Uma Ferramenta de Simulação do Processo de Substituição de Páginas em Gerência de Memória Virtual

Fagner do Nascimento Fonseca, Flávia Maristela S. Nascimento

[1]Departamento de Computação
Instituto Federal da Bahia (IFBA)
41.301-015 – Salvador – BA – Brasil

fagneolli@gmail.com, flaviamsn@ifba.edu.br

Abstract. *This paper presents the development of an application as a support tool for disciplines that address operating systems. The software simulates the process of page replacement in virtual memory management. The application serves not only to demonstrate the functioning of page replacement algorithms, but also allows students to implement their own algorithm and monitor their functioning.*

Resumo. *Este trabalho apresenta o desenvolvimento de uma aplicação como ferramenta de apoio para disciplinas que abordam o tema Sistemas Operacionais. O software simula o processo de substituição de páginas em gerência de memória virtual. A aplicação serve não só para demonstrar o funcionamento dos algoritmos de substituição de páginas, mas também permite que o aluno implemente o seu próprio algoritmo e possa acompanhar o seu funcionamento.*

1. Introdução

A memória é de fundamental importância para o funcionamento de um sistema de computação, pois é nela que são armazenados os dados necessários para o funcionamento dos programas, inclusive do sistema operacional. O ideal seria que o sistema tivesse a sua disposição uma grande quantidade de memória, que fosse barata e não-volátil, ou seja, que os dados não fossem perdidos quando o computador fosse desligado. [Sílberschatz et al. 2004, Tanenbaum 2010].

A tecnologia que dispomos hoje ainda não nos permite essa realidade, portanto, os sistemas operacionais obedecem uma hierarquia de memória, que é dividida nas seguintes categorias: registradores, cache, memória principal, armazenamento secundário e armazenamento permanente. Os registradores representam a memória de menor capacidade e de maior velocidade, ao passo que os dispositivos de armazenamento permanente, como discos as fitas magnéticas são, em geral, mais baratos e mais lentos. [Stuart 2011, Tanenbaum 2010].

Com o passar do tempo, o computador foi exigindo cada vez mais espaço da memória. Isso ocorreu não só pelo aumento na complexidade e no tamanho dos programas, mas também devido ao avanço das técnicas utilizadas para gerenciamento de memória. A memória principal ainda era um recurso muito caro para a época e embora seu preço estivesse diminuindo e o seu tamanho aumentando, a quantidade de programas que demandavam grandes quantidades de memória aumentava muito mais, agravando ainda mais

o problema [Stuart 2011, Deitel et al. 2005, Flynn and McHoes 2002, Tanenbaum 2010, Stallings 2003].

Para tentar resolver esta questão de memória escassa, foi criado o conceito de memória virtual, com a qual é possível criar a ilusão de que o sistema operacional possui uma quantidade maior de memória disponível do que a que realmente possui. Assim, programas muito grandes que antes eram impossibilitados de executar, por causa do tamanho da memória disponível, agora podem executar [Deitel et al. 2005].

Entender como tudo isso funciona, no entanto, não é uma tarefa das mais simples. Em geral, nos cursos da área de Computação, os alunos de disciplinas como Arquitetura de Computadores e/ou Sistemas Operacionais apresentam dificuldade para entender a interação entre hardware e software que envolve o processo de gerência de memória, mais especificamente os algoritmos responsáveis pela administração e funcionamento da memória virtual. Um dos fatores que pode levar a isto é o alto grau de abstração requerido por esta temática, bem como o elevado nível de conhecimento acerca do funcionamento de cada um dos elementos envolvidos. Por estas razões, muitas vezes pode ser difícil para o aluno conseguir visualizar de maneira clara como todo esse processo funciona. Experiências observadas entre professores e alunos mostra como é difícil a compreensão dos assuntos dessa disciplina, bem como a aplicação prática dos mesmos [Machado and Maia 2001].

A forma tradicional que a disciplina é ensinada, em geral utilizando apenas teoria, dificulta o entendimento e em alguns casos pode até desmotivar o aluno, que pela falta de ferramentas capazes de mostrar na realidade os conceitos vistos na teoria, fica bastante longe do objeto de estudo. Um curso de sistemas operacionais com apenas aulas teóricas não possibilita que os alunos compreendam e assimilem os conceitos abordados. Pode-se observar que muitos alunos concluem a disciplina conhecendo diversos conceitos e algoritmos, mas sem saber como integrar tudo o que foi aprendido [Machado and Maia 2001, Maziero 2002].

