Simulador de Rotinas do Sistema Operacional para Auxílio às Aulas Teóricas

Thiago Pirola Ribeiro[1], **Refael Lucas Bernardes**[2], **Edgard Araujo Lobo**[3]

[1]Docente do Curso de Sistemas de Informação – Faculdade de Computação
Universidade Federal de Uberlândia (UFU) – Campus de Monte Carmelo
38.500-000 – Monte Carmelo – MG

[2]Mestrando em Ciência da Computação – Faculdade de Computação
Universidade Federal de Uberlândia (UFU) – 38.400-902 – Uberlândia – MG

[3]Graduado em Sistemas de Informação – Instituto de Ciências Exatas e Tecnológicas
Universidade Federal de Viçosa (UFV) – Campus Rio Paranaíba
Caixa Postal 22 – 38.810-000 – Rio Paranaíba – MG

tpribeiro@fc.ufu.br, rafaelbernardes0@gmail.com, edgard.lobo@ufv.br

Abstract. *This paper addresses the hard understanding of the real dinamism of the computational events in the operating systems course. Two softwares are presented here in order to improve this learning process from the graphical illustration of strategies for managing: (i) the first one simulates the routines of real and virtual memory management and (ii) the second one shows how the process scheduling in architectures with multiprocessors works. Results show the proposed softwares give a great support to the conceptual classes of the operating system course and, consequently, they make the classes more dynamics and allow the simulation of different scenarios.*

Resumo. *Com o intuito de diminuir o impacto negativo causado pela difícil caracterização do real dinamismo dos eventos computacionais abordados na disciplina de Sistemas Operacionais, foram desenvolvidos neste trabalho dois softwares: (i) um para simulação das rotinas de gerenciamento de memória real e virtual e (ii) outro abordando o funcionamento do escalonamento de processos em arquiteturas com multiprocessadores ou com processadores multicore. Como resultado os softwares ilustram de forma gráfica as estratégias para gerenciamento, dando suporte as aulas conceituais da disciplina de Sistemas Operacionais tornando-as mais dinâmicas, além de possibilitar a simulação de diversos cenários.*

1. Introdução

Atualmente a grade curricular de todos os cursos de graduação na área da informática apresenta a disciplina de Sistemas Operacionais, a qual proporciona ao aluno conhecimento sobre a arquitetura de um Sistema Operacional. O aluno passa a entender melhor os recursos internos do Sistema, podendo assim, gerenciar melhor os recursos do computador, e no que se diz respeito à desenvolvimento, poderá criar aplicações que tirem o máximo de proveito dos recursos presentes.

Alguns projetos já realizam simulações de Sistemas Operacionais [Maia 2001], [Reis et al. 2009], [de Carvalho et al. 2006], [Rocha et al. 2004] e [Kioki 2008], buscando de uma maneira sucinta e objetiva, ajudar no ensino das disciplinas que trabalham com esse assunto. Esses autores desenvolveram ferramentas com o objetivo de simular as principais funcionalidades de um sistema operacional, permitindo certa interação com o usuário/aluno de forma a facilitar a assimilação dos conceitos ensinados na disciplina.

Os Sistemas Operacionais Modernos trabalham sobre novas plataformas de hardware, e com isso o desenvolvimento de novas simulações se faz necessário. Esses novos hardwares trabalham com mais de um núcleo e/ou mais de um processador, além de grandes volumes de memória, algo que fica muito abstrato para ser compreendido pelos alunos baseando se apenas na teoria vista em sala de aula.

Para auxiliar nas aulas que envolvem Sistemas Operacionais, mais especificamente a gerência de processos e gerência de memória, inicialmente foi pensado apenas na criação uma animação exemplificando cada algoritmo, porém atualmente, os alunos têm a necessidade de grande interação com o objeto de estudo. Baseando-se nesse princípio, foram desenvolvidos simuladores de rotinas de Sistemas Operacionais Modernos com interface para interação do usuário. O foco deste trabalho foi a gerência de processos utilizando mais de um núcleo e/ou processador e gerência de memória real e virtual. Com uma interface interativa, pretende-se facilitar o aprendizado do aluno através da visualização dos algoritmos explicados pelo professor.

2. Análises Iniciais

Para dar auxílio às aulas conceituais da disciplina é possível utilizar sistemas reais ou simuladores. Os simuladores podem ser classificados em simuladores genéricos que abordam diversos aspectos de um Sistema Operacional e simuladores específicos que simulam apenas um aspecto de Sistemas Operacionais de forma mais aprofundada.

