ULBRA CENTRO UNIVERSITÁRIO LUTERANO DE PALMAS

Recredenciado pela Portaria Ministerial nº 1.162, de 13/10/16, D.O.U. nº 198, de 14/10/2016
AELBRA EDUCAÇÃO SUPERIOR - GRADUAÇÃO E PÓS-GRADUAÇÃO S.A.

Luan Borghesan

OSLIVE: simulação do algoritmo de múltiplas filas do módulo de escalonamento de processos

Palmas – TO

2020

Luan Borghesan

OSLIVE: simulação do algoritmo de múltiplas filas do módulo de escalonamento de processos

Trabalho de Conclusão de Curso (TCC) II elaborado e apresentado como requisito parcial para obtenção do título de bacharel em Sistemas de Informação pelo Centro Universitário Luterano de Palmas (CEULP/ULBRA).

Orientador: Prof. M. Madianita Bogo Marioti.

Palmas – TO

2020

Luan Borghesan

OSLIVE: simulação do algoritmo de múltiplas filas do módulo de escalonamento de processos

Trabalho de Conclusão de Curso (TCC) II elaborado e apresentado como requisito parcial para obtenção do título de bacharel em Sistemas de Informação pelo Centro Universitário Luterano de Palmas (CEULP/ULBRA).

Orientador: Prof. M.e Madianita Bogo Marioti.

Aprovado em: _____/_____/_____

BANCA EXAMINADORA

_____

Prof. M.e Madianita Bogo Marioti

Orientador

Centro Universitário Luterano de Palmas – CEULP

_____

Prof. M.e Cristina D'Ornellas Filipakis

Centro Universitário Luterano de Palmas – CEULP

_____

Prof. M.e Fabiano Fagundes

Centro Universitário Luterano de Palmas – CEULP

Palmas – TO

2020

AGRADECIMENTOS

Agradeço primeiramente a Deus, por iluminar meus passos nesta caminhada e permitir que eu chegasse até aqui.

Agradeço aos meus pais e aos meus irmãos pelo apoio e incentivo desde sempre, pelos momentos que estivemos reunidos, as visitas inesperadas ou até mesmo quando estavam apenas de passagem, foram ótimos momentos. A minha companheira, que esteve firme ao meu lado, mesmo nos momentos difíceis.

Agradeço aos professores que tive a oportunidade de conhecer e trabalhar junto, e também por todos os ensinamentos e oportunidades que recebi. De modo especial, minha orientadora que sempre foi muito paciente e dedicada.

Enfim, agradeço a todos os que colaboraram direta ou indiretamente com esse projeto.

RESUMO

BORGHESAN, Luan. **OSLIVE: simulação do algoritmo de múltiplas filas do módulo de escalonamento de processos**. 2020. 40 f. Trabalho de Conclusão de Curso (Graduação) – Curso de Sistemas de Informação, Centro Universitário Luterano de Palmas, Palmas/TO, 2020.

O OSLive é um ambiente de estudo *online*, que utiliza de recursos como simulação, exercícios, gamificação etc., para auxiliar os alunos da disciplina de Sistemas Operacionais a compreender o conteúdo aplicado com mais facilidade. Este trabalho apresenta um projeto de inclusão da simulação do algoritmo de múltiplas filas no Módulo de Simulação dos Algoritmos de Escalonamento de Processos do OSLive. Além de tornar o ambiente de estudo mais completo para o aluno, a simulação do algoritmo de múltiplas filas proporcionará a visualização de como as políticas de escalonamento de processos são implementadas em conjunto, que é a forma como os SOs utilizam em suas políticas de escalonamento. A fim de sustentar a ideia de incluir a simulação das múltiplas filas, é apresentado um breve referencial teórico, que aborda o uso da simulação para auxiliar no processo de aprendizagem de conteúdos complexos e apresenta uma seção sobre Sistemas Operacionais, com foco nos algoritmos de escalonamento, em especial o de múltiplas filas. A metodologia deste projeto consiste no estudo dos conceitos referentes ao escalonamento de processos, na elaboração da lógica para o algoritmo da simulação, na implementação desse algoritmo e nos testes de funcionalidades. Os resultados apresentam a nova versão da aplicação, com a simulação do Algoritmo de Múltiplas Filas, além de explicar a lógica do algoritmo criado para a simulação de escalonamento de processos. A aplicação possibilita a visualização dos algoritmos básicos trabalhando em conjunto, o que auxilia na compreensão dos conceitos abordados na disciplina de sistemas operacionais.

PALAVRAS-CHAVE: Sistemas Operacionais, Algoritmos de Escalonamento de Processos, Simulação Computacional.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CEULP/ULBRA	Centro Universitário Luterano de Palmas
CPU	Unidade Central de Processamento
E/S	Entrada e Saída
FIFO	First In First Out
RR	Round Robin
SJF	Shortest Job First
SO	Sistema Operacional

SUMÁRIO

1 INTRODUÇÃO

Um sistema operacional (SO) funciona como um intermediário entre o usuário e o *hardware* do computador, coordenando a execução dos programas que executam tarefas para o usuário, além de gerenciar os recursos disponíveis (SILBERSCHATZ et al., 2004). Ao fornecer a base para a execução dos programas, espera-se que o SO ofereça uma interface amigável e, com isso, o usuário possa executar suas tarefas sem maiores dificuldades.

Dada a importância de um SO para o contexto da computação, a disciplina de Sistemas Operacionais está presente em grande parte dos currículos dos cursos da área da computação do Brasil, por ser uma disciplina recomendada pelo MEC (2012). Essa disciplina é fundamental para entender o funcionamento de um SO e como este realiza o gerenciamento do computador, o que torna o conteúdo da disciplina bastante complexo, uma vez que para compreender o funcionamento do SO é necessário entender diversos algoritmos que representam as ações realizadas pelos SOs.

A disciplina engloba diversos conteúdos referentes ao funcionamento de um SO, que atua em quatro áreas de gerenciamento: Gerência de Entrada e Saída, Gerência de Memória, Gerência de Arquivos e a Gerência de Processos. Essas áreas de gerência englobam os mecanismos dos SOs responsáveis pela gestão dos recursos de *hardware* disponíveis.

Na área de Gerência de Processos está compreendido o escalonamento de processos que, segundo Stuart (2010), é o mecanismo utilizado para a escolha do próximo processo que será executado pela *CPU*. Durante seu ciclo de vida, os processos prontos para executar ficam esperando sua vez na Fila de Aptos (SILBERSCHATZ et al., 2004), sendo escolhidos por meio de critérios definidos nas políticas de escalonamento dos SOs, que são baseadas nas definições conceituais dos algoritmos de escalonamento.

Os principais algoritmos de escalonamento de processos são: algoritmo *First In First Out* (FIFO), que escolhe os processos por ordem de chegada; algoritmo *Shortest Job First* (SJF), que escolhe o processo com o menor tempo de execução primeiro; algoritmo por Prioridade, que escolhe os processos de maior prioridade primeiro; e algoritmo *Round Robin* (RR), em que os processos são executados pela *CPU* de forma intercalada, executando por fatias de tempo pré-definidas, chamadas de *quantum*.

Compreender esses algoritmos requer do aluno uma boa capacidade de abstração, por envolver conceitos relacionados às rotinas executadas em baixo nível pelo SO, de modo que o usuário que utiliza o computador não consegue visualizar essas execuções, o que dificulta a compreensão dos conceitos. Por exemplo, ao solicitar a execução de um aplicativo, o usuário do computador vê apenas o que acontece no monitor, porém, o SO teve que criar um processo para executar o aplicativo, esse processo disputa os recursos com outros processos como memória e *CPU*, entre outras situações que ocorrem sem que o usuário veja.

Em vista disso, no CEULP/ULBRA está sendo desenvolvido o OSLive, que é uma plataforma *web* que tem como objetivo auxiliar os alunos da disciplina de Sistemas Operacionais no processo de ensino e aprendizagem, disponibilizando recursos como simulações, gamificação e exercícios. O OSLive é composto por módulos, de forma que cada um aborda um conceito específico relacionado a uma das áreas de gerência dos SOs.

Um dos módulos do OSLive que já está implementado é o de Simulação dos Algoritmos de Escalonamento de Processos, que oferece a simulação dos quatro algoritmos básicos de escalonamento. Porém, os SOs implementam suas políticas de escalonamento usando os algoritmos de escalonamento em conjunto, o que, conceitualmente, é classificado como algoritmo de múltiplas filas.

Como mostra o trabalho de Jimoyiannis (2001), que consistiu em realizar um teste de desempenho com dois grupos de alunos, Grupo 'A' e Grupo 'B'. No comparativo dos resultados obtidos, foi possível observar uma superioridade significativa no desempenho dos alunos do Grupo 'B' em que foi utilizada a simulação, de forma que ela foi considerada um fator determinante para esse resultado.

Porém, entender a simulação isolada de cada um dos quatro algoritmos básicos não é suficiente para que o aluno compreenda as políticas de escalonamento dos SOs, que são baseadas em algoritmos de múltiplas filas. Esses algoritmos de múltiplas filas utilizam os algoritmos básicos em conjunto de acordo com a necessidade do sistema. Assim, nota-se a importância de incluir ao módulo de Simulação dos Algoritmos de Escalonamento de Processos, a simulação do algoritmo de múltiplas filas, que é uma ideia simplificada de como as políticas de escalonamento dos SOs usam os algoritmos em conjunto, a fim de possibilitar ao aluno o entendimento de como os algoritmos básicos trabalham em conjunto e, consequentemente, de como os SOs realizam o escalonamento de processos.

Nesse contexto, o objetivo geral do trabalho foi implementar a simulação do algoritmo de escalonamento de processo de múltiplas filas e acrescentar ao módulo de escalonamento de processos do OSLive. Para atingir esse objetivo geral, foram adotados os seguintes objetivos específicos: criar a representação gráfica do escalonamento de múltiplas filas; implementar a simulação do algoritmo de escalonamento de processos de múltiplas filas; integrar a simulação implementada ao módulo de escalonamento de processos do OSLive e; realizar e apresentar os testes funcionais.

2 REFERENCIAL TEÓRICO

Essa seção apresenta o conceito de simulação e como ela pode ser utilizada para auxiliar no processo de ensino e aprendizagem na disciplina de Sistemas Operacionais, além de apresentar os conceitos e os processos relacionados ao funcionamento de um Sistema Operacional.

2.1. Simulação Computacional como Apoio ao Aprendizado

A simulação é uma metodologia para representação de problemas complexos, por meio da reprodução de sistemas já existentes no mundo real ou que ainda estejam em fase de planejamento. Dessa forma, é possível estudar, analisar e avaliar esses sistemas com o objetivo de entender seu comportamento. Além disso, permite testar hipóteses novas e desconhecidas sobre como ou porque determinados eventos ocorrem nos sistemas (SHANNON 1998).

A simulação computacional, segundo De Jong (1991), é um fenômeno, um processo, um sistema ou um aparato que é reproduzido em um modelo implementado como um programa de computador. Na reprodução, a saída do programa deve ser inferida a partir dos dados de entrada do usuário.

