[[1]](#footnote-1)

Simulador de Escalonador de Processos: Relatório de Implementação

Henrique S. Pinheiro, Sávio O. Camacam

*Resumo*—Este relatório detalha o processo de implementação de um simulador para escalonamento de processos de sistemas operacionais. As descrições apresentadas nesse trabalho tratam do funcionamento de algoritmos específicos aplicados nos três principais modelos de sistemas operacionais para um ambiente de simulação controlado.

*Abstract*—Este relatório detalha o processo de implementação de um simulador para escalonamento de processos de sistemas operacionais. As descrições apresentadas nesse trabalho tratam do funcionamento de algoritmos específicos aplicados nos três principais modelos de sistemas operacionais para um ambiente de simulação controlado.

*Palavras-chave*—Escalonamento, Sistemas Operacionais, Filas de Escalonamento, Processos, Chaveamento, Tabela de Processos.

# introdução

E

esse documento relata o percurso feito na implementação de módulos que foram necessários para executar cinco políticas de escalonamento usadas em Sistemas Operacionais em Lote e Sistemas Operacionais Interativos. As políticas implementadas foram: *Round Robin, SJF (Shortest Job First), FCFS (First Come, First Served), Múltiplas Filas e Random.* As implementações foram feitas com base num Modelo de Simulador, com Estruturas previamente definidas em módulos, fornecido previamente, onde o seu funcionamento geral e as estruturas usadas são descritas e explicadas em tópicos próprios. Pela necessidade de implementação das particularidades de cada política, foi reservado um espaço para descrever o funcionamento de cada uma delas de acordo as modificações necessárias. Módulos adicionados e a interface entre o modelo original são descritos logo em seguida e por fim são apresentados dados com informações de desempenho de cada uma das políticas com os testes realizados seguidos das nossas conclusões sobre o desempenho dessa atividade, a complexidade envolvida nos cálculos e sincronização do escalonamento entre múltiplas políticas e o aprendizado com a modularização do problema.

# O Simulador

O simulador funciona com a execução de um programa em torno dos seguintes módulos:

* Módulo de Leitura de Arquivos;
* Módulo de Blocos de Controle de Processos;
* Módulo de Políticas de Escalonamento;
* Módulo de Eventos;

Esses grupos de funções abrigam as estruturas necessárias para a implementação das políticas.

A simulação se inicia com a leitura do arquivo de experimento passado como parâmetro para o simulador e partir dele, o arquivo de processos é carregado com os eventos que ocorrerão na sua descrição.

Na inicialização do simulador todos os processos são instanciados numa lista de processos novos com variáveis de contagem inicializadas, mas além dessa lista, são criadas outras duas onde são mantidos os processos representando seus estados, como Lista de Processos Bloqueados, que são aqueles à espera de uma solicitação de CPU e Lista de Processos Prontos, como aqueles à espera para usar a CPU, assim o simulador se mantém em execução na sequência de chaveamentos de processos e troca de eventos até que essas listas se esvaziem ou que não haja nenhum processo mais em execução.

No primeiro ciclo de execução, depois da inicialização de todos os registros do arquivo de processos como novos, o primeiro processo está pronto para ser escalonado e quando é feita a verificação se o tempo de entrada do primeiro registro é igual ao tempo do relógio contando naquele instante, o mesmo registro é movido de novos para prontos.

Caso haja algum processo em execução, seu tempo de execução continua sendo atualizado e em cada tempo de relógio, é feita a verificação se para aquele processo há algum evento programado para aquele dado tempo, onde se espera um evento de Término ou de bloqueio.

Para a identificação de um Evento de Término do processo, é feita uma chamada para a função de término do processo dada aquela política, então o processo recebe o valor do relógio indicando o momento em que foi encerrado e assim é feita a atualização do acumulador dos tempos de retorno de todos os processos e então, uma string definida para exibição da sequência de término dos processos é concatenada com o PID desse último processo encerrado a ser exibido no final da execução do experimento.

# Estruturas

Para arquivos a serem lidos – Arquivo de Experimento e Arquivo de Processos – temos a estrutura que contém nome da Política, o Arquivo de Processos, nome do Arquivo de Saída e o ponteiro para o Algoritmo de Escalonamento, para o primeiro arquivo e Número de Processos e um vetor de ponteiros para Blocos de Controles de Processos dos processos lidos no arquivo.

A estrutura politica\_t é a mais importante por conter os ponteiros de função de manipulação dos eventos para chaveamento e tem a seguinte estrutura:



# Divisão de Tarefas

Contando que um dos componentes da equipe era familiarizado com o conceito de programação em módulos e conhecedor prévio do problema para escalonamento de processos, optamos pela divisão dos trabalhos de implementação para um que se sentia mais à vontade com implementação solo e a escrita e relatoria do processo para outro.

