MOSS - UMA FERRAMENTA PARA O AUXÍLIO DO ENSINO DE SISTEMAS OPERACIONAIS

Antonio V. Costa, Arthur Silva, Silvio Fernandes, Francisco Tailanio de Macedo

Departamento de Computação - Centro de Ciências Exatas e Naturais, Universidade Federal Rural do Semi-Árido (UFERSA), Mossoró, Rio Grande do Norte, Brasil.

{antonio.costa,silvio,arthur.alves}@ufersa.edu.br,
tailanio@outlook.com

Resumo. *Este artigo apresenta MOSS, uma ferramenta integrada ao MARS para o ensino/aprendizado de Sistemas Operacionais (SO). Além dela, foram propostas atividades sobre gerenciamento de processos e gerenciamento de memória as quais foram realizadas por voluntários que cursaram SO em diferentes semestres. A quantidade de acerto dos voluntários nas atividades e avaliação feita por eles sobre a ferramenta e as atividades validam a metodologia proposta.*

Abstract. *This paper presents MOSS, an integrated MARS tool for teaching and learning Operating Systems (OS). In addition, activities were proposed to process management and memory management, which were carried out by volunteers who attended OS in different periods. The amount of volunteers' success in the activities and their evaluation of the tool and activities validated the proposed methodology.*

1. Introdução

A disciplina de Sistemas Operacionais está presente em praticamente todo curso de graduação em computação ou informática. Tal disciplina apresenta os conceitos e estratégias algorítmicas que servem como ponte entre o *hardware* e *software*.

As principais funções do SO estão no gerenciamento do acesso ao processador, a memória, a entrada/saída e sistema de arquivos. O gerenciamento resolve a disputa dos recursos físicos e lógicos compartilhados pelos programas, que do ponto de vista do SO são chamados processos, de forma justa para o propósito do SO. Para tais funções, o SO precisa implementar soluções otimizadas para o *hardware* e ainda oferecer interfaces abstratas tanto para o programador quanto para o usuário das aplicações.

Assim, o ensino de SO envolve desafios em relação ao detalhamento de conceitos e implementações na prática, de modo que o aluno experimente algo o mais próximo possível do real em um tempo limitado do curso dessa disciplina.

Portanto, esse artigo propõe a implementação de uma ferramenta (MOSS) para o MARS [K. Vollmar and P. Sanderson 2006], apresentadas nas próximas seções. O objetivo da MOSS é fornecer auxílio didático, de forma que seja possível visualizar as respostas de um SO para processos executando no simulador MARS.

O artigo está organizado da seguinte forma: a seção 2 apresenta referencial teórico e alguns trabalhos relacionados; a seção 3 discute a implementação da

DOI: 10.5753/cbie.sbie.2018.755

ferramenta MOSS; na seção 4 são apresentados os experimento utilizando o MOSS; na seção 5 é mostrado os resultados do experimento; a seção 6 apresenta as conclusões e trabalhos futuros; e a seção 7 lista as referências utilizadas no artigo.

2. Referencial Teórico

Um dos autores mais referenciados no estudo de Sistemas Operacionais (SO) é o Andrews S. Tanenbaum [Tanenbaum 2009]. No seu livro [Tanenbaum 2008], ele apresenta os conceitos e a implementação do SO MINIX [A. S. Tanenbaum 2006] e defende que para entender os conceitos na prática, um estudante de computação precisa "dissecar" o código de um SO. O MINIX evoluiu mas seu livro texto não conseguiu acompanhar, estando atualmente dessincronizados, o que dificulta uma metodologia de ensino usando os dois. Em [Du, Wenliang, and Wang 2008] é apresentado um ambiente de laboratório virtual que utiliza uma infraestrutura híbrida de MINIX e Linux para o ensino de tópicos relacionados a segurança. Também usando um SO real, modificável em atividades durante uma disciplina, está o MiniOS [Otero, Rafael, and Aravind 2015], implementado por meio de um kernel virtual junto com um kernel Linux.