Esse recurso tem sido utilizado nas mais diversas áreas do conhecimento, permitindo realizar testes em ambientes simulados e verificar resultados antes de iniciar alguma ação no ambiente real, evitando gastos de recursos uma vez que se tem conhecimento prévio dos resultados. Além disso, a possibilidade de visualizar o funcionamento de sistemas que não seriam visíveis no mundo real, pode contribuir na compreensão de fenômenos acerca destes, para fins didáticos, científicos ou profissionais.

Jimoyiannis (2001) realizou um estudo que teve como finalidade verificar se a utilização de *software* que fornece simulações envolvendo os conceitos de física pode auxiliar os alunos a compreender melhor esses conceitos. Para isso, os grupos ‘A’ e ‘B’ receberam aulas de física mas apenas o grupo ‘B’ recebeu aulas em laboratório com a utilização do *software*. Os resultados obtidos através do questionário foram os seguintes:

- Questão 1: o grupo ‘B’ obteve 7 acertos a mais;

- Questão 2: novamente o grupo ‘B’, obteve 2 acertos a mais;
- Questão 3: dos dois grupos, apenas um aluno do grupo ‘B’ acertou;
- Questão 4: o grupo ‘B’ obteve 5 acertos a mais que o outro grupo;
- Questão 5: apenas 1 aluno de cada grupo acertou.

Com os resultados do questionário é possível perceber um melhor desempenho do grupo que teve acesso às aulas com o *software* de simulação. Esse resultado, reforça a idéia de que poder visualizar um cenário que envolva os conceitos abstratos abordados pela disciplina de SO pode facilitar a compreensão.

Outro exemplo é a utilização da simulação no estudo de sistemas moleculares, em que *softwares* resolvem equações de Newton para representação do movimento dos átomos, apresentando informações sobre a evolução temporal dos sistemas moleculares (MARQUES, 2005). Por meio da simulação da resolução das equações de Newton é possível realizar a ilustração de milhões de átomos e suas interações, o que não poderia ser visto no mundo real.

Araújo et al., (2007) apresenta mais um exemplo de uso de simulação, que fala sobre a utilização da simulação computacional no aprendizado de física, referente a Lei de Gauss para a eletricidade e Lei de Ampère a nível de física em geral. As simulações consistem em representações gráficas de conceitos abstratos como fluxo do campo elétrico e campo magnético para a Lei de Gauss e para a Lei de Ampère, o laço amperiano e a corrente líquida envolvida. Com isso, o aluno pode inferir dados para que sejam geradas novas situações.

Inserido nesse contexto e com foco no ensino e aprendizagem na área da computação, a simulação computacional pode ser utilizada para representar o funcionamento de diversos mecanismos e técnicas. Essa representação permite a visualização e manipulação de ações dos sistemas, que não são visíveis ao usuário o que pode facilitar na compreensão de conceitos.

Como por exemplo, na disciplina de Estruturas de Dados, quando os alunos precisam compreender conceitos abstratos relacionados à estrutura de uma árvore binária de busca, a fim de entender como funciona a organização, armazenamento e recuperação dos dados. Para auxiliá-los a compreender esses conceitos, Silva (2019), desenvolveu uma aplicação *web* que faz a representação gráfica desses conceitos. A aplicação permite que o aluno observe o comportamento de criação e exclusão de nós da árvore e, também, que aprenda conceitos como: profundidade, grau da árvore, quantidade de nós, entre outros.

Outra disciplina na qual a simulação é útil para auxiliar o processo de ensino e aprendizagem é a de Sistemas Operacionais, de forma que um *software* pode representar rotinas que ocorrem em baixo nível, que não são visíveis para o usuário do computador como, por exemplo, o escalonamento de processos, mecanismo de paginação por demanda, entre outros. Dessa forma, a simulação do funcionamento destes mecanismos poderá auxiliar os alunos na compreensão dos conceitos que são vistos em sala de aula, referentes ao funcionamento de um Sistema Operacional.

Portanto, a simulação computacional utilizada como apoio ao processo de ensino e aprendizagem pode trazer resultados bastante positivos. Além de facilitar a compreensão do conceito ou processo que esteja sendo simulado, pode despertar mais interesse por parte do aluno fazendo com ele direcione mais atenção a determinada disciplina.

2.2. Sistemas Operacionais

O Sistema Operacional gerencia a interação entre as aplicações e o *hardware* do computador e, de modo geral, os Sistemas Operacionais executam as solicitações do usuário como, por exemplo, a impressão de um documento. Para isso, o SO oferece mecanismos para realizar o gerenciamento dos recursos disponíveis e produzir o resultado desejado (DEITEL; DEITEL; CHOFFNES, 2005).

Os serviços oferecidos pelo SO são divididos em quatro áreas de gerência. As áreas de Gerência são: Gerência de Memória; Gerência de Arquivos; Gerência de Entrada e Saída e; a Gerência de Processo.

A área de Gerência de Memória é responsável por fornecer os meios necessários para que os diversos processos possam compartilhar a memória de forma segura e eficiente. Por exemplo, organizar o uso da memória, de modo que a memória utilizada por um determinado processo fique protegida para que ela não seja sobrescrita por um outro processo (OLIVEIRA, 2001).

A área de Gerência de Arquivo coordena tudo relacionado a como os arquivos são estruturados, nomeados, acessados, protegidos, implementados etc.. Esses arquivos são unidades lógicas de informações criadas pelos processos. No gerenciamento de arquivos deve-se atender a três requisitos essenciais: permitir armazenar grande quantidade de informações; as informações devem sobreviver ao término do processo que as está utilizando e; permitir múltiplos acessos simultâneos (TANENBAUM e BOS, 2016).

A área de Gerência de E/S controla os muitos periféricos que são utilizados nos computadores, como a interação direta do usuário com *mouse,* teclado, monitor etc., na interação com outros computadores como *modems*, placas de rede etc.. Sendo assim a área de Gerência de E/S está fortemente ligado ao *hardware* do computador (OLIVEIRA, 2001).

É na área de Gerência de Processos que está compreendido o escalonamento de processos, que é o foco do trabalho e, por isso, será explicada com mais detalhes na próxima seção.

2.2.1. Gerência de Processos

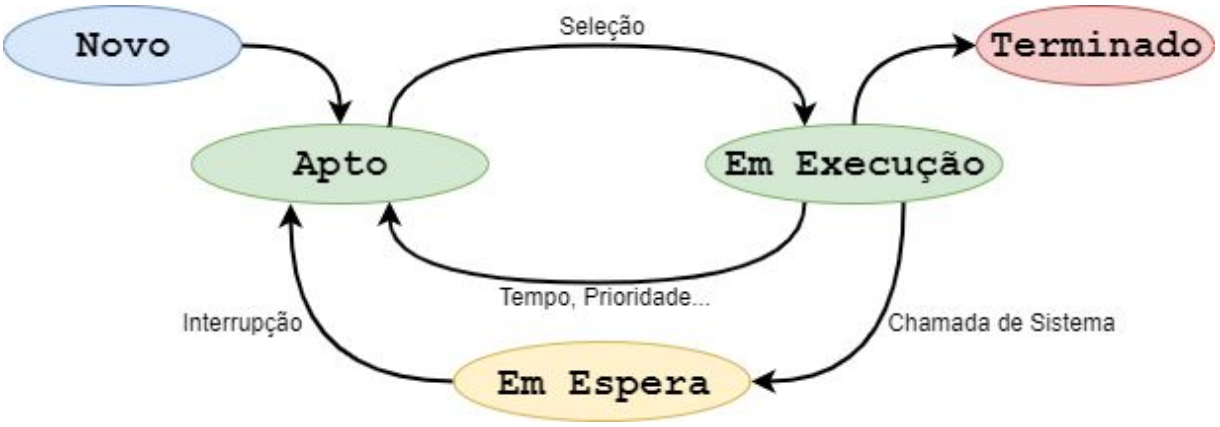
A Gerência de Processos é um dos componentes de um SO responsável por fazer a gestão de todos os processos que executam tarefas do usuário e, também, tarefas do próprio sistema operacional. Assim, é fundamental para o funcionamento dos SOs atuais, que oferecem a possibilidade de multiprogramação.

A maioria dos computadores pessoais modernos oferecem suporte a multiprogramação, que é o fato de o SO permitir que vários processos sejam carregados no sistema concorrendo aos recursos, de forma que pareça que estejam sendo executados simultaneamente (TANENBAUM e BOS, 2016). “Para implementar a multiprogramação, o SO deve repartir os recursos do sistema entre todos os processos.” (TOSCANI, 2003, p. 17).

O conceito de processo surgiu juntamente com a multiprogramação, sendo que um processo é um conjunto de instruções de um determinado programa, ou seja, um programa em execução é um elemento passivo que não altera seu próprio estado, já o processo é um elemento ativo que durante a execução das instruções do programa ele altera seu próprio estado. De modo que um programa pode criar vários processos para atender às solicitações do usuário, por exemplo, ao executar um *browser* são executados vários processos, alguns para baixar as imagens outros para carregar e renderizar os componentes de entrada para o usuário e etc. (OLIVEIRA, 2001).

Os processos concorrem à *CPU* e, durante o ciclo de execução, passam por estados distintos que são definidos como estado de processo. A Figura 1 representa o diagrama de estado de processo.

Figura 1 - Diagrama de Estado de Processo



Fonte: Compilação do autor[1]

Tanenbaum e Bos (2016) descrevem os estados de um processo da seguinte forma:

- Novo: o processo acaba de ser criado;
- Apto: o processo está pronto para ser executado, ele aguardando na fila de aptos a sua vez de ser executado pela *CPU,* que está ocupada com outro processo. A transição do estado apto para o estado em execução ocorre quando o escalonador seleciona um processo para executar na *CPU*, essa escolha é realizada com base em regras definidas nas políticas de escalonamento;
- Em Execução: o processo se encontra na *CPU*, executando suas instruções. Neste estado, o processo poder ser retirado da *CPU* devido a uma chamada de sistema como, por exemplo, uma solicitação de Entrada/Saída, ou por determinação das regras da política de escalonamento utilizada. Pode ocorrer, também, do processo finalizar seu tempo de execução e terminar;
- Em Espera: o processo está bloqueado aguardando a ocorrência de algum evento, como por exemplo um evento de E/S, durante a espera pode ocorrer também uma interrupção e o processo voltar para a fila de aptos. Por exemplo, um processo está esperando uma impressão, quando esta é finalizada gera uma interrupção que notifica o SO que o processo de E/S finalizou e;
- Terminado: o processo finaliza seu tempo de execução, chegando ao fim do seu ciclo de vida.

[1] Alterações feitas a partir de imagem coletada no livro de Silberschatz et al., (2004).

A troca de estados executando para apto é realizada pelo escalonador. De acordo com Stuart (2010), o escalonador é o responsável por selecionar qual próximo processo vai ser executado na *CPU*. Ele também é responsável por ordenar os processos na fila de aptos quando esses saem do estado esperando para apto. Na próxima seção serão apresentados os algoritmos de escalonamento de processos, que determinam os critérios utilizados pelo escalonador para fazer o gerenciamento de uso da *CPU*.

2.2.2. Algoritmos de Escalonamento

O escalonador determina quando e por quanto tempo um processo vai ser executado pela *CPU*, baseando-se em critérios como a prioridade de um processo, o tempo de execução ou até mesmo há quanto tempo o processo está aguardando para ser executado pela *CPU* (DEITEL, DEITEL e CHOFFNES, 2005). São os algoritmos de escalonamento que determinam quais critérios serão utilizados para realizar o escalonamento dos processos.

