Análise do simulador SOsim

Anna G. M. Oliveira[1], Naomi C. Ribes[2]

[1]Centro de Desenvolvimento Tecnológico – Universidade Federal de Pelotas (UFPEL)
Praça Domingos Rodrigues – Centro – 96010-450 - Pelotas – RS – Brasil

agmoliveira@inf.ufpel.edu.br, ncribes@inf.ufpel.edu.br,

Resumo. *Este trabalho apresenta uma ferramenta para auxiliar no ensino da disciplina de sistemas operacionais. A ferramenta foi desenvolvida para simular o funcionamento de um sistema operacional, explorando os conceitos de gerenciamento de disco, memória e processos de forma integrada e interativa.*

1. Visão Geral do Simulador

O SOSim[1] desenvolvido pelo prof. Luiz Paulo Maia é um simulador de Sistemas Operacionais que oferece uma experiência prática para estudantes e professores de Ciência da Computação. A interface gráfica desta ferramenta é simples e fácil de usar, permitindo que os alunos experimentem e compreendam conceitos fundamentais de Sistemas Operacionais de forma prática e interativa, funcionando como apoio ao ensino e aprendizado dos mesmos.

As janelas disponíveis neste SOSim incluem a janela de Gerência de Processos, que permite visualizar e controlar os estados e comportamentos dos processos em execução; a janela de Gerência de Memória, que permite simular o gerenciamento da memória do sistema, visualizando os blocos de memória e como os processos estarão alocados no mesmo; e a janela de Gerência do Processador, que apresenta informações sobre o escalonamento da CPU e o uso de recursos do sistema. Além disso, temos a Janela de Console SOsim que mantém o registro dos processos, assim como, opções de ajuda e configurações.

1.1. Janela Console SOsim

Na janela de Console do SOsim temos os ícones de customização do sistema, nela o usuário pode determinar quais janelas serão apresentadas, ver as estatísticas e parâmetros do sistema, além das opções de Ajuda e Sair. Além disso, são apresentadas as informações sobre o tempo decorrido desde a inicialização do simulador em um temporizador em segundos, assim como o monitoramento do número de processos e a utilização percentual da memória.

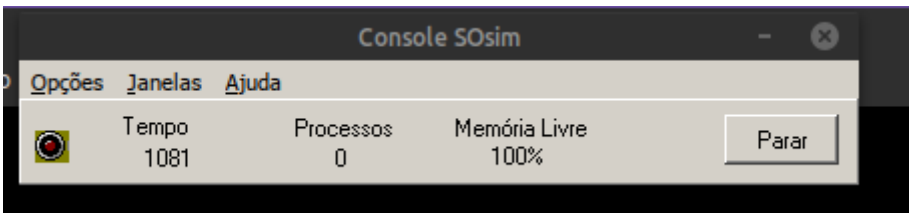

Figura 1: Janela de Console SOsim.

1.2. Janela de Gerência de processos

Na janela de gerência de processos do SOsim é possível visualizar uma lista de processos ativos no sistema, incluindo informações como o identificador de processo (PID), prioridade, estado do processo, tempo de execução e quantidade de frames. Os usuários podem realizar diversas operações nessa janela, incluindo a criação, término, suspensão e finalização de processos, assim como a atribuição de uma cor para os mesmos.

Para o tempo de execução, temos a mensuração do tempo de UCP (Unidade Central de Processamento) em sua janela de gerenciamento de processos. Essa funcionalidade permite que os usuários visualizem e monitorem a quantidade de tempo de processamento que cada processo consome em um determinado intervalo de tempo.

Além disso, a janela de gerenciamento de processos também permite que os usuários alterem as prioridades dos processos em execução, para experimentar como diferentes políticas de escalonamento de processos afetam o desempenho do sistema. Por último, temos a coluna de cor para os processos, onde o usuário pode escolher uma cor para a representação e diferenciação destes.