Dentre os sistemas reais, foram analisados:

- **Minix** [Tanenbaum and Woodhull 2006]:
  Sistema Operacional que pode ser utilizado para ensino gratuitamente, sendo compatível com Unix.
- **Linux** [Torvalds 2013]:
  Sistema Operacional suportado por diversas plataformas de hardware e distribuição gratuita.
- **FreeBSD** [The FreeBSD Foundation 2013]:
  Suportado pelas plataformas Intel e Compaq Alpha é gratuito podendo ser utilizado para qualquer aplicação inclusive gratuita.
- **Tropix** [TROPIX 2008]:
  Sistema Operacional desenvolvido no NCE/UFRJ (Núcleo de Computação Eletrônica /Universidade Federal do Rio de Janeiro) e possuí filosofia Unix, sendo sua distribuição gratuita.

Os sistemas reais analisados podem ser utilizados para o ensino gratuitamente, porém não apresentam de forma fácil uma interface para que o usuário consiga visualizar o funcionamento dos algoritmos implementados, obrigando o usuário a interagir com o código fonte que, muitas vezes, está escrito em linguagens como C++, Assembly e Java.

Essa iteração com o código fonte, pode muito mais atrapalhar uma explicação em sala de aula do que auxiliar os alunos na compreensão do algoritmo.

Dentre os diversos simuladores, foram analisados:

- **SOsim**
  Desenvolvido na linguagem Pascal utilizando paradigma de orientação a objetos, por [Maia 2001] como trabalho de mestrado. A ferramenta que possibilita através de uma interface gráfica a assimilação de alguns dos conceitos abordados em sala de aula na disciplina de Sistemas Operacionais.
- **SSOG**
  Desenvolvido por [Kioki 2008] e implementado na linguagem Java utilizando paradigma de orientação a objetos. A ferramenta abrange os temas de gerência de processos, gerência de processadores e gerência de memória, além de possuir um módulo de animação que exibe problemas como o do filósofo, que trata da alocação de recursos críticos compartilhados por mais de um processo. Com relação a memória o SSOG trata apenas da questão da alocação de memória e a demonstração da quantidade de memória utilizada pelos processos e a quantidade de memória livre restante.
- **wxProc**
  Desenvolvido por [Rocha et al. 2004], esta ferramenta que visa o escalonamento em multiplataforma. Busca auxiliar na fixação dos conceitos da disciplina de forma gráfica. Tem uma desvantagem, pois o escalonamento em multiplataforma é uma subárea de gerência de processos, não tem um grande foco nas disciplinas de Sistemas Operacionais.
- **S^2O**
  Desenvolvido por [de Carvalho et al. 2006] como trabalho de conclusão de curso. A ferramenta desenvolvida com objetivo de simular o comportamento de um sistema operacional em relação à gerência do processador. Também auxilia na fixação dos conceitos visto em sala de aula.
- **TBC-SO/WEB**
  Desenvolvido na linguagem de programação Java (*applets*) por [Reis et al. 2009], é simulador de caráter educativo com interface gráfica para utilização via web. Apresenta o conteúdo teórico do evento simulado, porém aborda poucos assuntos referentes a gerência de memória.

A Tabela1 apresenta um resumo dos simuladores analisados. Pode-se observar que nem todos os simuladores analisados realizam a gerência de memória e a gerência do processador, além da técnica de substituição de páginas não ser abordada por nenhum deles.

Tabela 1. Comparativo entre os simuladores analisados

Simuladores	Gerência de Memória			Gerência de Processador	Gerência de Processos
	Alocação de páginas	Swapping	Substituição de páginas		
SOSim	X	X		X	X
SSOG	X			X	X
WxProc					X
S^2O				X	X
TBC-SO/WEB	X				X

Apesar do grande auxílio obtido pelos simuladores analisados, nenhum deles ilustra de forma gráfica e visualmente fácil, o gerenciamento de memória virtual e o gerenciamento de processos utilizando mais de um núcleo e/ou processador.

3. Desenvolvimento

Foram desenvolvidos dois simuladores: uma para a gerência de memória e outro para a gerência de processador. Apesar de terem sido desenvolvidos independentes, a estrutura de controle interna dos dois programas são similares o que facilitará na integração dos dois simuladores, podendo ser incorporado outras funcionalidades e simulações.

Para o desenvolvimento dos simuladores foi utilizada a linguagem Java utilizando paradigma de orientação a objetos com a IDE Netbeans da Sun, juntamente com ferramentas encontradas na biblioteca OpenGL para auxiliar na implementação dos recursos gráficos dos simuladores. Além disso, para um melhor desenvolvimento foram levados em consideração algumas técnicas de IHM (Interface Homem-Máquina) relacionados à estrutura dos diálogos, uso de cores, ícones, gráficos e 3D, comandos e linguagem natural [da Rocha and Baranauskas 2003].

