03_1_escalonamento.pdf - Google Drive

Dúvidas em C:

➔ C reference - cppreference.com