Os algoritmos de escalonamento podem ser preemptivos ou não preemptivos. Os algoritmos preemptivos permitem que o escalonador retire o processo que está sendo executado pela *CPU*, sem que ele tenha finalizado seu ciclo de execução. Já nos algoritmos não preemptivos, o processo libera a *CPU* quando finaliza seu ciclo de execução ou quando ocorre uma chamada de sistema, como uma solicitação de Entrada/Saída, por exemplo (SILBERSCHATZ et al., 2004).

Existem quatro algoritmos básicos de escalonamento de processos: FIFO (*First In First Out*), SJF (*Shortest Job First*), Prioridade não preemptivo, Prioridade preemptivo e RR (*Round Robin*). Para cada um deles será apresentado um exemplo, de forma que todos os exemplos utilizam os processos informados na Tabela 1.

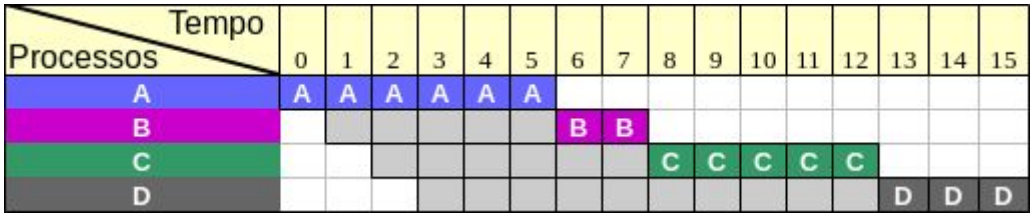
Tabela 1 - Tabela de Processos

Processos	T. Chegada	T. Execução	Prioridade
A	0	6	1
B	1	2	3
C	2	5	0
D	3	3	2

No algoritmo FIFO os processos executam por ordem de chegada, ou seja, o primeiro a chegar é o primeiro a executar, de forma que os processos são inseridos ao final da fila de aptos e para ser executado pela *CPU* é sempre escolhido o primeiro processo da fila. É um algoritmo não preemptivo, no qual os processos deixam a *CPU* quando finalizam seu tempo

de execução ou quando ocorre uma chamada de sistema como, por exemplo, uma solicitação de Entrada/Saída (OLIVEIRA et al., 2009). A Figura 2 apresenta um exemplo da utilização do algoritmo FIFO, para os processos indicados na Tabela 1.

Figura 2 - Escalonamento com Algoritmo FIFO

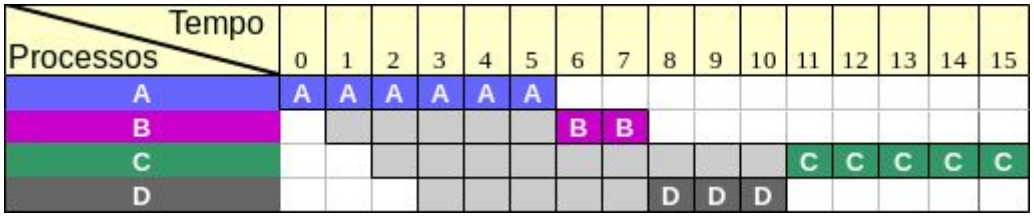


Como é representado na Figura 2, o processo ‘A’ chegou no tempo 0 e foi o primeiro a executar. Durante a execução de ‘A’, os processos ‘B’, ‘C’ e ‘D’ chegaram na fila de aptos, nos momentos 1, 2 e 3 respectivamente, foram inseridos no final da fila e ficaram esperando a sua vez de serem executados. Conforme a ordem de chegada, ao término da execução de ‘A’ o processo ‘B’ foi executado pela *CPU,* em seguida o processo ‘C’ e por fim, o processo ‘D’ foi executado somente quando o processo ‘C’ liberou a *CPU*.

No algoritmo SJF é executado o processo com o menor trabalho primeiro, isso faz com que o tempo médio de espera na fila de aptos seja menor (OLIVEIRA et al., 2009). Os processos são inseridos na fila de aptos de acordo com o tempo de execução, de forma que os processos com o menor tempo de execução serão inseridos no início da fila de aptos.

O SJF é um algoritmo não preemptivo, de modo que o escalonador não retira o processo da *CPU* mesmo que um processo com tempo de execução menor fique apto. Dessa forma, o tempo de execução é levado em consideração apenas para ordenação da fila de aptos. A Figura 3 apresenta um exemplo da utilização do algoritmo SJF, para os processos indicados na Tabela 1.

Figura 3 - Escalonamento com Algoritmo SJF



Como pode ser observado na Figura 3, o processo ‘A’ começou a executar e, como o algoritmo é não preemptivo, ele executou até finalizar seu tempo de execução. Enquanto ‘A’ executava os processos ‘B’, ‘C’ e ‘D’ chegaram na fila de aptos e foram ordenados em ordem

crescente de tempo de execução. Desse modo, o processo ‘B’ executou primeiro por ter o tempo de execução inferior ao tempo de execução dos demais processos, em seguida o processo ‘D’ foi executado, pois o tempo de execução era menor que o tempo de execução do processo ‘C’, sendo este o último a executar.

No Algoritmo de Prioridade, para cada processo é definida uma prioridade e, dessa forma, os processos são inseridos na fila de aptos em ordem decrescente de prioridade, ou seja, processo com prioridade zero são os de maior prioridade, os valores são inversamente proporcionais. Este algoritmo pode ser preemptivo ou não preemptivo, para o algoritmo não preemptivo, o processo em execução não é interrompido mesmo que um processo de maior prioridade fique apto, neste caso, o novo processo é inserido no início da fila de aptos. Já no algoritmo preemptivo, caso um processo de maior prioridade fique apto, o processo em execução é retirado da *CPU* e inserido novamente na fila de aptos, dando lugar ao processos de maior prioridade (SILBERSCHATZ et al., 2004).

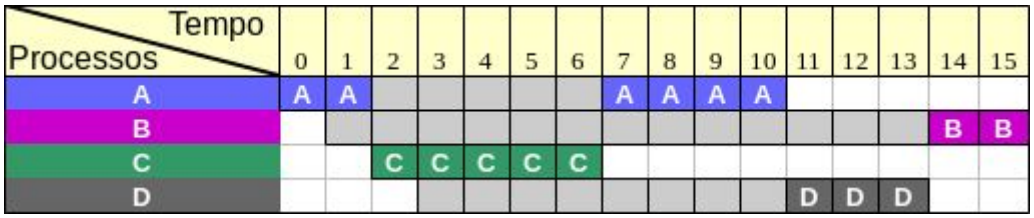
As Figuras 4 e 5 apresentam exemplos da utilização dos algoritmos de prioridade não preemptivo e preemptivo, respectivamente, considerando os processos da Tabela 1. Para estes exemplos considere as seguintes prioridades: processo ‘A’, prioridade 1; processo ‘B’, prioridade 3; processo ‘C’, prioridade 0 e; processo ‘D’, prioridade 2 (quanto menor o número maior a prioridade).

Figura 4 - Escalonamento com Algoritmo de Prioridade não Preemptivo

Tempo / Processos	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A	A	A	A	A	A	A										
B															B	B
C							C	C	C	C	C					
D												D	D	D		

Para o algoritmo de prioridade não preemptivo, Figura 4, pode-se observar que no tempo 0, o único processo na fila de aptos era o processo ‘A’ o qual já foi selecionado para ser executado pela *CPU*. Durante a execução de ‘A’, chegaram na fila de aptos os processos ‘B’ de prioridade 3, ‘C’ de prioridade 0 e ‘D’ de prioridade 2. Pode-se observar que mesmo com a chegada do processo ‘C’ de maior prioridade, o processo ‘A’ não foi interrompido, isso é devido ser um algoritmo não preemptivo. Ao término da execução de ‘A’ o primeiro processo a ser executado pela *CPU* foi o processo ‘C’ por ter prioridade 0, em seguida ‘D’ era o processo de maior prioridade e foi selecionado para ser executado pela *CPU* e, por fim, o processo ‘B’ que foi o segundo a chegar na fila de aptos mas era o de menor prioridade.

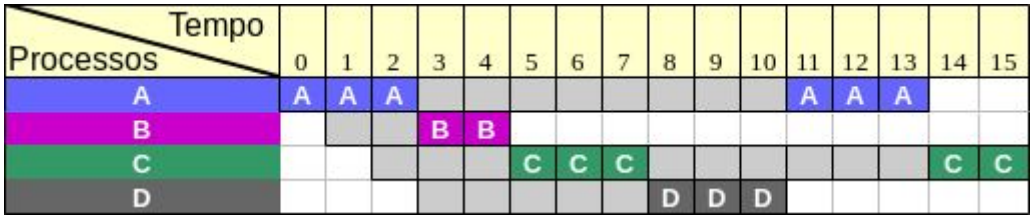
Figura 5 - Escalonamento com Algoritmo de Prioridade Preemptivo



A Figura 5 apresenta um exemplo utilizando o algoritmo de prioridade preemptivo. Como no exemplo anterior, o processo ‘A’, de prioridade 1, era o único na fila de aptos e foi o primeiro a executar. Durante a execução de ‘A’ chegaram novos processos na fila de aptos, no momento 1, chegou o processo ‘B’ de prioridade 3, no momento 2 chegou o processo ‘C’ de prioridade 0. Como a prioridade de ‘C’ era maior que a prioridade de ‘A’, que estava em execução, ocorreu a preempção, em que o processo ‘A’ é retirado da *CPU* e inserido novamente na fila de aptos de acordo com sua prioridade e o processo ‘C’ foi executado pela *CPU,* mesmo que ‘A’ não tenha finalizado seu tempo de execução. Durante a execução de ‘C’, chegou na fila de aptos o processo ‘D’ de prioridade 2, que foi inserido antes de ‘B’ que tinha prioridade 3. Ao término da execução de ‘C’, o processo ‘A’ foi selecionado novamente para ser executado pela *CPU*, por possuir maior prioridade. Após a execução de ‘A’, foi selecionado o processo ‘D’ e, por fim, quando o processo ‘D’ liberou a *CPU* o processo ‘B’ foi executado.

No Algoritmo *Round Robin* (RR), os processos são ordenados na fila de aptos em ordem de chegada, assim como no FIFO. Porém, cada processo recebe uma quantidade limitada de tempo de *CPU,* que é chamada de fatia de tempo ou de *quantum*. Na execução de um processo, caso ele atinja o tempo limite definido pelo *quantum*, mesmo que ele não tenha finalizado seu tempo de execução, ocorrerá uma preempção, de modo que será inserido no final da fila de aptos e o escalonador selecionará o primeiro processo da fila para ser executado pela *CPU* (DEITEL, DEITEL e CHOFFNES, 2005). A Figura 6 apresenta um exemplo da utilização do algoritmo RR, para os processos indicados na Tabela 1.

Figura 6 - Escalonamento com Algoritmo RR



Para o exemplo da Figura 6 considere-se o *quantum* de valor 3. Como se pode observar, os processos executam por ordem de chegada, porém executam por apenas 3 tempos intercalando o uso da *CPU*. Os processos ‘A’, ‘B’, ‘C’ e ‘D’ chegaram na fila de aptos nessa ordem e, por isso, executaram em sequência.