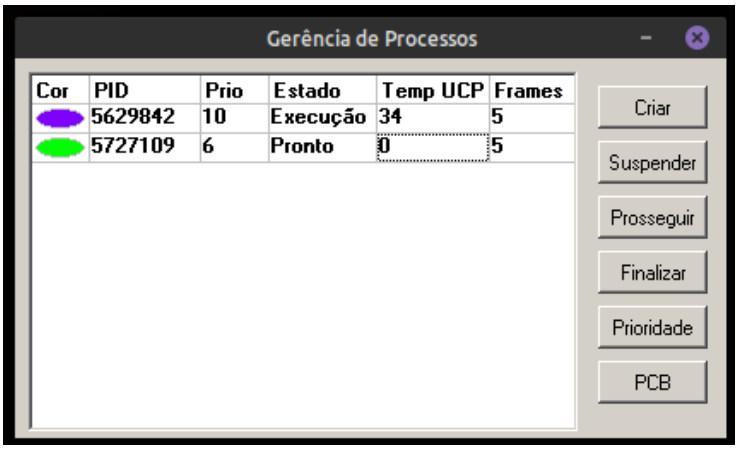

Figura 2: Janela de gerência de processos SOsim.

1.3. Janela de Gerência do Processador.

A janela de gerência do processador simula o gerenciamento da CPU (Unidade Central de Processamento) em um Sistema Operacional. Ela apresenta informações sobre o uso da CPU, como o processo que está em execução e a lista de processos prontos. Nela, existe um ícone de espera para processos de I/O, ou seja, processos que utilizam do sistema de entrada e saída. Para poder simular essa operação é mostrado um contador de sete segundos que simula o tempo de resposta do dispositivo utilizado. Também temos uma linha para a visualização dos processos suspensos.

Além disso, existem três botões de escala móveis para que o usuário possa ajustar e determinar este tempo de espera do I/O, assim como a fatia de tempo do processo e o clock da UCP.

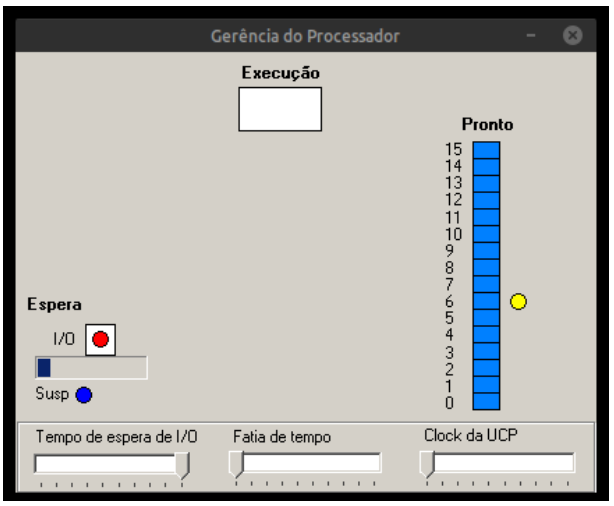

Figura 1.3: Janela de Gerência do Processador.

1.3. Janela de Gerência Memória.

Como última janela da configuração padrão temos a janela de gerência de memória, onde pode-se visualizar 100 blocos de memória numerados de 0 a 99. Quando é criado um processo na janela de gerência de processos é atribuída uma quantidade máxima de frames. Estes são imediatamente alocados na memória de acordo com a capacidade representada por Tam. LPL. Além disso, o Tam. LPM indica o tamanho da página menos utilizada, que é um parâmetro aplicado em situações onde é necessária a substituição na memória física quando não há espaço disponível para alocar uma nova página.

Quando um processo é finalizado, sua área de memória é finalizada sem que haja realocação dos outros processos. O tamanho máximo de frames permitido por processo são cinco.

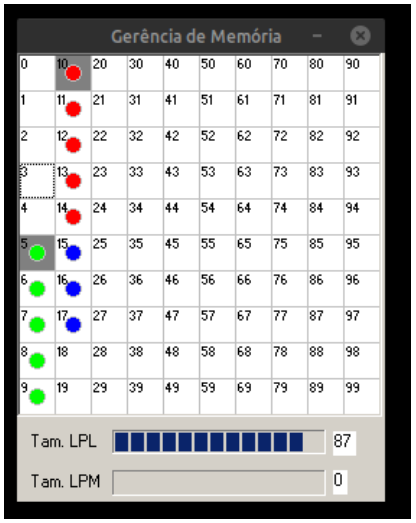

Figura 1.4: Janela da gerência de Memória

2. Sobre processos

Para nossa execução, utilizaremos três processos. O azul, CPU-bound com 5 de prioridade e 5 frames, o vermelho IO-bound (disco) de 6 de prioridade e 4 frames, e o

amarelo, IO-bound (terminal) de 5 de prioridade e 3 frames. Os PIDs podem variar uma vez que foram feitas várias execuções para abranger todo o aspecto do trabalho.