Conforme foram sendo desenvolvidas as telas e simuladores, foram apresentados à três docentes que ministram a disciplina de Sistemas Operacionais que sugeriram algumas alterações para que os simuladores ficassem mais interativos e pudessem melhorar a exemplificação durante às aulas. A alteração de posição e método de exibição das filas dos processos no gerenciamento de processador foi uma dessas sugestões.

3.1. Gerência de Memória

Para o desenvolvimento do simulador de gerência de memória foram analisados diversos métodos de alocação de memória e casos de ocorrência de *page-fault*. Após as análises foram consideradas para o desenvolvido o simulador, as estratégias que mais apareciam nas referências didáticas da disciplina.

Podem ser simuladas as estratégias de alocação: *First-fit*, *Worst-fit* e *Best-fit*. As estratégias de falta de páginas (*Page-fault*) que podem ser simuladas são:

- **FIFO (*First-In-First-Out* - Primeiro a Entrar - Primeiro a Sair)**
  Utiliza um critério simples para substituição de páginas: a página selecionada para ser substituída é a página que está a mais tempo na memória principal [Machado and Maia 2007], ou seja, a primeira página a entrar na memória principal é a primeira página a ser substituída.
- **Relógio**
  O algoritmo FIFO tem como característica negativa: o descarte de páginas que são frequentemente utilizadas pelo sistema. Para solucionar isso é utilizado um bit de referência em cada página, além de organizar todas as páginas que estão na memória em uma lista circular com um ponteiro indicando a página mais antiga [Tanenbaum 2010].
- **WSClock**
  Organiza as páginas em uma estrutura de lista circular com um ponteiro apontando para a página mais antiga. Esse algoritmo faz uso de um bit de referência, um bit de modificação além do campo instante da última referência utilizado no algoritmo Working Set [Tanenbaum 2010].

- **LFU (*Least-Frequently-Used* - Menos Frequentemente Utilizado)**
  Utiliza a frequência de uso das páginas como critério de substituição de páginas. Quando ocorre uma falta de página e consequentemente necessitasse de uma substituição de página, é selecionado para ser substituída a página que foi menos referenciada [Machado and Maia 2007].
- **LRU (*Least-Recently-Used* - Menos Recentemente Utilizado)**
  Utiliza o tempo que a página está sem ser referenciada como critério de substituição de páginas. Quando ocorre uma falta de página e consequentemente necessitasse de uma substituição de página, é selecionado para ser substituída a página que está há mais tempo na memória sem ser referenciada, ou em outras palavras, utilizando esse principio pode-se deduzir que provavelmente uma página que não foi referenciada recentemente não será referenciada em um futuro próximo [Machado and Maia 2007].
- **NRU (*Not-Recently-Used* - Não Recentemente Utilizado)**
  Baseia-se na ideia de que páginas não utilizadas recentemente possuem poucas chances de serem utilizadas em um futuro próximo. Com isso o algoritmo NRU possui dois bits (um de referência e outro de modificação) para realizar o controle de quais páginas devem ser substituídas em caso de falta de página. O bit de referência indica se uma página foi referenciada (lida ou escrita) ou não, já o bit de modificação indica se uma página foi modificada (escrita) ou se permanece igual a quando foi carregada na memória [Tanenbaum 2010].

A Figura 1 apresenta a tela principal do simulador. Na parte superior da janela ficam os locais nos quais o usuário poderá interagir com o software, criando ou excluindo processos, escolhendo a estratégia de alocação e de *Page-Fault* que deseja simular, além de ter a opção de alterar algumas pequenas configurações para a simulação. Ao centro do programa fica a região onde ocorrem as animações do simulador, tendo a esquerda o desenho que representa a memória principal, assim como o tamanho de cada uma de suas páginas e, a direita pode-se observar o desenho que representa a memória secundária, além de um relógio que ilustra o tempo virtual corrente e o valor de τ (conjunto do trabalho). Na parte inferior do simulador é possível visualizar a tabela que mostra todas as características dos processos.

O programa possui a capacidade de simular três diferentes tipos de estratégias de alocação: *First-Fit*, *Best-Fit* e *Worst-Fit*. A estratégia de alocação é ativada automaticamente quando um processo é criado, de forma a alocar o processo na memória principal seguindo os critérios do algoritmo selecionado.

