Simuladores de Gerência de Memória e Processador para Auxílio às Aulas Teóricas de Sistemas Operacionais

Thiago Pirola Ribeiro[1], Rafael Lucas Bernardes Lima[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 as grades curriculares de todos os cursos de graduação na área da informática apresentam 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], [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.

Nas próximas seções serão feitas análises iniciais dos softwares presentes na literatura, a explicação e exemplificação dos softwares desenvolvidos, os experimentos e resultados e as conclusões.

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], Linux [Torvalds 2014], FreeBSD [The FreeBSD Foundation 2014] e Tropix [TROPIX 2008].

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 existentes, 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. Possui 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^{2}O**: Desenvolvido por [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. Apesar do grande auxílio que os simuladores analisados apresentam, 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.

Tabela 1. Comparativo entre os simuladores analisados

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

3. Desenvolvimento

Foram desenvolvidos dois simuladores: um 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 futura integração dos dois simuladores, podendo ser incorporadas 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

utilizadas algumas técnicas de IHM (Interface Homem-Máquina) relacionados à estrutura dos diálogos, uso de cores, ícones, gráficos e 3D, comandos e linguagem natural [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 posição e o 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.

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 uma tabela que mostrará todas as características dos processos.

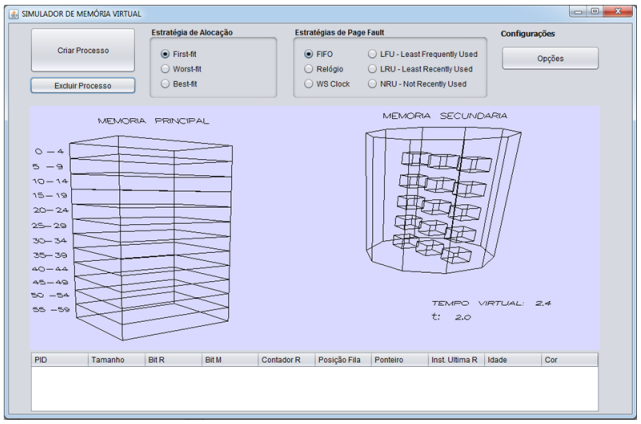

Figura 1. Tela principal do simulador de Gerenciamento de Memória

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. Além disso, possui 6 opções de simulação de estratégias de *Page-Fault*: FIFO (*First-In-First-Out* - Primeiro a Entrar - Primeiro a Sair), Relógio, LFU (*Least-Frequently-Used* - Menos Frequentemente Utilizado), LRU (*Least-Recently-Used* - Menos Recentemente Utilizado), NRU (*Not-Recently-Used* - Não

Recentemente Utilizado) 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 colunas da tabela da parte inferior.

A parte superior direita da Figura 2 (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”(marcada em 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.

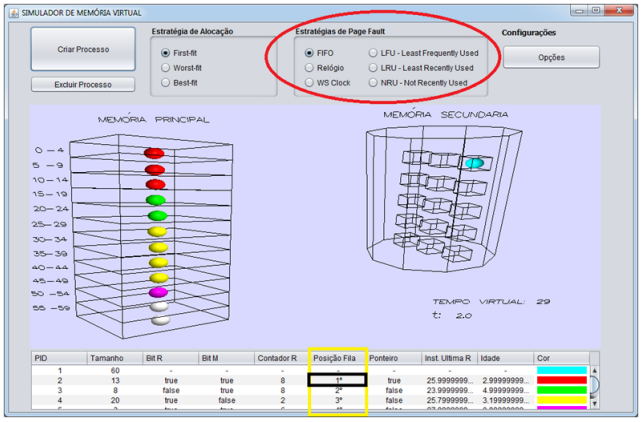

Figura 2. Apresentação *Page-Fault* utilizando a estratégia FIFO

A Figura 3 mostra o momento posterior a criação de um novo processo na cena ilustrada na Figura 2. 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 secundária após sair da memória principal deixando uma lacuna para que o novo processo possa ser alocado.

O processo representado pela esfera vermelha (id 2) foi escolhido como vítima, pois no momento antes da criação do processo (Figura 2), a célula da tabela circulada de preto 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.