Os possíveis estados que o simulador dá para o processo de I/O são: pronto e I/O. Sendo que o processo fica com o estado de pronto quando está na lista de prontos da gerência do processador, e fica com estado de I/O quando está simulando a espera pelo dispositivo de I/O. Além disso, o usuário pode atribuir manualmente o estado de suspenso além da finalização do processo. Outro estado que pode ser ocasionalmente atingido é o de Pfault, que configura a falta de página, e ocorre quando um processo precisa acessar uma página de memória virtual que não está presente na memória física.

A aplicação da fatia de tempo no simulador que é apresentada na janela de Gerência de Processador, diz respeito ao tempo que um processo tem para executar na CPU antes de ser interrompido e dar espaço para outro processo. No experimento executado, nota-se que, com a disposição de processos escolhida e com a opção na medida padrão na escala de fatia de tempo, houve muitas trocas de contexto, de forma que os processos de I/O passavam 1 segundo em execução e prontamente eram realocados para o timer de espera, enquanto que o processo azul de CPU-bound obteve um tempo de execução sete vezes maior, considerando o tempo de espera de I/O padrão do simulador.

O clock da UCP (Unidade Central de Processamento) ou clock do processador determina a velocidade de operação do processador e representa o número de ciclos de clock que a UCP pode executar por segundo. Com uma maior frequência de clock no simulador, temos uma execução das tarefas mais rápida. Em um ambiente real isso pode levar em a maior custo operacional, especificamente em energia e calor a ser gerado pelos componentes. Como não necessitamos de uma unidade de resfriamento ou de pagar a conta de luz, podemos elevar o nível da escala para obtermos um desempenho melhor.

O simulador, por padrão, vem com a política de escalonamento circular, ou Round Robin, e como visto em aula, temos o processador alocando um intervalo de tempo fixo chamado de quantum, que pode ser mudado na escala pelo usuário. Neste tipo de escalonamento é objetivado que cada um dos processos obtenha uma fatia de tempo justa, evitando starving (negligenciamento do processo na divisão de recursos).

Segundo testes feitos no simulador, na ocasião em que dois processos de CPU-bound estão na fila de prontos, mesmo utilizando o escalonamento circular puro, há starving se as prioridades são diferentes. Entretanto, em caso de prioridades iguais, o round robin implementa corretamente a divisão de tempo entre os processos.

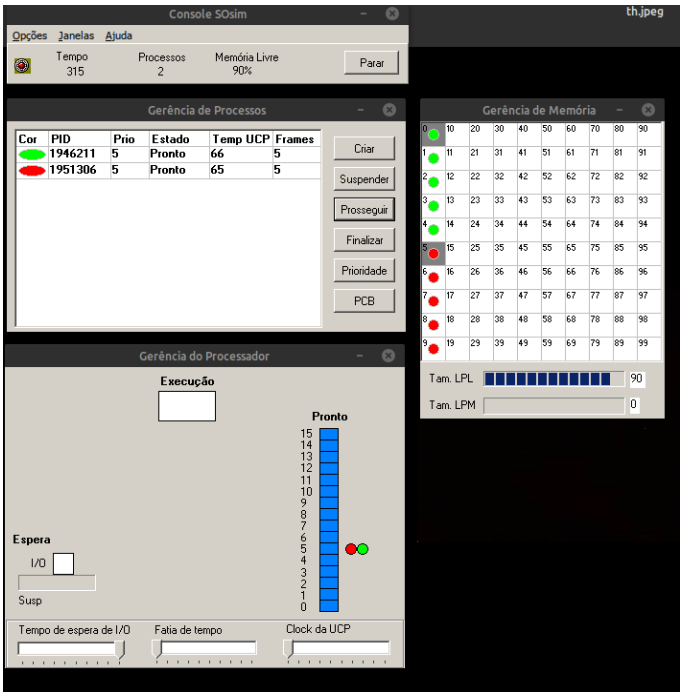

Figura 2: Exemplo de execução de processos de CPU-bound de prioridades iguais.

Em um exemplo alternativo, com dois processos I/O-bound de mesma prioridade, as fatias de tempo também são justas. Em prioridades diferentes teremos o mesmo resultado, por causa da fila de espera de processos I/O-bound.