Não se excluindo do processo de implementação, a relatoria do trabalho ocorreu concomitantemente com a implementação, uma vez que, de outra forma seria inviável, captar e registrar as dificuldades ocorridas.

# Implementação das Políticas

Em termos de hierarquia e prioridade de processos, em sistemas operacionais somos introduzidos a um princípio que trata da diferenciação entre mecanismo e política de escalonamento.

Uma vez que impreterivelmente, os processos são escalonados pelo núcleo, processos de usuários podem controlar como seus processos filhos serão escalonados a partir de políticas próprias que são passadas ao núcleo e executadas por ele.

## Round Robin

Caracterizado para operar em sistemas operacionais interativos ou servidores, o Round Robin também conhecido como algoritmo por chaveamento circular, trata o processo atribuindo-lhe um intervalo de tempo para uso da CPU de tamanho fixo, o quantum.

No simulador essa política tem cinco assinaturas de funções para manipulação dos processos:

No loop principal há a chamada de execução da política selecionada no experimento – callback – executada por uma função TICK para cada política, e embora ela não tenha desempenhado nenhuma função nas demais políticas, nessa, porém, foi de suma importância por ser a responsável pela contabilização do *timeslice* de cada processo com base no quantum passado como parâmetro.

No Round-robin, a cada contagem de tempo do relógio do processador, o processo que tem a atenção da CPU tem seu tempo de controle decrementado e nessa situação podem ocorrer duas situações: a primeira, o processo é interrompido e há um chaveamento para outro processo e a segunda, seu timeslice é encerrado e na função RR\_tick o processo em execução é inserido na lista de prontos e é aberto o chaveamento para um novo processo.

A criação de qualquer política passa por uma função comum que busca pelo arquivo de experimento e identifica qual é a política. No round-robin, ao acionar a função de criação da política, os parâmetros correspondentes são passados, como: a inserção de um novo processo no vetor de ponteiros de processos do Round-robin, a função tick, a função de finalização de processo e a função de desbloqueio do processo.

Na função RR\_escalonar, é retornado um *BCP (Bloco de Controle de Processos)* como o processo em execução no loop principal, e nessa função o programa verifica se a quantidade de blocos de processos é igual a zero, o que significa que não há nenhum processo carregado para aquela política, retornando assim NULL.

De outra forma, através de um índice que percorre todos os blocos no vetor de ponteiros de blocos do Round-robin, é verificado se o PID daquele bloco está na lista de bloqueados e sendo negativo esse resultado, é afirmativo que ele estará na lista de prontos e assim o processo é removido dessa lista e é retornado para o loop principal do programa para contabilização dos eventos correspondentes e atualização do seu tempo de execução.

## SJF (Shortest Job First)

Esse é um algoritmo que supõe previamente que são conhecidos todos os tempos de execução. Otimizado para sistemas em lote, o *Shortest Job First*, ou Tarefa Mais Curta Primeiro, trata da execução dos processos menores em primeira ordem, considerando que todos têm a mesma prioridade um processo mais curto não interfere em um mais lento e assim o tempo médio de retorno é menor.

## FCFS (First Come, First Served)

Provavelmente é o algoritmo mais simples na implementação do escalonamento, sendo basicamente uma fila de processos em que são escolhidos aqueles em sua ordem se chegada. Diferenciando do Round Robin em termos de tempo de execução, esse é um método em que o processo não tem um tempo fixo de uso da CPU podendo fazer uso dessa por quanto tempo seja necessário.



Na função de escalonamento para o *FCFS*, é feita uma iteração entre todos os processos na lista de prontos procurando por aquele com menor tempo para entrada, que no final é removido da lista e retornado como processo em execução.

## Múltiplas Filas

Aprimorado para a resolução de um problema de chaveamento de processos entre a memória principal e o disco, esse método atribui tempos de execução referentes a uma classificação específica de prioridades dada a uma faixa de processos semelhantes.

A política de múltiplas filas, por fazer uso da chamada das políticas existentes agrupando processos em níveis prioridade foi a mais complexa, tanto em termos de escalonamento, como na criação da própria política.

Na criação da política, com a passagem do arquivo de experimentos como parâmetro, a função inicializa uma estrutura de bloco de controle de processo para Múltiplas Filas que contém um vetor de ponteiros de políticas, para que cada posição receba uma nova política criada a cada linha encontrada no arquivo de Experimento.