Na parte superior da Figura 2 (parte circulada de vermelho) é possível visualizar o local que o usuário pode selecionar a estratégia de alocação que deseja simular. No exemplo da Figura 2 a estratégia *Best-Fit* foi a estratégia selecionada.

O programa possui 6 opções de simulação de estratégias de *Page-Fault*: FIFO, Relógio, LFU, LRU, NRU e WSClock. O algoritmo selecionado é ativado automaticamente quando é criado um novo processo no programa e, se não existir espaço suficiente na memória principal para a alocação desse novo processo, então seguindo os critérios do algoritmo selecionado pelo usuário o simulador escolhe o processo vítima para deixar a memória principal e se mover para a memória secundária, deixando espaço na memória principal para que o novo processo criado o ocupe. Esses critérios são representados nas

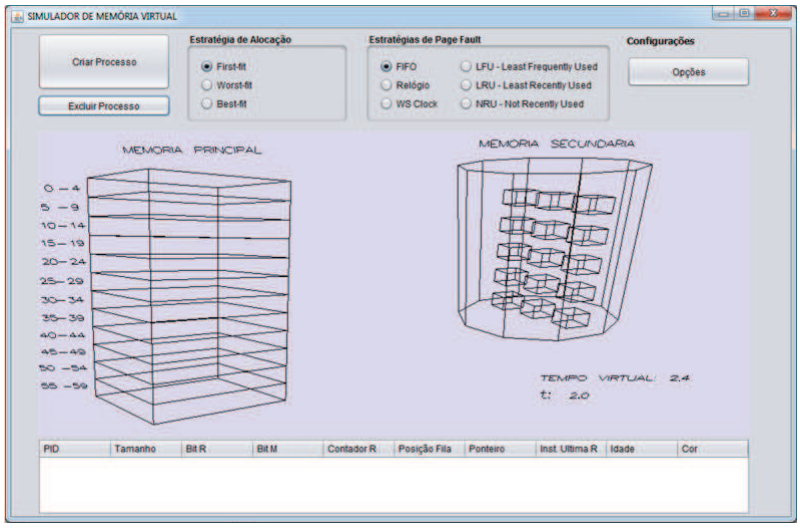

Figura 1. Tela principal do simulador de Gerenciamento de Memória

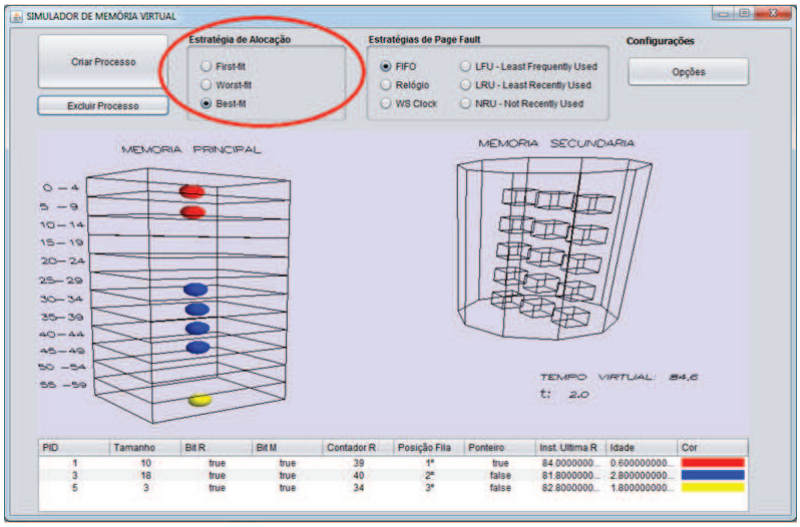

Figura 2. Estratégias de Alocação de Memória

colunas da tabela da parte inferior.

A parte superior direita da Figura 3 (parte circulada de vermelho) mostra o local que o usuário pode selecionar a estratégia de *Page-Fault* que deseja simular. É também possível visualizar que a estratégia selecionada nessa simulação é a estratégia FIFO, então a coluna ”Posição Fila”(circulada de amarelo) é a coluna que será usada como critério para a escolha do processo vítima a sair da memória principal para dar espaço ao novo processo criado.

A Figura 4 mostra o momento posterior a criação de um novo processo na cena ilustrada na Figura 3. O novo processo é representado pela esfera de cor azul e está sendo alocado na memória principal. O processo vítima para sair da memória principal é representado pela esfera de cor vermelha e está sendo alocado na memória secundaria após sair da memória principal deixando uma lacuna para que o novo processo possa ser alocado.