O uso de ferramentas práticas nesta disciplina é de fudamental importância, uma vez que busca atingir alguns objetivos práticos, tais como (a) consolidar a compreensão dos mecanismos e estruturas básicas do funcionamento de um núcleo de sistema operacional típico; (b) compreender claramente como as diversas partes constituintes de um sistema operacional interagem e se integram e (c) desenvolver a capacidade de propor e testar suas próprias soluções[Maziero 2002].

O presente trabalho apresenta uma ferramenta que simula o processo de gerência de memória virtual, especificamente os algoritmos de substituição de páginas, demonstrando graficamente como todo o mecanismo funciona. O objetivo do trabalho é servir como ferramenta de apoio para disciplinas que abordam essa temática.

O artigo segue a seguinte estrutura. A Seção 2 descreve a motivação da gerência de memória, explica sua hierarquia e organização. A Seção 3 explica como funciona a memória virtual e quais os seus desafios. A Seção 4 apresenta o estado da arte e cita alguns simuladores tanto de gerência de memória virtual quanto de gerência de memória simples. A Seção 5 apresenta a ferramenta desenvolvida. A Seção 7 mostra os possíveis projetos que podem ser iniciados tendo como base a ferramenta proposta.

2. Gerência de Memória

A memória não possui divisões uniformes onde cada processo pode ser armazenado, ela nada mais é do que um vetor de bytes. Portanto, é preciso saber como alocar a memória disponível para os processos que precisam executar. O método de gerência de memória que será empregado em um determinado sistema depende também do hardware que está disponível [Stuart 2011, de Oliveira et al. 2010, Sílberschatz et al. 2004].

Nos primeiros sistemas, somente um processo poderia ser executado por vez e ele permanecia na memória até terminar a sua execução. Uma parte da memória era dedicada ao sistema operacional e o restante era todo disponível para o processo que estivesse em execução. O programador era totalmente responsável por gerenciar a alocação da memória utilizada pelos processos. Se um processo fosse grande demais para ser carregado, a memória principal deveria ser aumentada ou então o programa deveria ser modificado de maneira que ele coubesse na memória[Deitel et al. 2005, Flynn and McHoes 2002, de Oliveira et al. 2010, Tanenbaum 2010].

Para tentar resolver o problema de somente um processo poder ser executado por vez, algumas estratégias foram criadas como por exemplo, a abordagem por partição fixa, que consistia em dividir a memória de maneira lógica em várias partições, que possuíam tamanhos iguais ou variados. Neste contexto, alguns problemas precisaram ser tratados como por exemplo proteger o espaço de memória de um processo, fragmentação interna e ainda a alocação de processos em partições dinâmicas [Deitel et al. 2005, Flynn and McHoes 2002, de Oliveira et al. 2010].

Independente se o esquema é de partições fixas ou dinâmicas, o sistema operacional precisa manter uma lista que guarda quais partições estão desocupadas. Quando um novo processo precisar ser carregado, o gerenciador precisa colocá-lo em alguma partição livre. Existem alguns algoritmos para percorrer a lista em busca de uma partição que possua tamanho suficiente para alocar o processo [Flynn and McHoes 2002].

A memória sozinha não tem capacidade para suportar todos os programas. Se um processo precisar executar e todas as partições da memória estiverem ocupadas, ele só poderá executar depois que algum processo liberar a partição e caso a mesma tenha tamanho suficiente para comportar novo processo. No sentido de solucionar este problema foi criada a técnica de *swapping*, que é responsável pela transferência temporária do processo para o disco e para a memória [Stuart 2011, Deitel et al. 2005, de Oliveira et al. 2010, Sílberschatz et al. 2004, Tanenbaum 2010], premissa fundamental para a implementação de Memória Virtual, explicada na próxima seção.