Usando metodologia de gameficação, [Hill, Ray, Blair, and Carver 2003] tem o objetivo de melhorar o aprendizado em SO, com "palavras cruzadas" e o jogo perguntas e respostas "Jeopardy!". Eles também propõem "batalha de threads", inspirado em batalha naval e "jogo da transição de estado dos processos". Usando a abordagem baseada em problema, [Yi-Ran, Cheng, Feng, and Meng-Xiao 2010] foca em escalonamento de processos e o problema de *deadlock.*

No sentido de diminuir a abstração, o uso de simuladores é particularmente interessante. O simulador SOsim é apresentado em [Machado, Berenger, and Maia 2007], que aborda as principais funções de gerenciamento de forma visual e simplificada. Em [Carvalho, Schimitz, Balthazar, Dias, Araújo and Monteiro 2006] é apresentado o S^2O para simulação do escalonamento de processos. Já em [Tonini, Alexssandro and Lunardi 2006] é apresentado um simulador multiplataforma, por ter sido implementado em Java, que já considera os aspectos de escalonamento e gerenciamento de memória. Simuladores mais tradicionais e mais acurados, em relação às funções de um SO, por trabalharem em nível instrucional, como RCOS [Jones and Newman 2002] e, sua versão web RCOS.Java [Jones, David, and Newman 2001]. O Nachos [Christopher, Wayne , Procter and Anderson 1992] é um framework de SO funcional voltado para arquitetura MIPS, inclusive com simulador dessa arquitetura embutido. MIPS é provavelmente a arquitetura mais utilizadas no ensino de Organização e Arquitetura de Computadores (OAC), e, dentre as várias ferramentas voltadas pra ela, destacamos o ambiente MARS (MIPS *Assembler and Runtime Simulator*).

A partir das características do MARS, principalmente das possibilidades de expansão das suas funcionalidades, desde 2015 tem sido experimentada uma metodologia interdisciplinar [Fernandes and Silva 2017]. Nela são propostas atividades de uso do MARS e no desenvolvimento de novas funcionalidades para ele, para o ensino e aprendizagem de OAC, como também para SO, e este último resultou no MOSS.

3. A Ferramenta MOSS

O MARS é um ambiente de desenvolvimento interativo para programação em linguagem assembly MIPS, destinado ao uso educacional. É multiplataforma, disponibiliza um editor de texto, um montador e um simulador. Também oferece um conjunto de chamadas de sistema (*syscall*) que simula diversos serviços de SO durante a execução dos programas e pseudo instruções. Integrado ao simulador há um conjunto de ferramentas (*tools*) que se conectam ao simulador e fornecem informações extras em tempo de simulação. O MARS ainda oferece, em sua implementação, classes abstratas que permitem a criação de novas *syscalls*, novas ferramentas, novas pseudo instruções ou novas instruções [Sanderson and Vollmar 2007].

MOSS (MIPS *O*perating System *Simulator*) é uma ferramenta que integra *syscalls* e *tools* desenvolvidas para o MARS para simulação das principais funções de um sistema operacional, na metodologia interdisciplinar adotada na UFERSA. Na disciplina de SO, é aproveitado o conhecimento prévio de OAC e do MARS, e a ordem da apresentação dos conceitos de [Tanenbaum 2009]. Dessa forma, os alunos são instruídos a entenderem o código do MARS para o desenvolvimento de extensões desse ambiente aplicando a SO. Portanto, baseado nessa metodologia foi desenvolvido o MOSS, utilizando o MARS como ambiente de simulação e teste.

O MOSS implementa 3 (três) gerenciadores de um SO, que são quase independentes: o gerenciador de processo (GP), o gerenciador de memória (GM) e o gerenciador de arquivos (GA), bem como as *syscalls* relacionadas a eles. MOSS foi implementado como uma *tool* do MARS, sendo então encontrado no menu *tool* do ambiente (Figura 1), e quando conectado ao ambiente principal (por meio do botão "*Connect to MIPS*") passa a observar e atuar no estado interno do simulador MIPS.

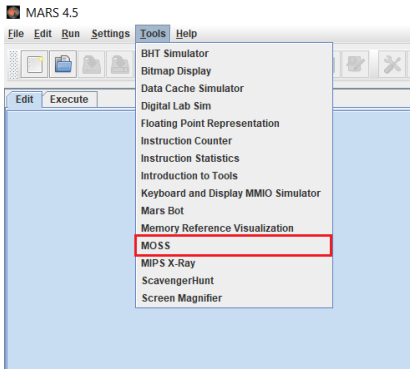


Figura 1. Menu *tools* do MARS com destaque para o MOSS.

Assim, como em todo SO, o elemento central do seu funcionamento é o conceito de processos, ou seja, os programas em execução, logo o GP do MOSS é o único que sempre deve estar ativo, sendo a ativação dos outros 2 opcionais. A Figura 2 mostra a janela principal do MOSS com os 3 gerenciadores escolhidos.