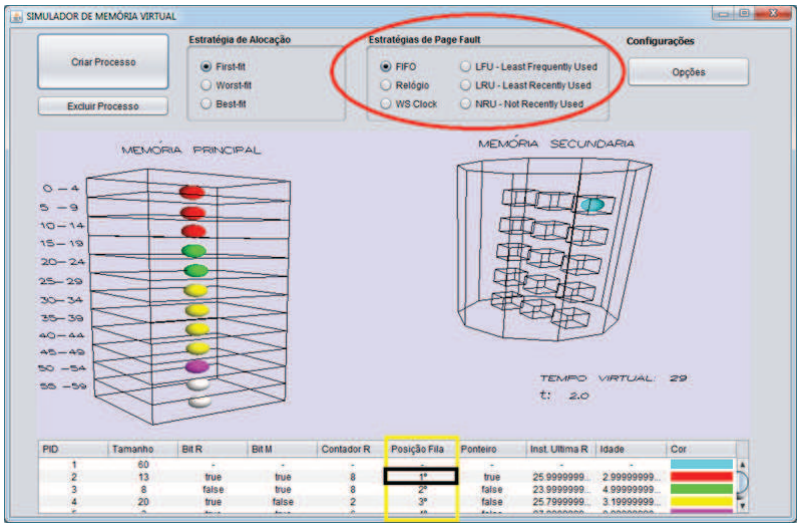

Figura 3. Apresentação *Page-Fault* utilizando a estratégia FIFO

O processo representado pela esfera vermelha de id 2 foi escolhido como vítima, pelo fato de que na Figura 3, que ilustra o momento antes da criação do processo a célula da tabela circulada de preto que é correspondente a posição na fila do processo de id 2 (processo vítima) indica que o processo está na primeira posição da fila, ou seja, seguindo os critérios da estratégia FIFO esse é o processo a ser retirado da memória principal.

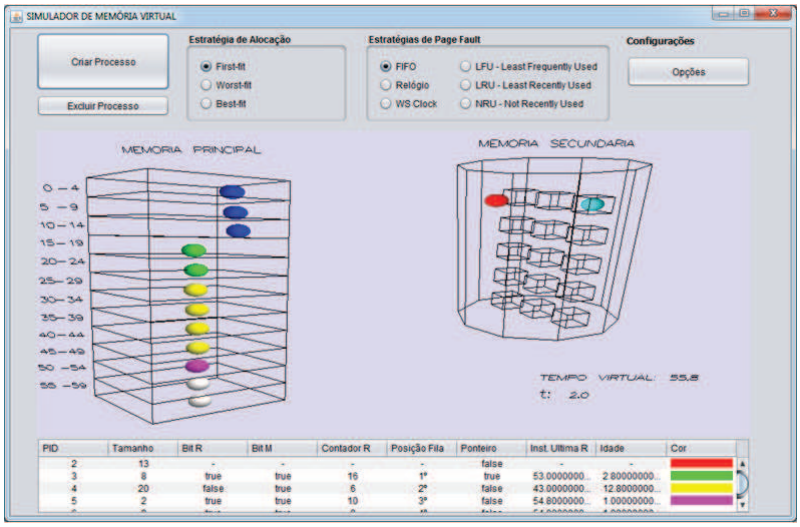

Figura 4. Apresentação *Page-Fault* utilizando a estratégia FIFO - Momento em que ocorre uma falta de página.

O simulador desenvolvido permite que o usuário altere algumas configurações como o valor do Conjunto do Trabalho (τ) e o tamanho da Memória Principal, por exemplo.

3.2. Gerência de Processador

Para o desenvolvimento do simulador para gerenciamento de processos em sistemas com múltiplos processadores foram analisados diversos algoritmos presentes na

literatura. Após as análises foi constatado que os algoritmos de Tempo Compartilhado, de Compartilhamento de Espaço e de Escalonamento em Bando eram os que mais apareciam nas referências didáticas da disciplina. Percebeu-se, também, que não seria viável a implementação dos três algoritmos de escalonamento, pois o Algoritmo de Compartilhamento de Espaço e o de Escalonamento em Bando acabam por ser muito similares: as mudanças de um algoritmo para o outro em relação à processamento são significativas, porém para fins de visualização gráfica, conforme proposto pelo trabalho, o usuário poderia não ter a percepção da mudança entre os algoritmos, podendo prejudicar o aprendizado.

Para se escolher entre os dois algoritmos para ser implementado, foi observado que, durante as pesquisas, o mais referenciado entre os autores e, que havia uma abordagem mais ampla era o algoritmo de Escalonamento em Bando, sendo este o implementado no simulador, além do algoritmo de Tempo Compartilhado.