3. Memória Virtual

Apesar das diversas técnicas citadas anteriormente otimizarem o uso da memória principal, alguns problemas ainda permanecem. Um processo precisa estar em um espaço contíguo para poder executar e precisa estar em sua totalidade carregado na memória. Outro problema é o número de processos executando em paralelo, que dependerá do tamanho da memória e do tamanho dos processos. Os processos que podem ser executados possuem um limite de tamanho, que é o tamanho da memória. Um processo que exceda esse tamanho nunca poderá ser executado. Uma primeira solução para esses problemas seria aumentar o tamanho da memória. Mas como já foi dito, o tamanho dos processos

vem aumentando numa taxa muito maior que a memória física. Exemplos de softwares que demandam muito espaço: simulação da criação do universo, simulação de uma aeronave. [de Oliveira et al. 2010, Tanenbaum 2010].

A técnica de memória virtual surge para solucionar esses problemas. Sistemas baseados em memória virtual dão aos processos a ilusão de que o computador possui mais memória do que a que ele de fato possui. Pois o gerenciador disponibiliza como espaço de endereçamento tanto a memória principal quanto a memória secundária (disco). Com isso, cria-se a ilusão de que o processo está o tempo inteiro carregado na memória principal [Deitel et al. 2005, Flynn and McHoes 2002].

3.1. Paginação

A paginação permite que o processo seja alocado em um espaço não-contíguo. A memória não precisa mais possuir partições de diversos tamanhos, pois as páginas são de tamanho fixo. O bloco da memória principal que guarda a página é denominado moldura de página e possui o tamanho exato da página. Quando um processo está pronto para carregar, as páginas que são necessárias são carregadas em qualquer espaço de memória que esteja disponível. Nesse tipo de esquema não existe fragmentação externa, mas pode ocorrer fragmentação interna, nesse caso a fragmentação será reduzida [Sílberschatz et al. 2004].

É preciso saber de alguma forma se determinada página está carregada na memória principal ou se está no disco. Uma solução empregada é o uso do bit válido-inválido. O gerenciador mantém uma tabela de páginas que mapeia todas as páginas, bem como se ela está carregada na memória principal ou não.

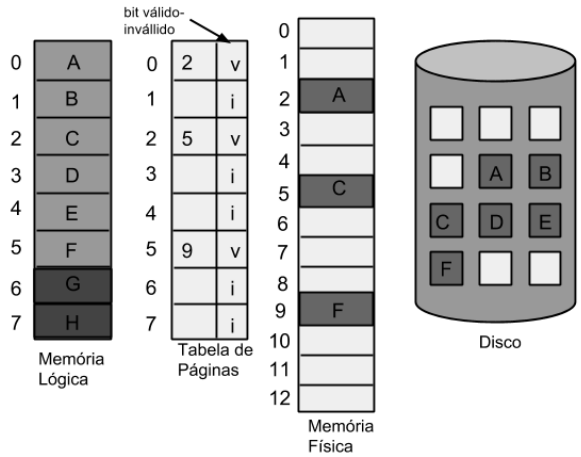

Figura 1. Tabela de página quando algumas páginas não estão na memória principal [Sílberschatz et al. 2004]

A Figura 1) apresenta a memória física com seus endereços (variando de 0 à 12) e a memória virtual, modelada no disco e a Tabela de Páginas, que apresenta a correlação dos endereços e a coluna que apresenta o bit de válido / inválido (“i” e “v”, respectivamente) [Sílberschatz et al. 2004].

Existem dois tipos de estratégias para paginação: a paginação por demanda e a paginação antecipada. Na paginação por demanda, é utilizada a busca por demanda onde o gerenciador espera que o processo referencie a página para então carregá-la na

memória principal. Na paginação antecipada, o gerenciador tenta prevê quais páginas o processo precisará e tenta carregá-las na memória antecipadamente [Deitel et al. 2005, de Oliveira et al. 2010].

Comparando paginação por demanda com a paginação antecipada, podemos observar que na paginação por demanda, apenas as páginas que de fato serão utilizadas pelos processos serão trazidas para a memória principal. Essas páginas serão carregadas uma por vez, conforme os processos as necessitam, esse fator aumenta o grau de multiprogramação, já que mais processos poderão ser carregados na memória principal. O problema é que o tempo que o processo espera por páginas que não estão carregadas pode ser muito caro [Deitel et al. 2005].