As políticas de escalonamento dos SOs implementam esses algoritmos em conjunto, baseando-se no algoritmo de Múltiplas Filas que será explicado na próxima seção.

2.2.3. Algoritmo de Escalonamento de Múltiplas Filas

A ideia de se ter uma política de escalonamento seguindo os critérios de apenas um algoritmo de escalonamento é inviável, uma vez que os SOs trabalham com situações em que os processos são classificados em diferentes grupos e exigem diferentes tipos de tratamento por parte do escalonador. Por exemplo, no caso do SO *Windows*: os usuários executam vários aplicativos ao mesmo tempo e devem ter a impressão de paralelismo, que é a ideia do algoritmo RR que executa um pouco de cada processo para parecer que estão executando em paralelo; os processos do SO devem ter maior prioridade que os processos de usuários e executar sem interrupções; além disso, entre os processos do SO usa-se o algoritmo FIFO, em que os processos são executados por ordem de chegada.

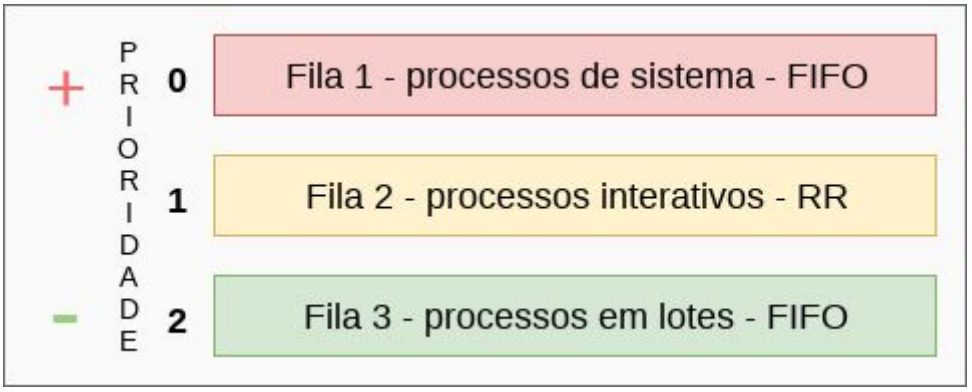
Assim, as políticas de escalonamento trabalham com a ideia de múltiplas filas, em que os algoritmos são usados em conjunto de acordo com a classificação dos processos do sistema. O algoritmo de escalonamento de múltiplas filas trabalha com mais de uma fila de aptos, sendo possível combinar o uso de diferentes algoritmos para melhor atender os diferentes grupos de processos. Com isso, para realizar a inserção dos processos nas filas, são verificadas algumas características do processo como: tamanho de memória, prioridade ou tipo do processo, que vão determinar em qual fila tal processo será inserido.

No escalonamento com múltiplas filas, existe um algoritmo que define o escalonamento entre as filas e outros que definem o escalonamento dentro de cada uma das filas. O algoritmo entre as filas determina quando os processos de cada fila serão executados. Já os algoritmos que definem o escalonamento dentro das filas determinam qual a forma de escolha entre os processos que estão em uma mesma fila.

A Figura 7 apresenta um exemplo de funcionamento do algoritmo de múltiplas filas, em que cada processo possui uma prioridade que determina em qual fila será inserido. Nesse caso, o algoritmo de escalonamento entre as filas é o algoritmo por prioridade preemptivo, no

qual cada fila tem prioridade em relação à fila de menor prioridade (SILBERSCHATZ et al., 2004). Nesse caso, os processos de prioridade 1 só executarão quando a fila de prioridade 0 estiver vazia e, da mesma forma, os processos de prioridade 2 executarão apenas quando a fila de prioridade 1 estiver vazia.

Figura 7 - Escalonamento Múltiplas Filas



Na situação apresentada na Figura 7, a fila 1 é para os processos do Sistema Operacional e no escalonamento dentro dela é utilizado o algoritmo FIFO; a fila 2 contém os processos interativos, que são os processos do usuário, e no escalonamento dentro dela é utilizado o algoritmo RR; e a fila 3 é para os processos em lote e o escalonamento dentro dela também utiliza o algoritmo FIFO. Essa situação é um exemplo simplificado do funcionamento das políticas de escalonamento.

As próximas imagens exibirão um passo a passo do escalonamento de múltiplas filas, apresentado no exemplo da Figura 7: fila 1, FIFO; fila 2, RR; fila 3, FIFO; e o algoritmo para o escalonamento entre as filas será de prioridade preemptivo. Para os exemplos serão considerados os processos da Tabela 2.

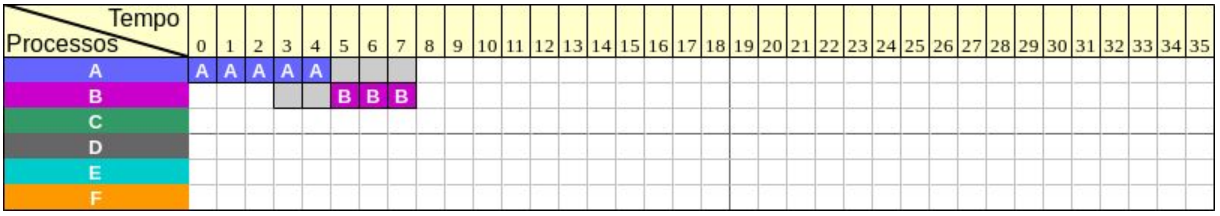
Tabela 2 - Tabela de Processos Múltiplas Filas

Processos	T. Chegada	T. Execução	Prioridade
A	0	8	1
B	3	10	1
C	8	5	0
D	11	4	0
E	13	5	2
F	18	4	1

Os processos são inseridos nas filas de acordo com suas prioridades, de modo que: os processos de prioridade 0, ‘C’ e ‘D’, são inseridos na fila 1; os processos de prioridade 1, ‘A’, ‘B’ e ‘F’, são inseridos na fila 2; e o processo de prioridade 2, ‘E’, é inserido na fila 3. A

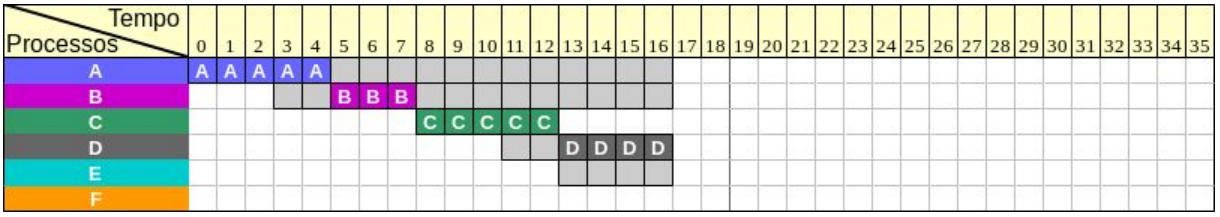
Figura 8 mostra o diagrama de uso da *CPU* até o momento em que ocorre a primeira troca de fila, devido a chegada de um processo com maior prioridade.

Figura 8 - Escalonamento Múltiplas Filas Parte 1



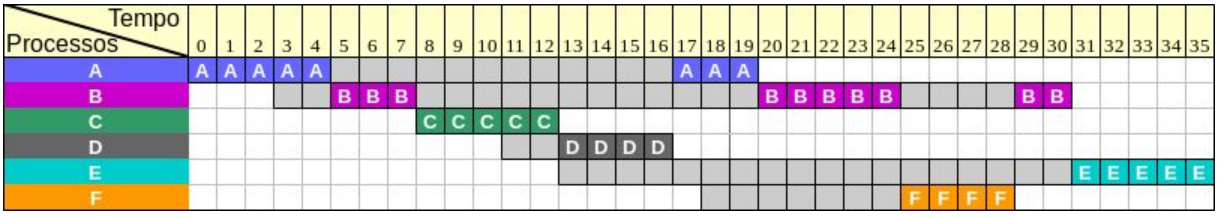
O primeiro processo a executar foi o processo ‘A’ da fila 2, algoritmo RR, que executou por um *quantum* de 5 tempos e, ao atingir o *quantum*, ocorreu a preempção, de modo que o processo ‘A’ foi tirado da *CPU* e inserido ao final da fila 2. Durante a execução de ‘A’, no momento 3, chegou na fila 2 o processo ‘B’ que executou assim que o processo ‘A’ liberou a *CPU*. No momento 8, mesmo sem ter executado todos os 5 tempos do *quantum,* o processo ‘B’ sofreu uma preempção devido a chegada do processo ‘C’ na fila 1, com maior prioridade, Figura 9.

Figura 9 - Escalonamento Múltiplas Filas Parte 2



Com o início do processo C, de maior prioridade, o processo ‘B’ é inserido novamente na fila 2 e o processo ‘C’ é executado. O algoritmo de escalonamento da fila 1 é o FIFO, então o processo ‘C’ executa até finalizar o seu tempo de execução. Durante a execução de ‘C’, no momento 12 o processo ‘D’ chega a fila 1 e é executado logo após o processo ‘C’ liberar a *CPU*. Enquanto ‘D’ executava o processo ‘E’ chegou na fila 3, que possui a prioridade mais baixa das três filas. Com isso ele segue aguardando até que finalize os processos da fila 2, como mostra a Figura 10.

Figura 10 - Diagrama do Escalonamento Múltiplas Filas Completo



Após a execução do processo ‘D’, é retomada a execução dos processos da fila 2 e o primeiro a executar foi o processo ‘A’ que precisou de apenas 3 tempos para finalizar sua execução. Durante a execução de ‘A’, no momento 19, o processo ‘F’ chegou a fila 2 e foi inserido ao final desta. Quando o processo ‘A’ deixou a *CPU*, o processo ‘B’ executou durante todo o *quantum* e então ocorreu a preempção quando ‘B’ foi inserido novamente na fila 2 e o processo ‘F’ executou por apenas 4 tempos do *quantum*, o que foi o necessário para ele finalizar seu tempo de execução. Ao término da execução de ‘F’ o processo ‘B’ foi executado novamente por mais 2 tempos do *quantum* e também finalizou seu tempo de execução. Por fim, não tendo mais processos nas filas de maior prioridade, o processo ‘E’ da fila 3 foi executado.