A ideia do algoritmo de Escalonamento em Bando é fazer com que todos os *threads* de um processo executem juntos, durante uma mesma fatia de tempo (*quantum*), sendo assim, se um desses *threads* envia uma requisição a outro, este receberá a mensagem quase que imediatamente, e também será capaz de responder quase que de imediato. Pode-se perceber que Escalonamento em Bando procura resolver um problema que ocorre no algoritmo de Compartilhamento de Espaço, no qual se tem um desperdício de tempo quando a CPU é bloqueada [Tanenbaum 2010].

Para que o Escalonamento em Bando funcione, todas as CPUs são escalonadas de maneira síncrona, com o tempo sendo dividido em uma fatia de tempo pequena. A cada nova fatia de tempo, as CPUs são reescalonadas de maneira síncrona e com um novo *thread* em cada uma. Se ocorrer de algum *thread* precisar ser bloqueado, sua CPU fica ociosa até o final da fatia de tempo [Tanenbaum 2010].

A Figura 5 apresenta a tela em que o usuário definirá as características dos processos. Nessa tela o usuário seleciona a quantidade de processos que serão simulados e, à medida que o usuário adiciona novos processos, esses são automaticamente inseridos na tabela. Alguns parâmetros (colunas) desses processos inseridos são editáveis (Thread, Tempo, Prioridade e E/S), sendo que o usuário tem a liberdade de definir o processo de acordo com as necessidades de visualização, existindo, também, a possibilidade do usuário selecionar o *CheckBox* de Balanceamento dos Processadores onde a carga de processamento será divida (em uma situação onde se tem 6 *threads* para serem executados, serão executados dois em cada processador). O usuário também pode selecionar Repetir Execução, na qual a simulação repetirá até que o usuário encerre a janela de simulação.

A Figura 6 apresenta a simulação do Algoritmo de Escalonamento Tempo Compartilhado baseada nas características que foram definidas de acordo com a Figura 5. Ao centro da Figura 6 existem as filas de prioridades que são os cilindros (as prioridades podem variar de 0 à 5), os processos que são representados pelas esferas maiores, e os *threads* que são representados pelas esferas menores (presentes dentro dos processos). Abaixo das filas de prioridades existe uma fila para as entradas e saídas, e um cubo que representa que o *thread* está aguardando por um evento de entrada e ou saída.

O lado direito da Figura 6 exibe três quadros que representam os processadores, cada quadro possui quatro cubos os quais representam os núcleos que cada processador

Figura 5. Tela para Criação e Visualização das informações dos Processos.

possui, quando os *threads* vão ser executados eles assumem o formato de esferas maiores, e vão representar o estado de execução quando essas esferas chegarem aos cubos (núcleos), encerrando a execução quando acabarem o tempo de execução.

Pode-se notar que os cinco *threads* do processo P - 0 (azuis) estão ocupando dois processadores, pois cada processador pode possuir até quatro núcleos disponíveis os quais, não são suficientes para execução de todos os *threads* de P - 0, então para que todos os *threads* sejam executados juntos são necessários cinco núcleos que só podem ser obtidos com uso de dois processadores. O mesmo ocorre com P - 1 (verde). O processo P - 2 (vermelho) possui cinco *threads* gastaria cinco núcleos para executar, porém só existem dois disponíveis, o escalonador então selecionará dois dos cinco *threads* de P - 2 para que sejam executados nos núcleos disponíveis e os outros três restantes irão aguardar na fila de espera. Pode-se notar, também, que os processos P - 4, P - 5 e P - 6 encontram se na fila de espera, porém com a prioridade 1 conforme definido, e os processos P - 7, P - 8, P - 9 estão a frente, pois foram definidos com prioridade 0.