O GP é encarregado de todos os aspectos do controle de "vida" dos processos e *syscalls* relacionadas, ou seja, pela criação, escalonamento, manutenção das informações e término dos processos. No MOSS, o escalonamento pode ser preemptivo e não-preemptivo, escolhido pelo usuário. O preemptivo é aquele que o próprio SO decide o momento de retirar um processo da CPU e colocar outro em seu lugar. Enquanto que o não-preemptivo, é aquele que o próprio processo abre mão da CPU para que outro processo seja escolhido para executar, para isso deve ser invocada uma *syscall* do MOSS no código *assembly*. Este gerenciador implementa somente uma *syscall*

(*SycallMOSS*) que possui 3 funções, a qual deve ser escolhida pelo usuário. Para escolher a função, valor do registrador $t0 (linhas 60, 81 e 88 da Figura 3) deve conter uma das opções: 1 para a criação de processo (*fork*), 2 para a troca de processos voluntária (*processChange*) e 3 para a finalização de processo (*processTerminate)*.

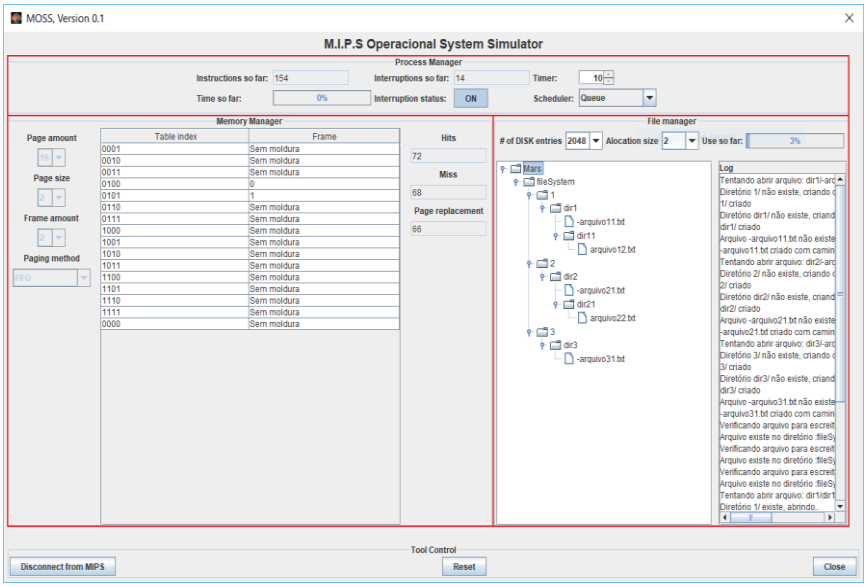


Figura 2. Interface do MOSS com todos os gerenciadores iniciados

```
48  #macro para fork
49  .macro fork (%labelStart, %labelEnd, %priority, %minPriority, %maxPriority)
50      addi $sp, $sp, -24
51      # empilhando
52      sw $a0, 20($sp)
53      sw $a1, 16($sp)
54      sw $a2, 12($sp)
55      sw $a3, 8($sp)
56      sw $v1, 4($sp)
57      sw $t7, 0($sp)
58
59          li $v0, 18
60          li $t7, 1
61          la $a0, %labelStart
62          la $a1, %labelEnd
63          la $v1, %priority
64          la $a3, %minPriority
65          la $a2, %maxPriority
66          syscall
67
68      # desempilhando
69      lw $t7, 0($sp)
```

```
70      lw $v1, 4($sp)
71      lw $a3, 8($sp)
72      lw $a2, 12($sp)
73      lw $a1, 16($sp)
74      lw $a0, 20($sp)
75
76      addi $sp, $sp, 24
77  .end_macro
78
79  #macro para processChange
80  .macro processChange
81          li $t7, 2
82          li $v0,18
83          syscall
84  .end_macro
85
86  #macro para processTerminate
87  .macro processTerminate
88          li $t7, 3
89          li $v0,18
90          syscall
91  .end_macro
```

Figura 3 - Trechos de códigos assembly: definição das macros do MOSS.