Neste experimento, temos os processos verde e vermelho como I/O-bound de prioridade 3, amarelo e roxo como misto de prioridade 2, e azul claro e azul escuro como CPU-bound. Como resultados, obtivemos que os processos de maior prioridade foram os que receberam menor tempo de UCP, por mais que eles tenham sido criados antes e todos os processos tenham a mesma quantidade de frames. Em segundo lugar em tempo de processamento temos os processos mistos, e em terceiro lugar, os CPU-bound.

Isso se deve ao fato de que os processos de CPU-bound raramente necessitaram usar a I/O. Obtendo maior tempo de execução já que a CPU estava vazia uma vez que todos os outros processos estavam sempre/quase sempre na lista de espera por I/O. Em uma situação com as prioridades invertidas, existe o starving de todos os processos que não são de CPU-bound. Uma vez com as prioridades, os processos de I/O e mistos nunca chegam a sair da lista de pronto, uma vez adicionados os processos de CPU-bound.

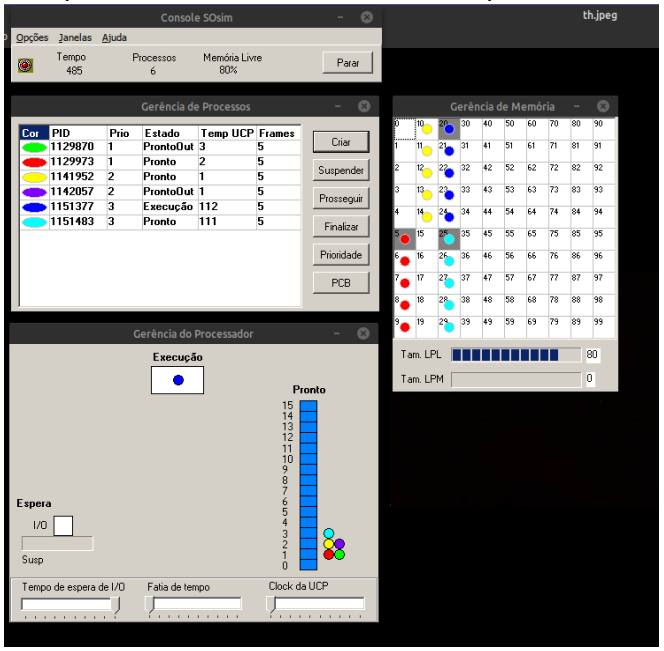

Figura 2.2: Experimento com prioridades invertidas.

3. Gerência de memória

Neste experimento utilizamos os mesmos processos do exercício anterior, com prioridades 3, 2, 1 respectivamente. Na iteração de paginação por demanda, o simulador rodou por 1000 eventos. Obtendo 24 page faults e com uma utilização de memória de 10%.

Já na paginação antecipada, não obtivemos absolutamente nenhum erro de page fault com os mesmos tipos e prioridades de processo na iteração de 1000 eventos. Entretanto, utilizamos 20% da memória, o dobro utilizado na paginação por demanda.

Isso ocorre porque a paginação por demanda por definição, só carrega as páginas necessárias quando as mesmas são solicitadas, o que pode levar a falta de páginas sendo a situação em que a página requisitada não está na memória e precisou ser buscada no disco rígido. Por outro lado, na paginação antecipada, todas as páginas necessárias e

adjacentemente prováveis para a execução de cada processo do programa foram carregadas na memória antecipadamente, mesmo que algumas delas não tenham sido necessárias, o que explica o uso maior de memória.