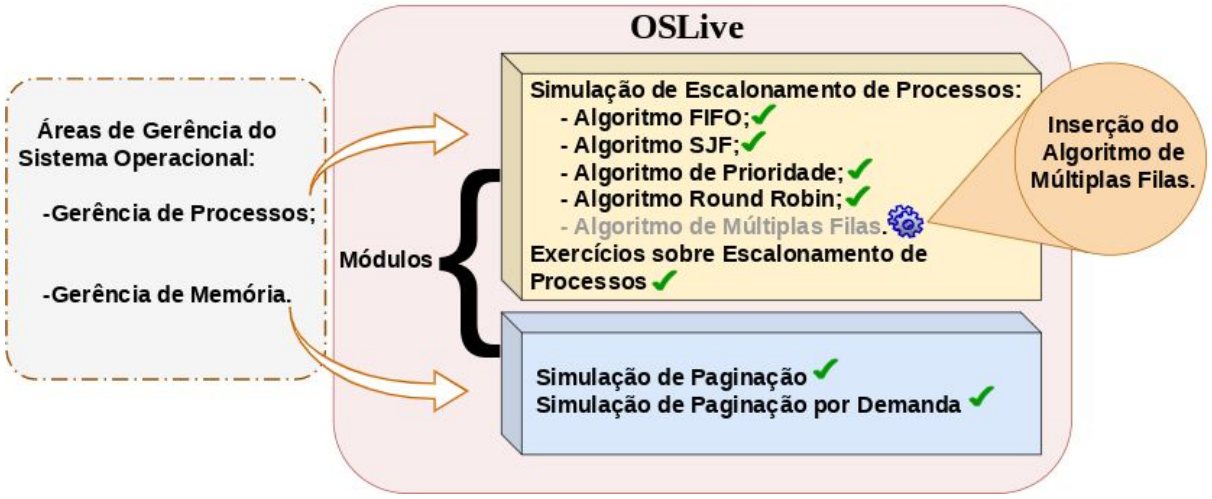
Assim como os livros didáticos, como Oliveira et al., (2009) e Silberschatz et al., (2004), costumam apresentar, a representação mostrou a execução com várias filas de aptos, com o intuito de facilitar o entendimento do algoritmo de múltiplas filas. Porém, na prática os SOs costumam implementar as filas em apenas uma lista encadeada, em que é selecionado sempre o primeiro processo da fila para ser executado pela *CPU*. Os processos são inseridos na lista encadeada de forma ordenada seguindo algum critério pré estabelecido nas políticas de escalonamento.

3 MATERIAIS E MÉTODOS

Essa seção apresenta os materiais, que são as ferramentas, e os métodos, que apresentam o fluxo de trabalho, que serão utilizados para realizar o desenvolvimento da simulação do algoritmo de múltiplas filas e a inserção dessa simulação no módulo de Simulação dos Algoritmos de Escalonamento de Processos, que faz parte do OSLive.

A Figura 11 apresenta a estrutura da plataforma OSLive, indicando as áreas de Gerência que já possuem módulos desenvolvidos e a principal funcionalidade de cada módulo.

Figura 11 - Estrutura do OSLive.



O OSLive disponibiliza quatro módulo em que são abordados conceitos referentes as duas áreas de Gerência de um Sistema Operacional. Sendo uma delas a área de Gerência de Memória em que estão compreendidos os módulos de Simulação da Paginação e Simulação da Paginação por Demanda. A outra é a Gerência de Processos que consiste nos módulos de Simulação dos algoritmos de Escalonamento de Processos que oferece a simulação dos quatro algoritmos básicos de escalonamento de processos e o módulo de Exercícios sobre Escalonamento de Processos.

Como indicado na Figura 11, a simulação do algoritmo de escalonamento de múltiplas filas foi inserida ao módulo de Simulação dos Algoritmos de Escalonamento de Processos Essa simulação é responsável por representar o funcionamento do algoritmo de múltiplas

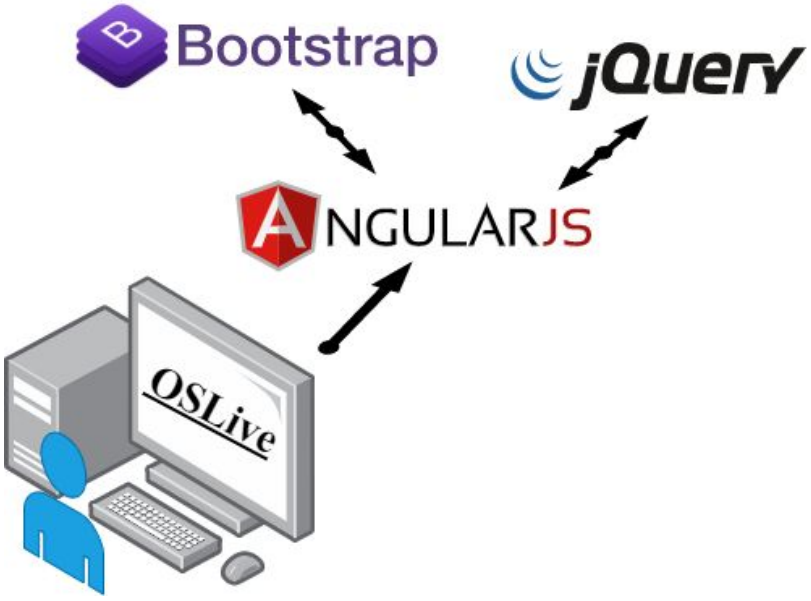
filas. Ou seja, a representação da combinação dos algoritmos básico que os SOs utilizam para gerenciar seus recursos de *hardware*.

3.1. MATERIAIS

As ferramentas que foram utilizadas, tanto para o desenvolvimento do algoritmo que realizará a simulação do escalonamento de múltiplas filas quanto para as alterações na Interface, são:

- Bootstrap: *framework* de código aberto para desenvolvimento de sites e aplicações *web* utilizando HTML, CSS e JS (BOOTSTRAP, 2019). Será utilizado para realizar as mudanças necessárias na interface da aplicação;
- AngularJS: *framework* para desenvolvimento web que oferece uma estrutura adaptativa além de vasto suporte para desenvolvimento de ambientes com visualizações dinâmicas (ANGULARJS, 2019). Será utilizado na implementação da funcionalidade que executará de escalonamento de múltiplas filas;
- JQuery: é uma biblioteca JavaScript que auxilia na manipulação de documentos HTML, manipulação de eventos, animações entre outros (JQUERY, 2019). Será utilizado na implementação das animações.

Figura 12 - Interação das Ferramentas



A interação das ferramentas utilizada no desenvolvimento da nova funcionalidade acontece como representado na Figura 12, de modo que toda a estrutura do projeto foi

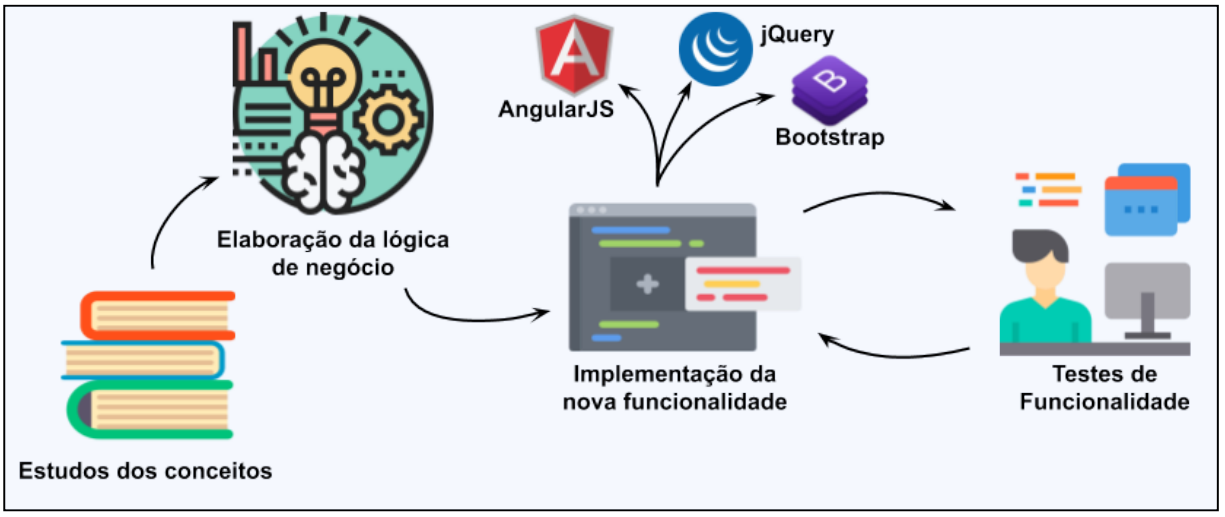
implementada usando o AngularJS, como as funções encarregadas do funcionamento da aplicação e toda a lógica de negócio para que ocorra a simulação.

O Bootstrap e o JQuery estão relacionados com a parte visual da aplicação, sendo que o Bootstrap foi usado para a criação da interface que o usuário utiliza, tornando o ambiente mais agradável. O JQuery foi utilizado para a implementação da animação da simulação, para que o usuário possa acompanhar a evolução de todo o processo de escalonamento de múltiplas filas.

3.2. Métodos

Os métodos definidos neste projeto, para a criação da nova funcionalidade do módulo de escalonamento de processo, foram executados conforme mostra o fluxo de atividades apresentado na Figura 13.

Figura 13 - Fluxo de Atividades



Primeiramente, foi preciso pesquisar e estudar os conceitos referentes ao escalonamento de múltiplas filas, a fim de compreender o seu funcionamento em detalhes e definir a sequência de ações que ocorre durante a sua execução, o que serviu de base para a implementação da simulação. Para esse fim, foram utilizados livros didáticos de Sistemas Operacionais e materiais de aula de professores dessa disciplina. Além disso, foram realizadas reuniões com a especialista da área, professora Madianita Bogo, que ministra a disciplina de Sistemas Operacionais no CEULP/ULBRA, para o esclarecimento de eventuais dúvidas e orientação.

Em seguida, foi planejada a implementação da simulação do algoritmo de escalonamento de múltiplas filas e a sua inserção na estrutura do módulo já existente. Isso inclui a definição da lógica do algoritmo de escalonamento de múltiplas filas e, também, a interação com o usuário para a utilização dessa nova funcionalidade.

A próxima etapa consiste na implementação da nova funcionalidade, que foi definida na etapa anterior, bem como a integração dessa funcionalidade ao módulo atual do ambiente OSLive. Ao inserir a nova funcionalidade, foi necessária a implementação de alterações na entrada de dados do módulo, o que implicou em mudanças no menu de configuração da aplicação.

Após a conclusão da implementação, foi realizada uma série de testes de funcionalidade em nível de teste de unidade sobre a nova funcionalidade da aplicação. Segundo Runeson (2006), o teste de unidade consiste em testar o menor módulo de um sistema separadamente, sendo um teste direcionado por parâmetros de Entrada/Saída fornecidos pelo desenvolvedor. A aplicação de testes unitários tem como objetivo verificar se o funcionamento da nova funcionalidade está de acordo com o proposto.

Além de testes de unidade sobre a nova funcionalidade, foram realizados testes direcionados às funcionalidades já existentes na versão anterior, para garantir que as alterações na inclusão da nova funcionalidade não afetaram a simulação dos demais algoritmos. Com isso, à medida que os erros foram sendo encontrados, eram realizadas as correções e novos testes para verificar se os erros foram corrigidos. As etapas de implementação e teste ficaram em ciclo até que o resultado esperado foi alcançado.