No MARS, para fazer uma chamada de sistema o programador deve incluir o código da chamada no registrador $v0, dependendo do tipo pode haver algum parâmetro em outro registrador, e em seguida a palavra reservada *syscall*. No nosso caso, a *syscall* tem o número 18 (linhas 59, 82 e 89 da Figura 3), o qual estava disponível no MARS. Para facilitar o uso da nossa *syscall*, foi criado um arquivo com macros para todas as opções, como pode ver visto no trecho de código da Figura 3. O arquivo das macros deve ser incluído no código *assembly* que irá utilizar as funcionalidades do MOSS, conforme a Figura 4 (linha 1). Em seguida, basta chamar as macros passando os parâmetros, quando necessário, para utilizar as funções da SyscallMOSS.

Para criar um novo processo deverá ser chamada a macro *fork* passando os seguintes parâmetros (linhas 4, 5 e 6 da Figura 4): uma *label* para a primeira instrução do processo (linhas 9, 13 e 20 da Figura 4), uma *label* para a última instrução do

processo (linhas 13, 20 e 27 da Figura 4, nesse caso reaproveitamos a *label* de início do processo seguinte como a *label* de fim do processo anterior), o valor da prioridade inicial do processo, o valor da prioridade mínima em que o processo pode chegar e o valor da prioridade máxima que o processo pode ter.

```
1  .include "macros.asm"
2  .text
3  #criação dos processos
4          fork(Programa1, Programa2, 8, 0, 15)
5          fork(Programa2, FimPrograma2, 10, 0, 15)
6          fork(Idle,Programa1, 6, 0, 15)
7  #esclonando o primeiro processo
8          processChange
9  Idle:
10         loop:
11         add $zero,$zero,$zero
12         j loop
13 Programa1:
14         addi $s1, $zero, 1 # valor inicial do contador
15         addi $s2, $zero, 10 # valor limite do contador
16         loop1: addi $s1, $s1, 1
17         beq $s1, $s2, fim1
18         j loop1
19 fim1: processTerminate
20 Programa2:
21         addi $s1, $zero, -1 # valor inicial do contador
22         addi $s2, $zero, -10 # valor limite do contador
23         loop2: addi $s1, $s1, -1
24         beq $s1, $s2, fim2
25         j loop2
26 fim2: processTerminate
27 FimPrograma2:
```

Figura 4 - Trechos de códigos assembly: utilização das macros para criação, troca e término dos processos.

Já para as opções de *ProcessChange* e *ProcessTerminate*, basta chamar a macro referente a opção desejada sem passagem de parâmetro.

Para simulação do ambiente concorrente de um SO no MARS, todos os processos são escritos em *assembly* em um mesmo arquivo, que deve ser compilado pelo próprio MARS. Para criar um processo a macro referente ao *Fork* deve ser invocada, passando os devidos parâmetros (linhas 4, 5 e 6 da Figura 4).

Se o escalonamento for não-preemptivo, dentro do código do processo deve ser chamada a macro referente ao *ProcessChange* (linha 8 da Figura 4) e no término do processo a chamada da macro *ProcessTerminate* (linhas 19 e 26 da Figura 4). Se o escalonamento for preemptivo, o programador não deve utilizar a macro *ProcessChange*, mas deve ter configurado o *timer* com a quantidade de instruções executadas que provocam a troca de processo automática.

Quando acontece a chamada *Fork*, o GP cria uma PCB (*Process Control Block*) exclusiva para o processo, o qual é identificado pelo seu PID (*Process Identifier*). Nesse PCB são armazenadas todas as informações do processo, como o valor atual dos registradores do MIPS para este processo, estado do processo (pronto, executando e bloqueado), entre outras. A cada *ProcessChange* ou interrupção feito pelo *timer*, o GP salva o contexto do processo, ou seja, obtém as informações dos registradores físicos do MIPS e as demais e as salva em sua PCB, colocando esse processo em estado de "pronto". Em seguida, executa um algoritmo de escalonamento para escolher um processo no estado de "pronto". Então, a PCB do processo escolhido é localizada, é feita a troca de contexto, e o estado do processo escolhido é alterado para "executando". O *ProcessTerminate* destrói a PCB do processo e o impossibilita de ser executado novamente. Esse gerenciador cria a ilusão do compartilhamento do processador fazendo uso de um despachante, que pode ser selecionado na interface. Na interface gráfica do MOSS, são exibida as informações sobre a quantidade de instruções executadas pelo processo e quantidade de interrupções restante para que o *timer* faça a interrupção (quando o escalonamento preemptivo for escolhido), de forma numérica e em porcentagem.