É preciso de uma boa heurística para carregar de uma só vez na memória um certo número de páginas que o processo possivelmente irá referenciar. O tempo em que um processo fica ocioso na memória de fato diminui, pois o número de E/S para o disco diminui. Entretanto, se for utilizada uma heurística ruim, o sistema pode apresentar pior desempenho do que um sistema com paginação por demanda, pois as páginas que serão carregadas antecipadamente poderão não serem as próximas páginas a serem referenciadas pelo processo. O número de páginas que serão carregadas de uma só vez também é um fator muito importante, pois quanto mais páginas forem carregadas, menos processos poderão ser executados em paralelo [Deitel et al. 2005].

Como várias páginas são carregadas de uma só vez, existe o tempo maior de espera para que essas páginas sejam carregadas do armazenamento secundário. Normalmente as técnicas de paginação antecipada carregam páginas que são contíguas no espaço de endereçamento virtual. O fato dessas páginas serem contíguas no espaço de endereçamento virtual, não significa que elas são contíguas no disco. Portanto, haverá uma sobrecarga no carregamento dessas páginas [Deitel et al. 2005].

O tempo que o processo espera na paginação antecipada e na paginação por demanda são quase os mesmos. Porém a paginação antecipada pode apresentar resultados melhores que a paginação por demanda, tudo vai depender dos critérios citados anteriormente para um bom desempenho desse tipo de paginação. Normalmente a estratégia de paginação antecipada é combinada com a paginação por demanda [Deitel et al. 2005].

3.2. Substituição de páginas

Quando um processo referencia uma página, pode ser que todas as molduras de páginas na memória principal já estejam ocupadas. A memória principal não é utilizada apenas para carregar dados e instruções de programas, outro uso dela, por exemplo, são os *buffers* de operações de entrada/saída, que por sinal ocupam uma considerável quantidade de memória. O evento em que um processo referencia uma página que não está carregada na memória é chamado de *page-fault* ou falta de página [Sílberschatz et al. 2004].

Quando isso ocorre, o gerenciador de memória deverá transferir alguma página carregada na memória para o disco e então carregar a página referenciada. Esse processo é chamado de substituição de páginas. O algoritmo a seguir descreve como esse processo funciona [Sílberschatz et al. 2004, Flynn and McHoes 2002]:

1. Localizar a posição da página desejada no disco.
2. Encontrar um quadro livre.

   (a) Se houver um quadro livre, use-o.
   (b) Se não houver quadros livres, use um algoritmo para selecionar um quadro.
   (c) Gravar a página vítima no disco; alterar as tabelas de página e quadro .
3. Ler a página vítima no disco; alterar as tabelas de páginas e quadro de acordo.
4. Retornar o processo de usuário.

Para realizar a substituição de páginas são necessárias duas transferências e isso aumenta o tempo de resposta do programa. Uma solução para esse problema é utilizar um bit de modificação. Se a página sofrer algum tipo de alteração esse bit será ativado. No momento da substituição de uma página, será verificado através do bit de modificação se a página sofreu alguma alteração. Caso ela não tenha tido alterações, ela poderá ser removida da memória, pois ela já está armazenada no disco. Com esse artifício conseguimos reduzir a sobrecarga dessa operação, visto que a página só será copiada da memória para o disco caso ela tenha sido modificada [Sílberschatz et al. 2004].

Para substituir uma página o sistema primeiro deve decidir qual página será substituída. Existem diversas estratégias de substituição. Para analisar o desempenho dessas estratégias elas são comparadas com a estratégia de substituição ótima. Essa estratégia diz que a página ideal a ser substituída é aquela que não será referenciada no futuro mais distante possível. A estratégia ótima serve apenas como parâmetro para analisar as outras estratégias. É impossível implementar o algoritmo ótimo, pois não tem como se prever o comportamento dos processos, ou seja, não tem como prever qual será a ordem exata em que ele referenciará as páginas e nem quais processos serão executados no futuro. As estratégias de substituição de página devem balancear a redução de falta de páginas com a sobrecarga gerada para o algoritmo funcionar [Deitel et al. 2005, Sílberschatz et al. 2004].