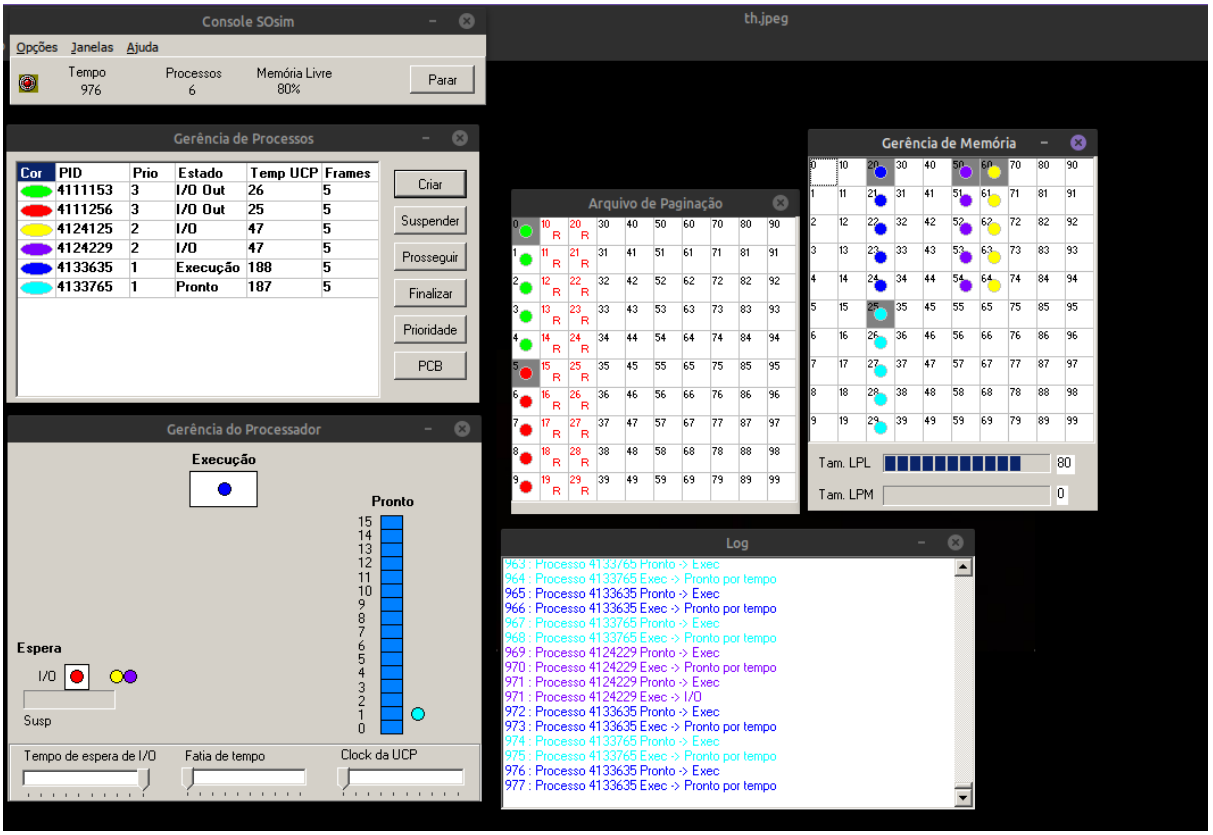

Figura 3: Simulador rodando com a política de paginação antecipada, paginação e log.

Quando clicamos em um processo na janela de processos podemos habilitar a tecla de PCB, onde podemos visualizar a tabela de páginas na janela de contexto do processo. Nela, podemos ver que, para o experimento mencionado, teremos 5 frames para cada processo, como habilitado pelo usuário. O NPV representa o número da página virtual, o NPR o número da página real, além do bit V que é o bit de validade, o bit M que é o bit de modificação, e o local onde está na memória, que pode estar na memória principal e na página. Cada tabela de páginas também possui o ID do processo, e, por isso, podemos ver como o sistema está gerenciando a memória virtual do processo e identificar possíveis problemas, principalmente com Page Faults. Sendo possível a adequação de políticas para melhorar seu desempenho.

Figura 3.1: Tabela de páginas do primeiro processo criado de I/O de prioridade 3.

4. Conclusão

Em conclusão, o simulador SOsim é uma ferramenta extremamente útil para ensinar alunos sobre sistemas operacionais. Ele possui uma interface amigável e recursos interativos, e é capaz de proporcionar aos alunos uma experiência prática e visual na aprendizagem dos conceitos fundamentais de sistemas operacionais.

O simulador foi particularmente prestativo no entendimento do funcionamento das prioridades no sistema, e o quanto o tipo do processo impacta no desempenho do modelo, por muitas vezes sendo mais crucial do que o nível de prioridade em si. Um aspecto satisfatório foi ver a disposição de memória em blocos e sua organização com o passar dos eventos, fato que remeteu a explicação em aula e a imagem ensinada dos blocos em disco, que apresentam organizações de forma muito parecida. Além disso, ver na prática as políticas de escalonamento para números grandes de processos rodados por consideráveis períodos de tempo, proporcionou uma visão mais ampla dos exercícios de escalonamento vistos em aula. De forma a mostrar claramente a importância de uma política de escalonamento adequada para seu conjunto de processos.

Entretanto, nos aspectos mais finos da interface visual com relação a bugs há oportunidade de melhoria no simulador.