4 RESULTADOS E DISCUSSÃO

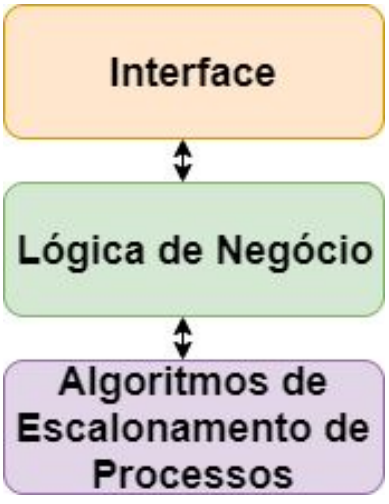
O projeto consistiu na inclusão do algoritmo de múltiplas filas ao Módulo de Simulação dos Algoritmos de Escalonamento de Processo do OSLive. Antes desse projeto, o módulo de simulação disponibilizava apenas as simulações dos quatro algoritmos básicos de escalonamento. Com este trabalho foi inserida a simulação do algoritmo de múltiplas filas, que consiste em combinar diferentes algoritmos de escalonamento, conforme é utilizado pelos SOs nas suas políticas de escalonamento.

Essa seção apresenta a arquitetura da aplicação, que explica as camadas em que ela é dividida, bem como a interface da aplicação. Além disso, apresenta, também, um exemplo de simulação utilizando o algoritmo de escalonamento de múltiplas filas e detalhes de implementação que serão apresentados em um pseudocódigo.

4.1. Arquitetura da Aplicação

A Figura 14 apresenta a arquitetura do Módulo de Simulação de Escalonamento de Processos, que foi implementado em camadas. Para inserir a simulação do algoritmo de múltiplas filas, foi necessário realizar alterações em todas as camadas.

Figura 14 - Arquitetura do Módulo de Escalonamento de Processos



Na camada Interface ocorre toda a interação com o usuário, a qual recebe os parâmetros de entrada que servirão de base para a realização da simulação, apresenta a

simulação dos algoritmos de escalonamento e os resultados estatísticos apresentados para cada processo.

A camada Lógica de Negócio é responsável pela comunicação entre a camada de Interface e a camada de Algoritmo de Escalonamento de Processos, realizando gestão das informações, de modo a atender as solicitações feitas pelo usuário na camada de Interface. A partir da solicitação do usuário é ela que coordena a interação, como, a chamada dos métodos que realizam o escalonamento e o retorno com os resultados.

Algoritmos de Escalonamento de Processos é a camada responsável por executar de fato a simulação dos algoritmos de escalonamento de processos. Para isso, é aplicado o conjunto de regras e critérios do algoritmo de escalonamento, a partir dos dados de entrada do usuário, gerando os resultados que são apresentados na Interface. Para realizar a simulação do algoritmo de múltiplas filas, foi preciso implementar novos métodos que combinam as regras e critério das políticas de escalonamento FIFO e RR.

4.2. Aplicação

Essa seção apresenta a versão atual do Módulo de Escalonamento de Processos do OSLive, após a inclusão da simulação do algoritmo de múltiplas filas. O módulo atual mantém o *layout* de entrada e saída do módulo original, tendo sido acrescentado somente o necessário para o funcionamento da simulação das múltiplas filas.

4.2.1. Interface

No algoritmo de múltiplas filas são usadas diferentes filas de aptos, nas quais são inseridos os processos que estão prontos para executar e estão aguardando serem escolhidos para serem executados pela *CPU*. Existe um algoritmo que define o escalonamento entre as filas, e para cada uma das filas é definido um algoritmo responsável pelo escalonamento dos processos que estão dentro da fila.

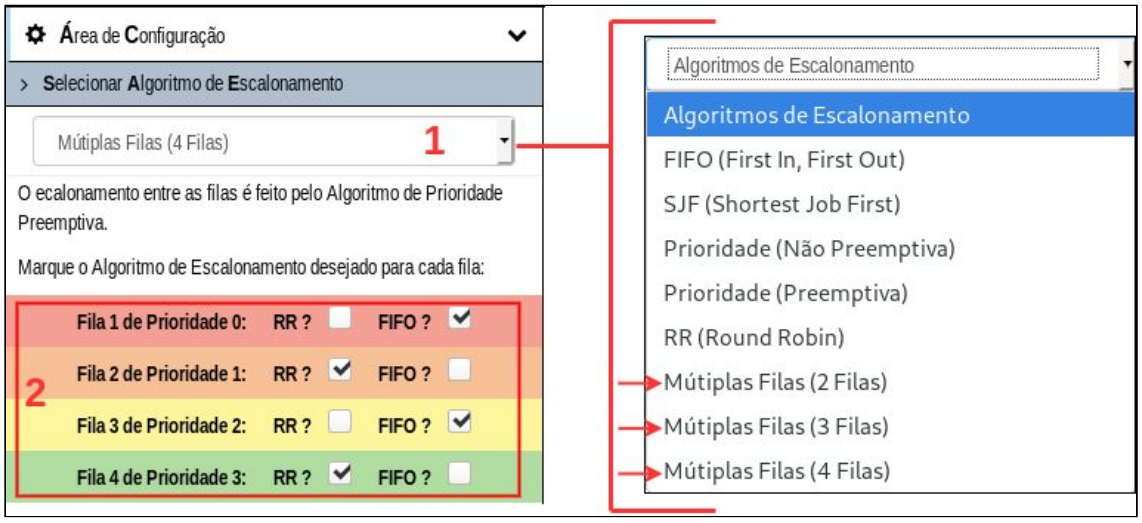
O algoritmo entre as filas determina de qual fila será retirado o próximo processo a ser executado pela *CPU*. SOs como *Linux* e *Windows* utilizam em suas políticas de escalonamento, o algoritmo por prioridade preemptiva, para realizar o escalonamento entre as filas.

Após a escolha da fila, o SO deve escolher qual processo dessa fila será executado pela *CPU*. A escolha desse processo é feita usando os critérios do algoritmo de dentro da fila

em questão. Por exemplo, se vai ser executado um processo da fila com prioridade 1 e dentro dessa fila é utilizado o algoritmo FIFO, os processos dessa fila serão escolhidos em ordem de chegada, até que seja inserido um processo em alguma fila com prioridade maior. Nas filas, SOs como *Windows* e *Linux* usam os algoritmos de escalonamento *First In First Out* (FIFO) e *Round Robin* (RR), de acordo com os processos em cada fila.

Para a simulação do algoritmo de múltiplas filas, definiu-se que, por padrão, o algoritmo para realizar o escalonamento entre as filas é o de prioridade preemptiva e para o escalonamento dentro das filas o usuário pode escolher entre os algoritmos *Round Robin* (RR) e *First In First Out* (FIFO). A Figura 15 apresenta a área de seleção do algoritmo de escalonamento de processos.

Figura 15 - Menu de Configuração Parte 1



Na área 1 da Figura 15, o usuário pode escolher um dos quatro algoritmos básicos para realizar a simulação individualmente, opção já oferecida anteriormente, ou uma das seguintes opções do algoritmo de múltiplas filas: ‘Múltiplas Filas (2 Filas)’, ‘Múltiplas Filas (3 Filas)’ e ‘Múltiplas Filas (4 Filas)’. Assim, o usuário pode escolher de duas a quatro filas para a simulação. Definiu-se até quatro filas pelo fato de que com essa quantidade é possível compreender bem o funcionamento do algoritmo, que é o objetivo da simulação oferecida. Além disso, ao trabalhar com um número maior de filas, isso poderia gerar confusão no usuário e seria necessário mais espaço disponível em tela para organização das filas.

Ainda na Figura 15 área 1, o usuário escolheu a opção ‘Múltiplas Filas (4 Filas)’ e, para cada fila, foi indicada sua prioridade automaticamente e as opções dos algoritmos que podem ser escolhidos para executar dentro de cada uma delas. A área 2 da imagem apresenta

o que aparecerá sempre que o usuário escolher uma das opções do algoritmo de múltiplas filas. Para realizar a escolha do algoritmo, basta selecionar um dos *checkbox* indicados com 'RR ?' ou 'FIFO ?'. Após a definição do algoritmo que será simulado, é apresentada a área de cadastro de processos, Figura 16.

Figura 16 - Menu de Configuração Parte 2

Na área 2 da Figura 16, o usuário pode definir o valor das 'Fatias de Tempo' que serão utilizadas pelo algoritmo RR na simulação. Essa opção só é oferecida para o usuário caso ele selecione o algoritmo RR, sendo uma opção de 'Fatia de Tempo' para cada fila. Como pode ser visto na área 2 da Figura 15, o usuário escolheu o algoritmo RR para as filas 2 e 4, dessa forma, foi disponibilizada para o usuário a possibilidade de escolher as fatias de tempo dessas filas, área 2 da Figura 16.

A área 3 é o formulário de cadastro de processos, que é apresentado ao usuário caso ele escolha o algoritmo de múltiplas filas para realizar a simulação. Os campos são todos obrigatórios, de modo que o usuário deve preencher o 'Nome Do Processo', 'Tempo De Chegada', 'Tempo De Execução' e selecionar a prioridade do processo no *combobox* 'Selecione a Prioridade', no qual são listadas as 4 prioridades possíveis, sendo que a fila na qual o processo será inserido é definida de acordo com a prioridade dele. Ao concluir o preenchimento do formulário, o usuário tem duas opções, 'Cadastrar', que salva o processo atual, e 'Cancelar', que limpa o formulário de cadastro.

O usuário também pode optar por gerar processo com valores aleatórios para a simulação do algoritmo de múltiplas filas, selecionando o *checkbox* sinalizado por 'Gerar processos com valores aleatórios?' na área 1 da Figura 16. Após a seleção dessa opção, é apresentado um botão 'Gerar Processos', que executa a criação de 4 processos, com todos seus valores definidos aleatoriamente.

4.2.2. SIMULAÇÃO

Essa seção apresenta uma simulação com as seguintes definições: o algoritmo de escalonamento escolhido foi o 'Múltiplas Filas (2 Filas)'; para a Fila 1, de prioridade 0, foi escolhido o algoritmo de escalonamento FIFO; e, para a Fila 2, de prioridade 1, foi escolhido o algoritmo RR com fatia de tempo 3. Na criação dos processos foi utilizado o gerador de processos aleatórios, o qual forneceu os processos apresentados na Tabela 3.

Tabela 3 - Processos da Simulação Múltiplas Filas 2 Filas

Processos	T. Chegada	T. Execução	Fila
A	0	4	2
B	3	5	2
D	4	3	1
E	5	4	1
G	1	6	2
H	8	4	1

A Figura 17 apresenta a tela da aplicação após a realização da simulação, que apresenta as seguintes informações: 'Diagrama de Uso da CPU', no qual fica registrado qual processo utilizou a *CPU* a cada momento; 'Tabela Resultado', na qual é apresentado para cada processo seu 'Tempo de espera', que é o tempo que o processo aguardou na fila até ser escolhido para ser executado pela *CPU,* seu 'Tempo Total', que é o tempo do ciclo de vida do processo, ou seja, é a soma do tempo de espera mais o tempo de execução, além de uma média geral tanto para os valores do 'Tempo de Espera' quanto para os valores do 'Tempo Total'; e, 'Resultado Gráfico', no qual é possível avaliar a relação entre o tempo de execução, o tempo de espera e o tempo total de cada processo.