4. Trabalhos Correlatos

4.1. Sosim

O sosim é uma ferramenta que foi criada para auxiliar professores e alunos nas disciplinas que abordam conceitos de sistemas operacionais. Esse sistema permite simular os conceitos implementados em sistemas operacionais multiprogramados. Através da ferramenta é possível visualizar de forma animada a gerência de processos e a gerência de memória virtual. Sobre gerência de memória virtual, o sistema permite, entre outras coisas, escolher a política de busca de página, se é por demanda ou antecipada e visualizar uma janela com dados estatísticos, um deles mostra a quantidade de falta de páginas desde a vinda do primeiro processo para execução [Maia 2001].

O sosim é capaz de auxliar no entendimento de diversos conceitos, no entanto, o sistema em si é muito confuso. Várias janelas são exibidas e executam ações ao mesmo tempo, o usuário fica perdido sem saber para onde olhar. A forma como a gerência de memória virtual é tratada é bem superficial. O usuário é capaz de ver que páginas estão o tempo todo sendo trocadas entre a memória e o disco, mas não tem como entender porque isso está acontecendo, nem como a memória virtual foi implementada. O sistema só permite simular o algoritmo de substituição de páginas FIFO.

4.2. SimulaRSO

Esse sistema simula o escalonamento de processos, escalonamento de discos e a paginação de memória. Na simulação de paginação de memória o sistema permite, entre outras

coisas, escolher entre as estratégias de substituição FIFO, algoritmo ótimo e a menos recentemente utilizada, escolher o número máximo de molduras de páginas na memória e visualizar graficamente o estado das molduras de páginas conforme as páginas vão sendo referenciadas[de Araújo Rodrigues and Pereira 2011].

4.3. Simulador de Memória Real e Virtual

Esse trabalho propões simular a gerência de memória real e virtual. O usuário pode inserir ou remover processos de maneira manual ou automática. Esses processos serão alocados na memória com base em algum algoritmo de alocação que pode ser escolhido [Moritz 2004].

O algoritmo de substituição de página implementado é o FIFO. Somente dois processos podem ser carregados, antes de carregá-lo é preciso realizar uma configuração de qual será seu identificador e o tempo de execução entre uma instrução e outra. A tabela de página informa os endereços virtuais e reais da página e se ela está na memória física ou no disco. A memória física mostra que está sendo ocupada também pelo sistema operacional e pelas tabelas de páginas dos processos.

5. Ambiente de Simulação

Na ferramenta desenvolvida, o usuário além de visualizar o funcionamento dos algoritmos de substituição de páginas e as interações que acontecem entre os diversos componentes que compõem esse mecanismo, poderá também implementar seus próprios algoritmos e colocá-los para executar juntamente com a aplicação. A ideia é que depois de estudar o algoritmo o aluno possa por em prática o que ele aprendeu. Normalmente o aluno precisa ter um enorme trabalho para simular esse processo, muitas vezes demora muito mais tempo implementando abstrações do sistema operacional do que se preocupando com a lógica do algoritmo em si. Com o sistema que está sendo proposto, só será preciso se preocupar com a lógica do algoritmo de substituição, pois o sistema já disponibilizará abstrações dos principais recursos necessários para a gerência de memória virtual, como por exemplo, memória principal, disco e tabela de páginas.

Ao final, o aluno poderá visualizar o que foi feito através de animações. É possível também interagir com o gerenciador simulado, adicionando e removendo processos e principalmente referenciando as páginas. Essa última funcionalidade é um diferencial comparado com outras ferramentas. Nas ferramentas correlatas analisadas, o usuário somente interage com o mecanismo de gerência de memória adicionando e removendo processos. Após adicionar um processo, ele fica apenas observando qual será o comportamento. Essa abordagem é de difícil compreensão, pois o aluno precisa prestar bastante atenção para não perder nenhum detalhe. O processo executa e referencia as páginas sem a intervenção do usuário.

Este trabalho propõe uma maior interação entre o usuário e o processo. Além de adicionar o processo, o usuário também vai ditar a ordem com que as páginas vão ser referenciadas em tempo de execução. Isso permite que o usuário tenha um melhor controle sobre o que está acontecendo e consiga criar diversas situações ao mesmo tempo em que já está analisando-as. Os recursos gráficos é outra questão muito importante para este trabalho, pois ele foi idealizado para ser um sistema com visual agradável.