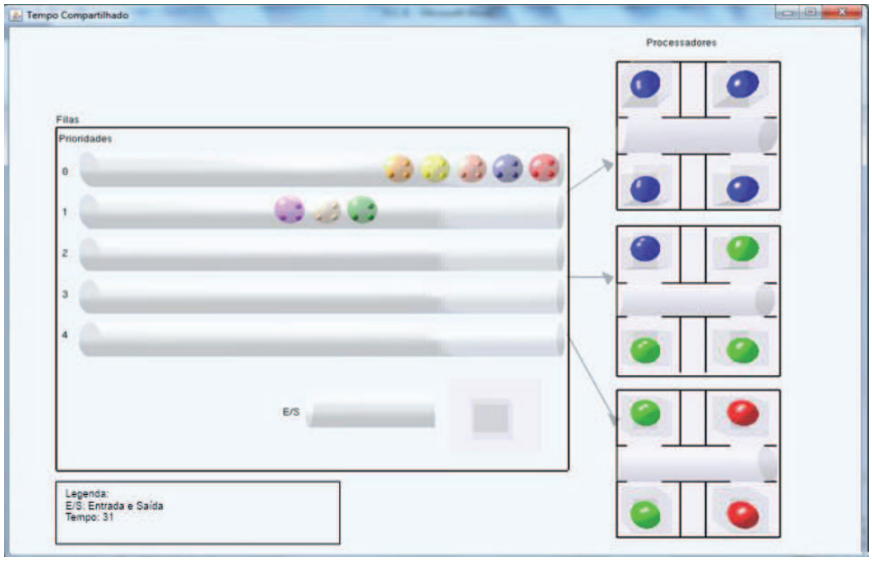

Figura 6. Simulação do Algoritmo Tempo Compartilhado.

A Figura 7 apresenta a simulação do Algoritmo de Escalonamento em Bando, na qual os *threads* de um processo P qualquer só serão escalonados se todos couberem nos núcleos dos processadores, pode-se notar que os processos P - 0 (azul) e P - 1 (verde) com 5 *threads* cada foram escalonados ocupando dez núcleos dos doze disponíveis, ficando assim dois núcleos ociosos, pois o próximo processo o P - 2 (vermelho) também possuí cinco *threads* e, de acordo com as condições do algoritmo, seus *threads* não podem ser escalonados de forma independente como no algoritmo de Tempo Compartilhado.

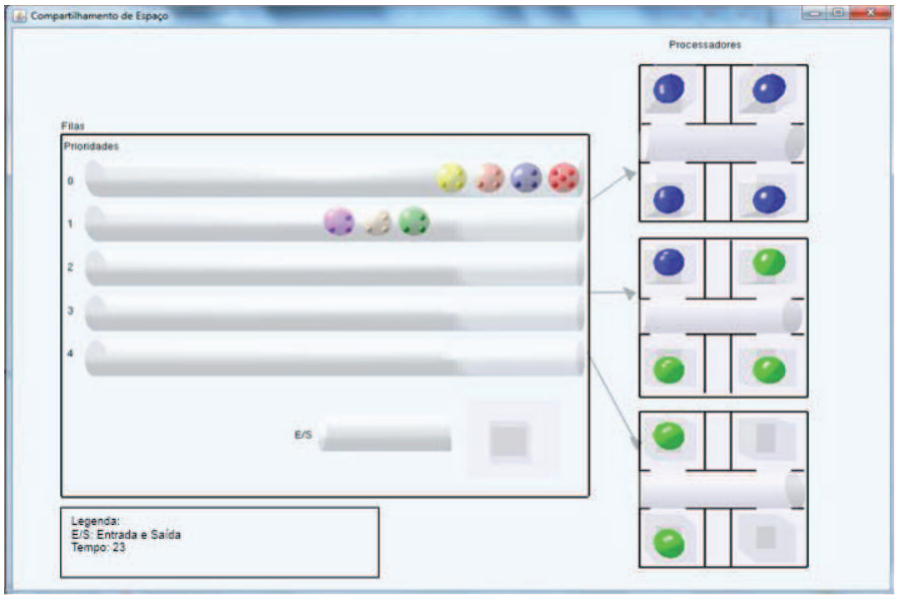

Figura 7. Simulação de Entrada e Saída, Algoritmo Escalonamento em Bando.

Com relação à Entrada e Saída, o algoritmo de Escalonamento em Bando e de Tempo Compartilhado funcionam da mesma maneira, o *thread* que solicita o evento de Entrada e Saída fica ocupando o núcleo até que o evento seja concluído após a conclusão todo o processo com os *threads* relacionados a ele volta para o fim da fila de espera como mostra a Figura 8.

4. Conclusões

Com os resultados obtidos neste trabalho é possível concluir que os simuladores desenvolvidos cumprirão seus papéis de auxiliar nas aulas teóricas de Sistemas Operacionais. Através do simulador o usuário tem a possibilidade de visualizar o comportamento dos processos e seu gerenciamento da memória e do processador, de acordo com as estratégias e situações que o próprio usuário desejar.

Toda a implementação foi feita com o auxilio da biblioteca OpenGL e acabou por tornar as telas, de certa, forma mais atrativa ao usuário, sendo que as simulações apresentam alguns efeitos visuais, os quais podem despertar curiosidade no aluno fazendo com que ele procure utilizar ferramenta.