Figura 17 - Simulação: Múltiplas Filas 2 Filas

Analisando o ‘Diagrama de Uso da CPU’, Figura 17, os processos ‘A’ e ‘G’ da Fila 2 foram executados antes dos processos ‘D’, ‘E’ e ‘H’ da Fila 1, pois até o instante 3 só haviam processos na Fila 2, que utiliza o algoritmo RR com a fatia de tempo definida em 3 tempos. Dessa forma, o primeiro processo a executar na *CPU* foi o processo ‘A’, que executou por uma fatia de tempo (3), foi interrompido e inserido novamente na Fila 2. O processo ‘G’ foi o próximo a ser escalonado para executar, o qual recebeu a mesma fatia de tempo que o processo ‘A’, porém, ele executou apenas 1 tempo e foi interrompido, dado que no instante 4 chegou na Fila 1, de prioridade 0, o processo ‘D’ de maior prioridade.

Os processos ‘D’, ‘E’ e ‘H’ chegaram nos instantes 4, 5 e 8, respectivamente, e foram executados em ordem de chegada pois, a Fila 1 utiliza o algoritmo FIFO. Não havendo mais processos na Fila 1, os processos ‘B’, ‘A’ e ‘G’, da Fila 2, foram executados considerando o algoritmo RR da Fila 2 de prioridade 1, dessa forma o uso da *CPU* fica intercalado entre esses processos, considerando a fatia de tempo 3.

Durante a simulação animada, as filas de aptos são preenchidas de acordo com a situação do momento, de forma que são inseridos os processos que se tornaram aptos e removidos os processos que foram escalonados para executar na *CPU*. Diante disso, foi feito o registro de quatro momentos das filas de aptos, para mostrar sua evolução, apresentados na Figura 18.

Figura 18 - Momentos Filas de Aptos

A Figura 18 apresenta a tela em momentos diversos da simulação, de modo que não se refere a uma sequência passo a passo. Os processos da fila 2 são escalonados de acordo com o

algoritmo RR, com isso são inseridos na fila em ordem de chegada. Os processos da fila 1 são escalonados de acordo com o algoritmo FIFO, por isso, também são inseridos na fila em ordem de chegada.

Na situação apresentada no quadro 1, o processo ‘D’ inicia sua execução interrompendo o processo ‘G’, que é inserido ao final da fila 2, na qual já havia os processos ‘B’ e ‘A’.

No momento indicado no quadro 2, o processo ‘E’, que passou pela fila 1, estava sendo executado pela *CPU* e, durante sua execução, chegou na fila 1 o processo ‘H’. Na fila 2 os processos ‘B’, ‘A’ e ‘G’ continuam aguardando a fila 1 esvaziar para, então, serem executados. Com isso, o usuário consegue visualizar que os processos da fila 2 só serão executados quando a fila 1 estiver vazia.

No momento indicado no quadro 3, o processo ‘G’ está sendo executado pela *CPU* e o processo ‘B’ está aguardando sua vez de ser executado.

No quadro 4, a simulação já está finalizada e é apresentado um histórico das filas de aptos, que traz todas as vezes que os processos entraram e saíram da fila de aptos. Na fila 1 é possível perceber que os processos foram executados em ordem de chegada, o que é uma característica do algoritmo FIFO, o qual foi escolhido para o escalonamento dos processos dessa fila. Já na fila 2, é possível perceber o uso intercalado da *CPU,* o que é uma característica do algoritmo RR, responsável pelo escalonamento dos processos dessa fila.

4.3. Implementação do Algoritmo

Nesta seção é apresentada a explicação da lógica seguida para implementar a simulação do algoritmo múltiplas filas. Para isso, foi elaborado um pseudocódigo, o qual foi separado em trechos que representam como foi implementada cada parte da simulação.

As variáveis criadas para a implementação foram: ‘processoAtual’, que guarda o processo que está sendo escalonado; ‘tempoCPU’ é um contador que representa cada momento da CPU, a cada iteração ele é incrementado; ‘quantum*’* guarda o valor do *quantum* de acordo com a fila do ‘processoAtual’; e, ‘contQuantum’ é responsável por contar os quantos ‘tempoCPU’ passaram, a partir da atualização da variável ‘quantum*’*, o que permite saber se o processo atingiu o tempo definido no *quantum*.

Além de variáveis de tipos primitivos, foram criadas três listas: ‘processosOrdenados’ armazena a lista de processos que ainda não foram executados em ordenados por tempo de

chegada e independente do algoritmo; ‘filaAptos’ armazena os processos que estão prontos para executar na *CPU*; ‘diagramaCpu’, que, a cada iteração, armazena o tempo de *CPU* e o processo que fez uso da *CPU* naquele momento. Apesar de estar sendo representado o algoritmo de múltiplas filas, na implementação é utilizada uma única fila ordenada por prioridade.

Todas as funções que serão apresentas são internas a um laço de repetição que será finalizado quando a listas ‘processosOrdenados’ e ‘filaAptos’ estiverem vazias e a variável ‘processoAtual’ também estiver vazia. A Figura 19 apresenta o trecho do pseudocódigo responsável por inserir os processos na ‘filaAptos’.

Figura 19 - Pseudocódigo: Inserir Processo Fila Aptos

```
1.  Se (processosOrdenados tiver tamanho maior que zero E o primeiro processo da lista
2.          processosOrdenados tiver o tempo de chegada igual a tempoCPU) {
3.
4.      Se (filaAptos tiver tamanho maior que zero)
5.          filaAptos recebe o retorno da função atualizarFilaAptos();
6.      Se Não
7.          inseri na filaAptos o primeiro processo da lista processosOrdenados;
8.
9.      remove o primeiro processo da lista processosOrdenados;
10. }
```

O trecho de código das linhas 4 a 9, Figura 19, só é executado se a condição da linha 1 for verdadeira, ou seja, caso a lista ‘processosOrdenados’ não esteja vazia e o seu primeiro processo tiver o tempo de chegada igual ao ‘tempoCPU’, o que indica que o processo ficou apto naquele momento e então será inserido na lista ‘filaAptos’. Na condição da linha 4, caso a lista ‘filaAptos’ não esteja vazia ela é atualizada, recebendo o retorno da função ‘atualizarFilaAptos()’, que tem o objetivo de inserir o primeiro processo da lista ‘processosOrdenados’ na posição correta da fila de aptos, de acordo com a sua prioridade e retorna a lista ‘filaAptos’ ordenada de acordo com o tempo de chegada e a prioridade dos processos. Caso a lista ‘filaAptos’ esteja vazia, o código a ser executado será o da linha 6, no qual o processo é inserido sem levar em conta ordenação, já que será o único processo na lista ‘filaAptos’. Na linha 9, o processo que foi inserido na lista ‘filaAptos’ é removido da lista ‘processosOrdenados’, para manter o controle dos processos que foram criados mas ainda não inseridos na lista ‘filaAtos’. A Figura 20 apresenta o trecho de código responsável por selecionar o processo a ser escalonado, ou seja, que será executado na CPU.

Figura 20 - Pseudocódigo: Selecionar Processo

```
1.  Se (filaAptos tiver tamanho maior que zero) {
2.      Se (processoAtual for vazio)
3.          processoAtual recebe o primeiro processo da filaAptos;
4.
5.      Se Não Se (tempo executado do processoAtual é igual ao tempo de execução) {
6.          calcula o tempo de espera;
7.          processoAtual recebe o primeiro processo da filaAptos;
8.      }
9.      Se Não Se (primeiro processo da filaAptos tem prioridade maior que o precessoAtual) {
10.         filaAptos recebe o retorno da função atualizarFilaAptos();
11.         processoAtual recebe o primeiro processo da filaAptos;
12.     }
13.     Se Não Se (a fila do processoAtual usa o algoritmo RR E quantum é igual ao contQuantum) {
14.         Se (a prioridade do primeiro processo da filaAptos é igual a do processoAtual) {
15.             filaAptos recebe o retorno da função atualizarFilaAptos();
16.             processoAtual recebe o primeiro processo da filaAptos;
17.         }
18.     }
19.     Se (processoAtual tiver sido atualizado) {
20.         removo o primeiro processo da filaAptos;
21.         quantum recebe zero;
22.         contQuantum recebe zero;
23.     }
24. }
25. Se Não {
26.     Se (processoAtual não for vazio E tempo de executado do processoAtual for igual ao
27.         tempo de execução) {
28.         calcula o tempo de espera;
29.         processoAtual recebe vazio;
30.     }
31.     Se Não se (a fila do processoAtual utiliza o algoritmo RR E o quantum é igual ao
32.         contQuantum)
33.         contQuantum recebe zero;
34. }
```

Na instrução da linha 1, Figura 20, se a lista ‘filaAptos’ não estiver vazia, a execução continua na linha 2. Caso o ‘processoAtual’ esteja vazio, ele recebe o primeiro processo que está na lista ‘filaAptos’. Caso contrário, é executada a linha 5, que verifica se o tempo executado pelo ‘processoAtual’ é igual ao seu tempo de execução, o que indica se o ‘processoAtual’ já finalizou seu tempo de execução ou não, sendo que para cada uma situações ocorre uma execução diferente.

Então, caso o ‘processoAtual’ não esteja vazio, e a condição da linha 5 for verdadeira, o tempo executado pelo ‘processoAtual’ é igual ao seu tempo de execução, o ‘processoAtual’ finalizou seu tempo de execução e então foi escalonado o próximo processo da ‘filaAptos’ para executar na *CPU*. Nesse caso, é realizado o cálculo do tempo de espera (‘tempoCPU’ - (tempo de chegada + tempo de execução)), o qual é inserido na tabela de resultados que é apresentado ao final da simulação. Em seguida, o ‘processoAtual’ recebe o primeiro processo da lista ‘filaAptos’, que é o próximo a executar.

Porém, se a condição da linha 5 for falsa, tempo executado pelo processoAtual' não é igual ao seu tempo de execução, o 'processoAtual' não finalizou seu tempo de execução, é feita a verificação da linha 9. Caso a prioridade do primeiro processo da lista 'filaAptos' seja maior que a prioridade do 'processoAtual', a lista 'filaAptos' é atualizada com o retorno da função 'atualizarFilaAptos()', de forma que o 'processoAtual' passa a ser o primeiro processo da lista 'filaAptos'. Esse trecho indica que ocorreu uma preempção, de modo que chegou na lista 'filaAptos' um processo de maior prioridade e então o 'processoAtual' foi interrompido e inserido novamente na lista 'filaAptos', dando lugar ao processo de maior prioridade.

Caso a prioridade do primeiro processo da lista 'filaAptos' não seja maior que a prioridade do 'processoAtual', é executada a instrução da linha 13, que consiste em verificar se a fila a qual o 'processoAtual' pertence utiliza o algoritmo RR e se o valor do 'quantum' é igual ao do 'contQuantum'. Essa verificação é feita para conferir se o 'processoAtual' já executou todo o tempo definido no 'quantum'. Em caso afirmativo, na linha 14, caso a prioridade do primeiro processo da lista 'filaAptos' seja igual a prioridade do 'processoAtual', a lista 'filaAptos' é atualizada com o retorno da função 'atualizarFilaAptos()' e o 'processoAtual' recebe o primeiro processo da lista 'filaAptos'. Ou seja, o 'processoAtual' atingiu o tempo definido no 'quantum', então, ocorre a preempção de modo que o 'processoAtual' é interrompido e inserido novamente na lista 'filaAptos' dando lugar ao próximo processo da lista 'filaAptos'.