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,

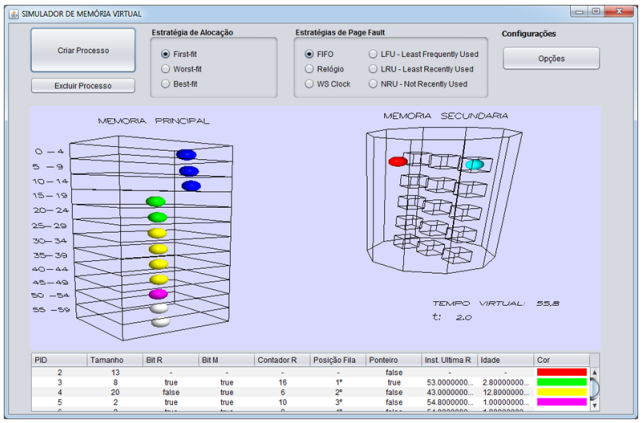

Figura 3. Apresentação *Page-Fault* utilizando a estratégia FIFO - Momento em que ocorre uma falta de página.

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 a escolha 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 existe 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 4 ilustra 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.

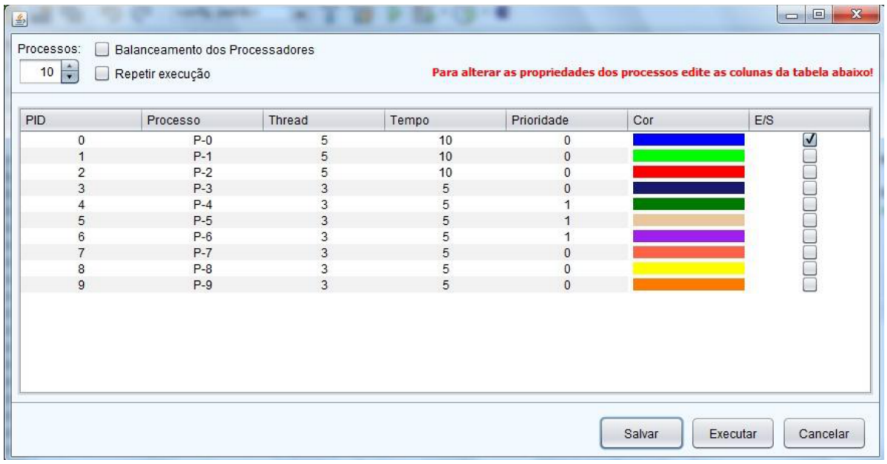

Figura 4. Tela para Criação e Visualização das informações dos Processos.

A Figura 5 apresenta a simulação do Algoritmo de Escalonamento Tempo Compartilhado baseada nas características que foram definidas de acordo com a Figura 4. Ao centro da Figura 5 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.

No lado direito da Figura 5 são exibidos três quadros que representam os processadores, cada quadro possui quatro cubos os quais representam os núcleos que cada processador 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.

A Figura 6 apresenta a simulação do Algoritmo de Escalonamento em Bando, no qual os *threads* de um processo P qualquer só serão escalonados se todos couberem nos núcleos dos processadores, nota-se que os processos P-0 (azul) e P-1 (verde) com

Figura 5. Simulação do Algoritmo Tempo Compartilhado.

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.

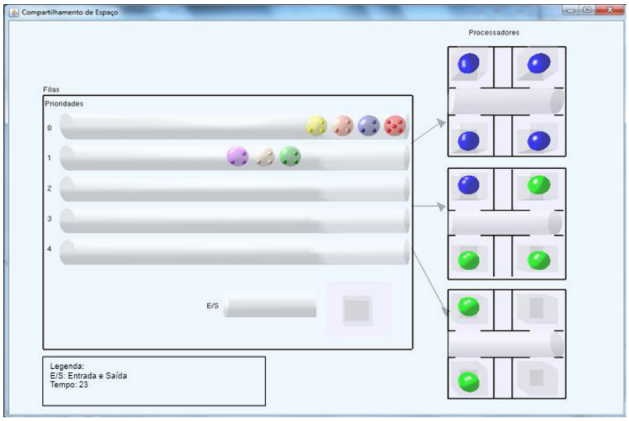

Figura 6. Simulação do Algoritmo Escalonamento em Bando.