Algoritmos que, por muitas vezes, são de difícil caracterização do seu real dinamismo utilizando apenas métodos de ensino totalmente teóricos, serão facilmente explicados através das visualizações das simulações, facilitando o processo de aprendizagem do

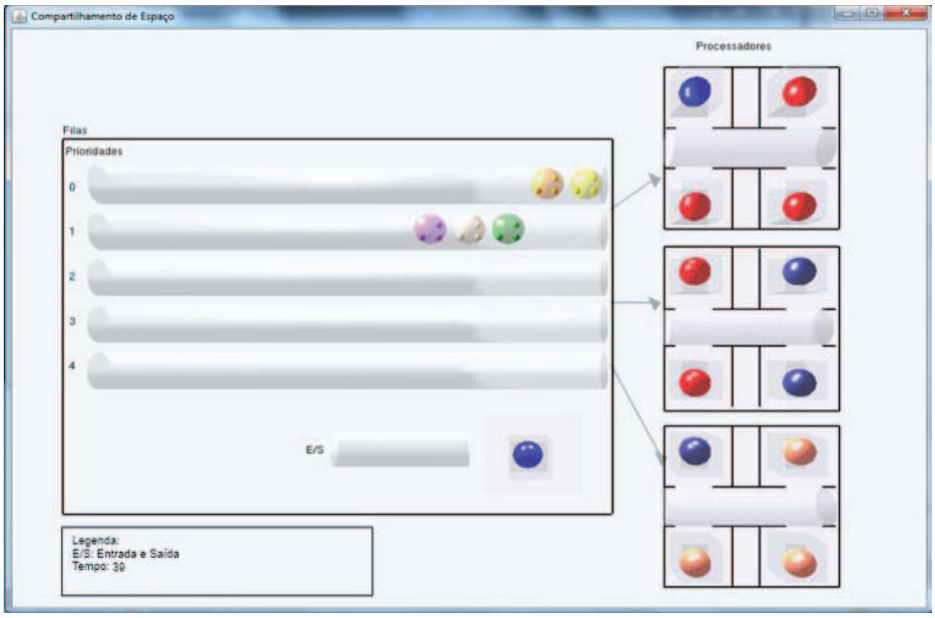

Figura 8. Simulação de Entrada e Saída, Algoritmo Escalonamento em Bando.

conteúdo e, consequentemente, aumentando a eficiência do processo de ensino e aprendizagem entre professores e alunos.

Como trabalhos futuros, o simulador será utilizado em disciplinas de Sistemas Operacionais para validação da melhora no aprendizado. Serão trabalhados dois grupos de alunos, sendo que apenas um desses grupos utilizará os simuladores e ao final serão avaliados os desempenhos dos alunos e estes avaliarão os simuladores.

Referências

da Rocha, H. and Baranauskas, M. (2003). *Design e avaliação de interfaces humano-computador*. Unicamp.

de Carvalho, D. S., da Rocha Balthazar, G., Dias, C. R., Araújo, M. A. P., and Monteiro, P. H. R. (2006). S2o: Uma ferramenta de apoio ao aprendizado de sistemas operacionais. In *XXVI Congresso da SBC – XIV Workshop sobre Educação em Computação (XIV WEI)*.

Kioki, E. Y.; Santiago, P. P. S. A. C. (2008). Um simulador didático como ferramenta de apoio ao ensino da disciplina de sistemas operacionais. *Revista INICIA*, 8(8):41–48.

Machado, F. and Maia, L. (2007). *Arquitetura de sistemas operacionais*. Livros Tecnicos e Cientificos - LTC.

Maia, L. P. (2001). Sosim: Simulador para o ensino de sistemas operacionais. Master's thesis.

Reis, F. P., Parreira Júnior, P. A., and Xavier Costa, H. A. (2009). Tbc-so/web: Um software educacional para o ensino de políticas de escalonamento de processos e de alocação de memória em sistemas operacionais. In *Anais do Simpósio Brasileiro de Informática na Educação*, volume 1.

Rocha, A. R., Schneider, A., Alves, J. C., and Silva, R. M. A. (2004). wxproc: Um simulador de políticas de escalonamento multiplataforma. *INFOCOMP Journal of Computer Science*, 3(1):43–47.

Tanenbaum, A. (2010). *Sistemas Operacionais Modernos*. Pearson Prentice Hall.

Tanenbaum, A. and Woodhull, A. (2006). *Sistemas Operacionais: Projetjos e Implementação*. Editora Bookman.

The FreeBSD Foundation (2013). The freebsd project. `http://www.freebsd.org/`.

Torvalds, L. (2013). The linux kernel. `http://www.kernel.org`.

TROPIX (2008). Projeto tropix - sistema operacional. `http://www.tropix.nce.ufrj.br`.