Na linha 19 é realizada a última verificação dentro do condicional da linha 1, sendo verificado se o 'processoAtual' foi atualizado, o que significa que a variável 'processoAtual' recebeu um novo processo. Então, se ele foi atualizado, é removido o primeiro processo da lista 'filaAptos' e as variáveis 'quantum' e 'contQuantum' recebem o valor zero, pois caso a fila do 'processoAtual' utilize o algoritmo RR as variáveis estão prontas para o uso.

Mas, se ao executar a instrução da linha 1, a lista 'filaAptos' estiver vazia, a execução desse trecho de código se inicia a partir da linha 26. Caso o 'processoAtual' não esteja vazio e seu tempo executado for igual a seu tempo de execução, o que indica que o 'processoAtual' finalizou seu tempo de execução porém, não há nenhum processo na lista 'filaAptos' para ser escalonado para executar na *CPU*. Então, é calculado o tempo de espera e o 'processoAtual' recebe vazio. Dessa forma, tem-se um intervalo em que a *CPU* fica ociosa até o momento de chegada do próximo processo, ou seja, nenhum processo foi executado por algum tempo.

Porém, se o tempo executado pelo ‘processoAtual’ não for igual ao seu tempo de execução, então é executada a linha 31, na qual é verificado se a fila a qual o ‘processoAtual’ pertence utiliza o algoritmo RR e se o ‘quantum’ é igual ao ‘contQuantum’. Isso indica que o ‘processoAtual’ já atingiu o tempo definido no ‘quantum’, mas ainda não finalizou sua execução. Não havendo mais outro processo na lista ‘filaAptos’, o ‘contQuantum’ será zerado permitindo que o ‘processoAtual’ execute por mais uma fatia de tempo. A Figura 21 apresenta o trecho de código responsável por inserir um intervalo vazio na lista ‘diagramaCpu’.

Figura 21 - Pseudocódigo: Inserir Intervalo Vazio

```
1.  Se (processoAtual for vazio E filaAptos tiver tamanho zero E lista processoOrdenados tiver
2.          tamanho maior que zero) {
3.      inseri um tempo vazio no diagramaCPU;
4.      incrementa o tempoCPU;
5.      retorna ao passo Inserir processo na fila de aptos
6.  }
```

No trecho apresentado na Figura 21, se o ‘processoAtual’ e a lista ‘filaAptos’ estiverem vazios e a lista ‘processosOrdenados’ não estiver vazia, indica que não existem processos para serem executados. Então, é executado o código que insere um intervalo vazio na lista ‘diagramaCpu’, realiza o incremento do ‘tempoCPU’ e retorna ao passo de inserir um processo na lista ‘filaAptos’. A Figura 22 apresenta o código responsável por atualizar e incrementar o ‘quantum’, além de finalizar a iteração.

Figura 22 - Pseudocódigo: Atualizar e Incrementar o Quantum e Finalizar Iteração

```
1.  Se (o processoAtual foi atualizado) {
2.      verifica se a fila do processoAtual utiliza o algorirmo RR e se for o caso, atualiza o
3.          valor do quantum;
4.  }
5.  Se (a fila do processoAtual utiliza o algoritmo RR)
6.      incrementa o contQuantum;
7.
8.  incrementa o tempo de execução do processoAtual;
9.  inseri um intervalo no diagramaCPU referente ao processoAtual;
```

Na linha 1, Figura 22, é verificado se o ‘processoAtual’ foi atualizado, para que seja conferido se a fila a qual o ‘processoAtual’ pertence utiliza o algoritmo RR e então é atualizado o valor do ‘quantum’ de acordo com o que foi definido no momento da configuração da simulação. Além disso, na linha 5, é realizado o incremento do ‘contQuantum’.

Por fim, as linhas 8 e 9 dispensam verificações pois são instruções comuns independentes de qual processo ou fila a que ele pertence. Na linha 8, é realizado o

incremento do tempo executado pelo ‘processoAtual’ e então é inserido um intervalo na lista ‘diagramaCpu’ referente ao ‘processoAtual’, indicando que aquele processo usou a *CPU* naquele momento.

Essa foi a lógica utilizada para realizar o escalonamento de processos de múltiplas filas, que foi inserido ao módulo de escalonamento de processos do OSLive.

5 CONSIDERAÇÕES FINAIS

O presente trabalho consistiu na criação da simulação do algoritmo de múltiplas filas e a integração dela ao Módulo de Simulação dos Algoritmos de Escalonamento de Processos do OSLive. A integração ao módulo de escalonamento de processos consistiu nas alterações realizadas para adicionar a simulação dos algoritmos de múltiplas filas ao módulo já disponibilizado. A entrada para a configuração da simulação e os resultados são apresentados ao usuário da mesma forma que as simulações anteriores já eram oferecidas.

Durante todo o desenvolvimento do projeto foi necessário um entendimento maior em relação aos algoritmos de escalonamento de processos, principalmente, o algoritmo de múltiplas filas. Além disso, foi necessário um estudo sobre as ferramentas utilizadas para desenvolvimento *web*.

Dessa forma, a versão atual do Módulo de Simulação dos Algoritmos de Escalonamento de Processos do OSLive está condizente com o que foi proposto no início do projeto. Com a aplicação, os alunos da disciplina de Sistemas Operacionais poderão utilizar a simulação do algoritmo de múltiplas filas como suporte nos estudos da disciplina. Ao executá-la, o aluno poderá visualizar o funcionamento dos algoritmos básicos trabalhando em conjunto, com a visualização do diagrama de tempo da *CPU* e das filas de aptos sendo preenchidas passo a passo, de forma animada. Com isso, o aluno poderá associar os conceitos da disciplina ao que acontece na prática, sem ter dificuldades ao tentar abstrair o que acontece durante o escalonamento de processos, o que não é visível para o usuário do computador.

Inicialmente, pensou-se em utilizar a implementação que o módulo já oferecia para a simulação do escalonamento de múltiplas filas. Porém, não foi obtido sucesso devido a forma com que a aplicação foi estruturada, tendo problemas no gerenciamento dos tempos dos processos e na apresentação dos processos passo a passo na fila de aptos, na simulação animada. Assim, após uma reunião com a especialista do domínio, decidiu-se realizar a implementação da simulação do algoritmo de múltiplas filas independente da implementação da simulação dos quatro algoritmos básicos, que já estava disponibilizada no OSLive. Com isso foi criado o código apresentado na seção 4.3, Implementação do Algoritmo.

Por fim, como trabalhos futuros pode-se: acrescentar a possibilidade do usuário testar seu aprendizado através de exercícios; acrescentar explicações do conteúdo para que o usuário possa consultar caso queira entender melhor o funcionamento; e acrescentar o controle de usuários, para que seja possível fazer um histórico de uso. Além disso, é importante fazer uma avaliação do Módulo de Escalonamento pelos alunos da disciplina Sistemas Operacionais, buscando o *feedback* dos alunos.

REFERÊNCIAS

ARAUJO, Ives Solano; VEIT, Eliane Angela; MOREIRA, Marco Antonio. Simulações computacionais na aprendizagem da Lei de Gauss para a eletricidade da Lei de Ampère em nível de física geral. **Revista electrônica de enseñanza de las ciências.** Vigo, España. Vol. 6, n. 3 (2007), p. 601-629, 2007.

ANGULARJS. [internet], 2019. Disponível em: <https://angularjs.org/>. Acesso em: 11 de setembro de 2019.

BOOTSTRAP, Team. [internet], 2019. Disponível em: <https://getbootstrap.com/>. Acesso em: 11 de setembro de 2019.

DEITEL, Harvey M.; DEITEL, Paul J.; CHOFFNES, David R.. **Sistemas Operacionais.** 3. ed. São Paulo: Pearson Prentice Hall, 2005. 784 p. (8576050110). Tradução de: Pearson Education Inc.. Disponível em: <https://bv4.digitalpages.com.br/?term=SISTMAS%2520OPERACIONAIS&searchpage=1&filtro=todos&from=busca&page=_4§ion=0#/legacy/315>. Acesso em: 16 out. 2019.

DE JONG, Ton. Learning and instruction with computer simulations. **Education and Computing**, v. 6, n. 3-4, 1991.

JQUERY. [internet], 2019. Disponível em: <https://jquery.com/>. Acesso em: 11 de setembro de 2019.

MARQUES, Gil da Costa. **Física:** tendências e perspectivas. Editora Livraria da Física, 2005.

MINISTÉRIO DA EDUCAÇÃO CONSELHO NACIONAL DE EDUCAÇÃO. PARECER CNE/CES N° 136/2012: Parecer CNE/CES nº 136/2012, aprovado em 8 de março de 2012 - Diretrizes [...]. Distrito Federal: Ministério da Educação, 2012. Disponível em: <http://portal.mec.gov.br/index.php?option=com_docman&view=download&alias=11205pces136-11-pdf&category_slug=julho-2012-pdf&Itemid=30192>.

OLIVEIRA, Rômulo Silva de; DA SILVA CARISSIMI, Alexandre; TOSCANI, Simão Sirineo. **Sistemas Operacionais.** 2001.

OLIVEIRA, Rômulo S.; CARISSIMI, Alexandre S.; TOSCANI, Simão S. **Sistemas Operacionais-Vol. 11:** Série Livros Didáticos Informática UFRGS. Bookman Editora, 2009.

RUNESON, Per. A survey of unit testing practices. **IEEE software**, v. 23, n. 4, p. 22-29, 2006.

SILBERSCHATZ, Abraham; GALVIN, Peter Baer; GAGNE, Greg. **Fundamentos de Sistemas Operacionais**. LTC Editora., 2004.

SILVA, Gedilson Pessoa da. **Desenvolvimento de uma aplicação web para auxiliar no ensino de Árvore Binária na disciplina de Estrutura de Dados**. 2019. Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação). Centro Universitário Luterano de Palmas, Palmas, Tocantins, 2019. Disponível em: <http://ulbra-to.br/bibliotecadigital/publico/home/documento/690>. Acesso em: 13 set. 2019.

SHANNON, Robert. E. Introduction to the art and science of simulation. In: WINTER SIMULATION CONFERENCE, 98CH36274., 1998, College Station. **Proceedings... .** College Station: IEEE, 1998. p. 7 - 14.

SOUSA, Caio Henrique de. **Implementação de ambiente web para simulação de escalonamento de processos**. 2017. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação). Centro Universitário Luterano de Palmas, Palmas, Tocantins, 2017. Disponível em: <http://ulbra-to.br/bibliotecadigital/publico/home/documento/336>. Acesso em: 16 out. 2019.

STUART, Brian L. **Princípios de sistemas operacionais**: projetos e aplicações. Cengage Learning, 2010.

TANENBAUM, Andrew S.; BOS, Herbert. **Sistemas Operacionais Modernos.** 4. ed. São Paulo: Pearson Education do Brasil Ltda, 2016. 778 p. (9788543005676). Disponível em: <https://bv4.digitalpages.com.br/?term=sistemas%2520operacionais&searchpage=1&filtro=todos&from=busca&page=68§ion=0#/legacy/315>. Acesso em: 13 out. 2019.

TOSCANI, Simão Sirineo; OLIVEIRA, Rômulo Silva de; CARISSIMI, Alexandre da Silva. **Sistemas Operacionais e Programação Concorrente:** Série Livros Didáticos - Número 14. Porto Alegre: Sagra Luzzatto S/a, 2003. 247 p. (85-2410682-4).