Em Linux, quando a janela de console SOsim é movida, ela retorna imediatamente ao ponto de origem; na janela de gerência de memória se ela for movida e houver um processo de I/O a ser executado, após o tempo de espera de I/O, a janela volta para seu lugar de origem também, impossibilitando ao usuário a reorganização das janelas. Além disso, todas as janelas sobrepõe todo e qualquer programa do sistema operacional do computador, de forma que, por exemplo, se o simulador estiver rodando e for aberto o chrome, o simulador sobrepõe o Chrome aberto; se em seguida for aberto o terminal, o simulador sobrepõe o terminal.

Se o simulador for minimizado não há como voltar para ele, uma vez que a janela de SOsim que aparece não permite click. A janela de log não permite ao usuário visualizar logs passados enquanto o simulador roda. O simulador cria um .txt para todas as runs que apenas consiste em “ 0 : Inicializando SOsim … 5 : SOsim pronto”, e se qualquer um desses logs antigos for apagado o simulador dá erros de inicialização constantes e persistentes criados mais rapidamente do que o usuário pode fechar para tentar resolver, resultando na obrigação de um reboot forçado na máquina. Sendo que, em nenhum momento é apagado o .ini, apenas os logs antigos.

Além disso, não tem uma tradução das siglas usadas (NPV, NPR, tam.LPM …), fazendo com que o usuário já tenha que ter conhecimento em SO para usar o simulador. No geral, o simulador cumpre seu papel didático.

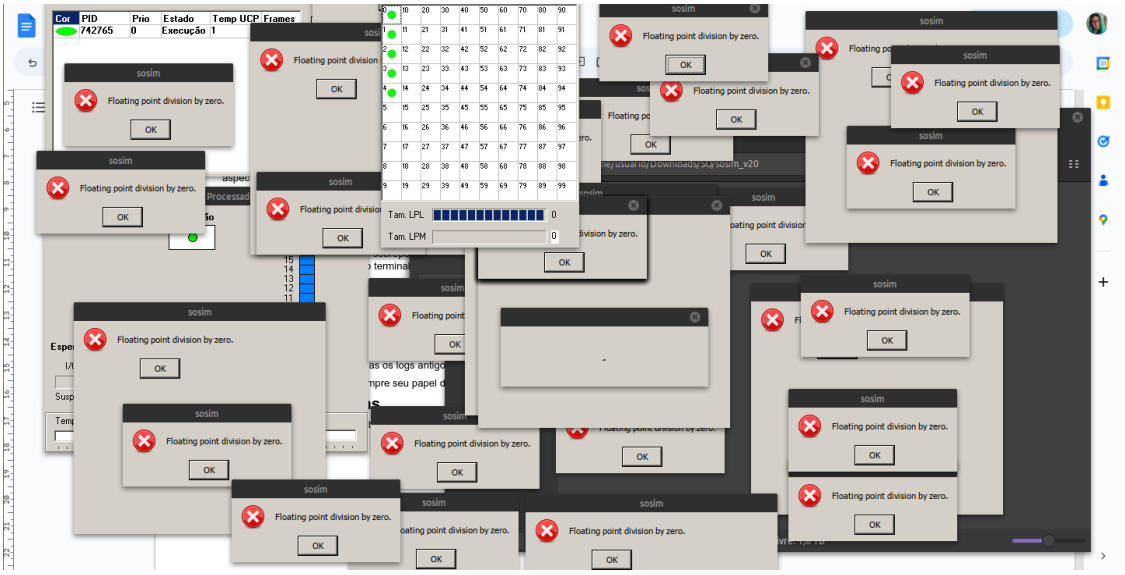

Figura 4: erros de log.txt.

Um outro simulador de recursos de sistemas operacionais (SimulaRSO)[2] que é um simulador web desenvolvido por estudantes de graduação da Universidade Católica de Santos, ele faz a simulação de escalonamento de processos, de disco e de paginação de memória. Além disso, os algoritmos podem ser alterados ou comparados entre si. A interface também é um recurso a ser destacado pelo software, que é de fácil utilização e permite a representação gráfica das simulações realizadas.

Referências

Luiz P. M. (2001) “SOsim: Simulador para o Ensino de Sistemas Operacionais Versão 2.0” http://www.training.com.br/sosim/

Caio R, (2011) “Projeto SimulaRSO - Simulador de Recursos de Sistemas Operacionais”, https://github.com/caio-ribeiro-pereira/SimulaRSO