O ambiente de simulação proposto, foi feito considerando a gerência de memória virtual com paginação sob demanda, tendo as páginas com tamanho fixo. O foco do trabalho em si, é somente os algoritmos de substituição de páginas e as interações entre o gerenciador, memória física e disco quando os processos são carregados e demandam espaço de memória. Por tanto, toda a parte de endereçamento e tradução dinâmica de endereço foi simplificada e não será mostrada na visualização da simulação, sendo sugerida como trabalho futuro.

O sistema é composto por um módulo gerenciador, que possui os principais componentes necessários para prover a simulação da gerência de memória virtual. Outra camada também de fundamental importância é a *view*, que irá representar graficamente tudo o que esta acontecendo no módulo gerenciador. Essa camada *view* utiliza o componente JavaFX, o qual provê os recursos necessários para que a visualização desejada seja possível.

As tecnologias utilizadas para desenvolver a ferramenta foram basicamente a linguagem Java e a biblioteca JavaFX. O JavaFX permite a criação de ambientes gráficos de maneira simplificada, foi escolhido pois foi a melhor ferramenta para criação de layout de tela encontrada.

5.1. Módulo gerenciador

Esse módulo foi construído de maneira que o seu funcionamento independe da camada *view*. Podemos realizar os testes realizando chamadas diretas ao módulo via console. Ou se alguém se interessar, nada impede que seja criada uma outra implementação da camada *view* utilizando outros componentes para representar o gerenciador.

Durante o desenvolvimento do trabalho, foi identificado que o determinante para o funcionamento correto de certa estratégia de substituição, é a forma como as dados relevantes para a mesma são mantidos. Normalmente, os algoritmos funcionam em cima de alguma estrutura de dados, mais comumente, filas, listas, árvores e matrizes. Essas estruturas normalmente guardam os endereços das páginas e sempre que uma página é referenciada elas são atualizadas. Essa atualização pode ser somente uma reorganização dos endereços ou pode ser a adição de algum dado que será utilizada depois para determinar qual página será substituída.

Normalmente a sequência de passos utilizada pelo gerenciador para substituir uma página é sempre a mesma. A página que será retornada no método *paginaParaSubstituir* da interface *EstruturaDados*, vai depender de qual estratégia de substituição de páginas a estrutura de dados está atendendo. Para entender melhor o que foi proposto, será mostrada a implementação do algoritmo FIFO sobre essa ótica. A interface EstruturaDados é definida conforme a Figura 2.

```
public void adicionarReferencia(Endereco endereco);
public void removerReferencia(Endereco endereco);
public Endereco paginaParaSubstituir();
public void removerReferencias(int idProcesso);
public Iterator<Endereco> createIterator();
```

Figura 2. Interface Estrutura Dados

Para que o algoritmo FIFO funcione, é necessário que o objeto que implemente a interface EstruturaDados possua uma fila. Assim, sempre que uma página for refe-

renciada, o endereço da mesma será adicionado na fila, exceto se esse endereço já tiver sido adicionado. Sempre que o método paginaParaSubstituir for chamado, o primeiro endereço da fila será retornado. Conforme pode ser observado, o bom funcionamento do algoritmo depende inteiramente da implementação da estrutura.

Se um usuário desejar adicionar uma nova estratégia, basta ele implementar essa interface EstruturaDados de maneira que ela mantenha os dados necessários e que retorne o endereço apropriado da próxima página a ser substituída no momento que ele for solicitado. O módulo gerenciador também suporta outra implementação desse fluxo. Mas na maioria dos casos isso não será necessário.

5.2. Camada view

Na camada de visão, existe um componente para representar cada componente do módulo gerenciador. Toda vez que algum componente do módulo gerenciador sofrer alguma alteração a camada de visão será atualizada para representar o estado atual do gerenciador.