Assim, é feita uma varredura desse arquivo, coletando as quarenta políticas indicadas e a cada linha é feita a verificação do nome da política dentre as quatro demais e assim é acessado na estrutura de blocos própria para múltiplas filas, uma posição das políticas instanciadas para a criação da política correspondente chamando a criação da própria função já implementada.

Devido a modularização adotada com a passagem de um arquivo como parâmetro para a criação da política em específico, notamos que é efetivamente usado apenas na criação da política Round-robin a para as demais, é um parâmetro totalmente dispensável e nesse quesito, a verificação de criação do Round-robin foi a única a ter um tratamento diferenciado devido a questão de leitura do arquivo. Como essa é a única política que espera um parâmetro do arquivo, é necessária sua abertura para leitura desses dados, então foi criado um arquivo temporário com a gravação do parâmetro e esse arquivo temporário foi passado para a criação da política.

Como essas políticas já são implementadas e para tanto, executam seu próprio escalonamento, fica a cargo do FCFS a iteração entre processos a serem escalonados a partir de sua faixa de prioridades. Ao ser encontrar a primeira ocorrência daquela faixa, a política é retirada do vetor de ponteiro de políticas e executada.

## Random

Seu funcionamento é simples: um processo aleatório é escolhido entre os processos prontos para serem executados. No bloqueio, o processo é colocado na lista de BLOQUEADOS e um outro processo aleatório é executado. No desbloqueio, o processo é simplesmente colocado na lista de processos prontos.

A manipulação dos processos pela política *Random* se dá pelo uso das funções cujas assinaturas são:



Como não há mudanças nas funções triviais de cada política, a distinção entre elas fica a cargo da função POLITICA\_escalonar. No caso da Política Random, com a escolha de um processo aleatório da sua lista de processos usamos a biblioteca *time.h* com a função *rand* gerando um valor aleatório no intervalo de 0 ao tamanho da lista de prontos que podem ser escalonados.

# Novos Módulos

Esse tópico ficou reservado para descrição das alterações que foram feitas na configuração original. No programa principal por exemplo, foi inserido a impressão das informações do diagrama de eventos requerido na descrição do projeto.

O diagrama é uma *string* para exibição da sequência em que os eventos de todos os processos acontecem. Dentre criação do processo, com a entrada do processo pela primeira vez na execução, bloqueio, desbloqueio e término, é adicionado a esse diagrama, o momento no relógio em que ele acontece e o identificador do processo.

O Tempo Médio de Espera, é um cálculo resultante da média de chaveamento dos processos numa política, calculado como o somatório dos tempos no relógio de entrada e reentrada dos processos dividido pela quantidade de processos. Esse valor é demonstrado ao final da execução do experimento.

O tempo de retorno é estatisticamente o tempo médio do momento em que uma tarefa em lote é submetido até o momento em que ele é terminado. Ele indica quanto tempo, em média, o usuário tem de esperar pelo fim do trabalho. Esse valor está sendo calculado a cada evento de término do processo, o somatório acumulado dos processos com a diferença do tempo da última execução com o tempo da primeira execução daquele processo.

A vazão é a quantidade média de tarefas que o sistema termina num intervalo de tempo, no nosso programa é dado pela soma da quantidade de processos executados sob a contagem da média de vazão.

# Aproveitamento de Módulos

## Overflow do Contador

Parágrafo 1.

## Redundância na Busca de Prontos

Na interpretação do código, tomando por base a implementação já feita do algoritmo Round-robin, identificamos certa redundância na busca pelo processo a ser escalonado em duas listas sem chegarmos a um consenso do porque a busca se sucedeu dessa forma.

Ao ser realizada a iteração por todos os processos no vetor de ponteiros de blocos dessa política é tomado um bloco de cada vez dada a posição encontrada naquele momento e com esse bloco, com chamadas da função auxiliar LISTA\_BCP\_buscar, é passado o PID do processo, mas a ser buscado na lista de bloqueados, e só depois é feita a busca na lista de prontos, o que analisamos ser preciso que se houvesse algum processo a ser escalonado, esse já estaria na lista de prontos sem a necessidade de uma verificação a mais.

## Leitura de Arquivos de Processos

Parágrafo 1.

## Carregamento de Prontos

Parágrafo 1.

# Conclusão

Conclusão do trabalho aqui.

Apêndice

Apêndice aqui.

Agradecimentos

Agradecimentos aqui.

Referências e Notas de Rodapé

Referências aqui.

# Referências

1. Esse trabalho foi submetido como objeto de avaliação da disciplina de Sistemas Operacionais do curso de Bacharelado em Ciência da Computação da Universidade Tecnológica Federal do Paraná. [↑](#footnote-ref-1)