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, como mostra a Figura 7. Após a conclusão todo o processo com os *threads* relacionados a ele volta para o fim da fila de espera.

4. Experimentos e Resultados

Com o intuito de verificar e validar os simuladores desenvolvidos, estes foram apresentados à alunos que já haviam cursado a disciplina de Sistemas Operacionais e, foi aplicado um questionário para avaliação dos simuladores. A escolha pelos alunos já terem cursado se deve ao fato da temática de sistemas com multiprocessadores ou com processadores *multicore* ser trabalhada apenas no final da disciplina.

O questionário foi composto de 7 questões e tinham como opções de respostas: ”Discordo Totalmente”, ”Discordo em Parte”, ”Indiferente”, ”Concordo em Parte”e

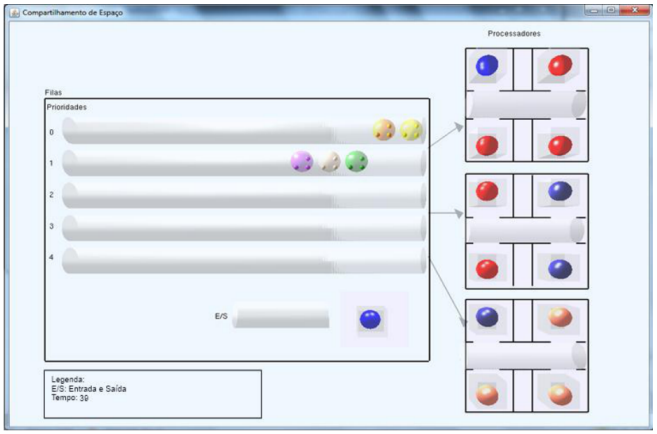

Figura 7. Simulação de Entrada e Saída, Algoritmo Escalonamento em Bando.

"Concordo Totalmente". As questões aplicadas foram:

1. O simulador é fácil de utilizar;
2. A interface do simulador é amigável;
3. O simulador ajuda a despertar o interesse pelo assunto;
4. Utilizar o simulador auxilia na compreensão/assimilação dos conceitos;
5. O simulador facilita entender as políticas abordadas por ele;
6. O simulador aborda todos os conceitos do assunto;
7. Deve-se utilizar o simulador como ferramenta de ensino.

O questionário foi aplicado à 76 alunos de 3 instituições de ensino diferentes no período de 2013/2 à 2014/1 e, o resultado é apresentado na Tabela 2. Em resumo, a maioria dos alunos concordou que o simulador é fácil de utilizar, sua interface é amigável e que deve ser utilizado como ferramenta de ensino. Todos concordaram que o uso do simulador auxilia na compreensão/assimilação dos conceitos e que facilita entender as políticas abordadas por ele.

Tabela 2. Resultado do questionário

Questões	Discordo Totalmente	Discordo em Parte	Indiferente	Concordo em Parte	Concordo Totalmente
1		3,15%	17,63%	69,19%	10,03%
2		6,52%	19,06%	57,31%	17,11%
3				82,58%	17,42%
4			15,92%	74,67%	9,41%
5				83,27%	16,73%
6	2,23%	21,45%		57,84%	18,48%
7			13,62%	44,25%	42,13%

5. Conclusões

As implementações dos dois simuladores preenchem lacunas que os outros simuladores analisados neste trabalho não abordavam: o *Page Fault* e os múltiplos processadores e/ou *multicores*.

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 tornou as telas 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 conteúdo e, consequentemente, aumentando a eficiência do processo de ensino e aprendizagem entre professores e alunos, conforme demonstrou a pesquisa realizada.

Como trabalhos futuros, os simuladores serão integrados em um só simulador e serão implementados de novos algoritmos. Além disso, serão aplicados questionários aos alunos que estão cursando a disciplina, para avaliarem os softwares.

Referências

Carvalho, D. S., Balthazar, G. R., 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.

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.

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

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 (2014). The freebsd project. `http://www.freebsd.org/`.

Torvalds, L. (2014). The linux kernel. `http://www.kernel.org`.

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