Para que as classes de visão sejam atualizadas sempre que o estado do gerenciador mude e sempre que eventos oriundos de alguma ação do usuário ocorra, foi implementado o padrão de projeto *Observer*. Esse padrão define duas interfaces, uma para objetos observáveis e outra para objetos observadores. Um objeto observável pode ser observado por vários objetos observadores. Sempre que o objeto observado mudar de estado, os objetos que o observam são atualizados de maneira apropriada caso a atualização seja relevante

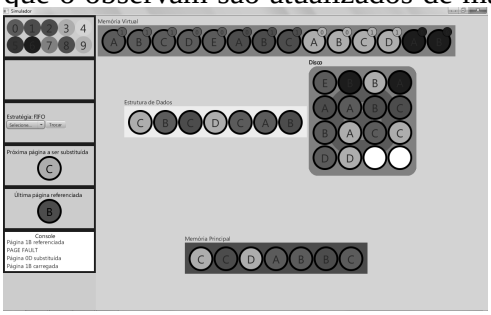

Figura 3. Screenshot do sistema

6. Validação e Testes

Para a validação do modelo proposto, foi aplicada uma aula de gerência de memória virtual utilizando a ferramenta em uma turma de primeiro semestre do curso de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal da Bahia, Campus Salvador, inscritos na disciplina de "Arquitetura de Computadores e Software Básico", que trata do assunto de Gerência de Memória, de forma introdutória. Os conceitos relacionados à gerência de memória virtual foram abordados utilizando a ferramenta como recurso visual, uma vez que o assunto já tinha sido abordado previamente em sala de aula. Após a explicação foi realizada a demonstração dos algoritmos, ao passo em que a teoria envolvida também era explicada. No final da aula os alunos responderam a um questionário sobre o que acharam da ferramenta. Apenas sete alunos estavam presentes na aula. Os resultados obtidos do questionário podem ser visualizados a seguir:

1. O simulador te ajudou a entender a gerência de memória virtual? (Sim/Não) - 85,71% dos alunos respondeu que sim e 14,29% respondeu que não.

2. O simulador te ajudou a entender o funcionamento dos algoritmos de substituição de páginas propostos? (Sim/Não) - 71,43% dos alunos respondeu que sim e 28,57% dos alunos respondeu que não.
3. O que você achou da visualização da simulação? (Ruim/Regular/Boa/Ótima) - 71,43% dos alunos respondeu que a visualização é boa e 28,57% respondeu que é ótima.
4. Existe algum ponto da gerência de memória virtual que você acredita ter faltado na simulação? (Sim/Não) - 100% dos alunos respondeu não.
5. O simulador é fácil de usar? (Sim/Não) - 85,71% dos alunos respondeu que sim e 14,29% respondeu que não.
   A ideia é estender esta análise para alunos da disciplina de Sistemas Operacionais, já que estes estão mais fortemente atrelados a este conteúdo.

7. Trabalhos Futuros

O simulador proposto é o ponto de partida para a criação de uma ferramenta que simule o funcionamento de um sistema operacional completo. Para entender o funcionamento de um sistema operacional, seria necessário muito tempo de estudo devido ao grau de complexidade e as diversas formas de implementações existentes. Além do fato de sair do escopo de um trabalho como este. Sendo assim, a ferramenta proposta simula uma pequena parte de um sistema operacional e deixa margem para que diversos outros trabalhos sejam criados com base no que já foi feito. Algumas sugestões para este trabalho são (a) simulação da tradução dinâmica de endereço virtual para o endereço real; (b) simulação da gerência de processos e (c) simulação da gerência de disco.

Referências

de Araújo Rodrigues, A. and Pereira, C. R. (2011). Simulador de recursos de sistemas operacionais. Acessado em 10/02/2014.

de Oliveira, R. S., Toscani, S. S., and da Silva Carissimi, A. (2010). *Sistemas Operacionais*. Bookman.

Deitel, Deitel, and Choffnes (2005). *Sistemas Operacionais*. Pearson Education Brasil.

Flynn, I. M. and McHoes, A. M. (2002). *Introdução aos Sistemas Operacionais*. Thomson.

Machado, F. B. and Maia, L. P. (2001). Um framework construtivista no aprendizado de sistemas operacionais - uma proposta pedagógica com o uso do simulador sosim.

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

Maziero, C. (2002). Reflexões sobre o ensino prático de sistemas operacionais.

Moritz, G. (2004). Simulador de mecanismos de gerência de memória real e virtual.

Sílberschatz, A., Galvin, P., and Gagne, G. (2004). *Sistemas Operacionais*. Campus.

Stallings, W. (2003). *Arquitetura e Organização de Computadores*. Prentice Hall.

Stuart, B. L. (2011). *Princípios de Sistemas Operacionais*. Cengage Learning.

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