O GM mantém o controle sobre quais partes da memória estão em uso e quais não estão, podendo alocar memória aos processos quando eles precisam e liberando-nas quando estes terminam. Efetua também a paginação e a troca de páginas, ou parte delas

(*swapping*), entre memória e disco quando a memória principal não é suficiente. Na interface do GM é possível configurar a quantidade de páginas por processo, tamanho de cada página, quantidade de molduras por processo, o algoritmo de substituição de páginas (Não usada recentemente, fila, segunda chance e menos recentemente usada). Ele mostra as informações sobre o estado atual da tabela de páginas virtuais do processo em execução, assim como à qual moldura ela está associada a página virtual. Esse gerenciador não oferece *syscall*. Mas todos os endereços de memória do segmento de código são monitorados, de modo que acessos fora do limites do espaço de endereçamento dos processos são proibidos pelo MOSS. Os endereços de memória são tratados como virtuais, de modo que precisam ser traduzidos e verificados na respectiva tabela. Em caso de *page miss*, o algoritmo de substituição é invocado. Todo esse processo é exibido graficamente para o usuário.

Por último, o GA, atua como um simplificador para a organização, o compartilhamento e o gerenciamento de grandes volumes de informação de forma abstrata para o programador. Na interface do MOSS para este gerenciador, é possível configurar a quantidade de entradas no disco (tamanho do disco) e o tamanho de cada alocação. Na interface do gerenciador é visto informações sobre a porcentagem de disco usado, assim como uma árvore e um *log* que mostra todos os diretórios, subdiretórios e arquivos criados pelas chamadas de sistema fornecida pelo MARS.

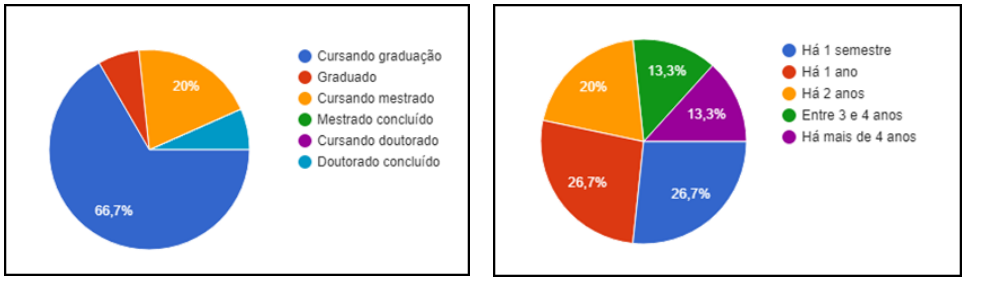
O GA não implementa nenhuma nova *syscall*, ele utiliza as já existem no MARS para o controle e manuseio de arquivos, cujos código vão de 13 a 15. Para tanto, alteramos essas *syscalls* para chamarem o código do nosso GA antes de executarem seus próprios códigos, não alterando em nada seus funcionamentos.

4. Experimento de Utilização do MOSS

Para validação do MOSS como ferramenta didática, foi desenvolvido um tutorial[1] para configuração/utilização do mesmo, bem como um texto explicativo detalhando essa ferramenta. Tal tutorial propõe 2 atividades prática (uma para o GP e outra para o GM) e uma avaliação sobre a ferramenta e as atividades propostas. Cada atividade prática propõe 5 questões de múltipla escolha e ao final retorna para o usuário um *feedback* com as respostas corretas e a pontuação total dele. Ainda não foram propostas atividades para o GA pois as funções atuais permitem pouca interação com o usuário, e por isso, novas funcionalidades estão sendo incluídas em tal gerenciador.

O experimento foi realizado com 15 voluntários da UFERSA que cursaram SO em diferentes semestres, permitindo uma amostragem mais heterogênea. A distribuição de escolaridade (Figura 5(a)) e há quanto tempo estes voluntários concluíram a disciplina (Figura 5(b))) estão sumarizados a seguir. A maioria dos voluntários ainda está cursando a graduação (66,7%), seguido por estudantes do mestrado em ciência da computação (20%). Também houve 6,7% graduado e mesmo número para doutorado concluído. Em relação ao tempo de conclusão da disciplina, fica mais claro a heterogeneidade dos voluntários, onde a maioria cursou há pouco tempo (26,7% há 1 semestre e 26,7% há um ano), contudo os demais tempos de conclusão estão entre 20% e 13%.

[1] Diponível em: https://docs.google.com/document/d/18c72Xg4mte3OM1B9X9j7sGuT1qk-1DlVR5u92czES_Y/edit



(a) Escolaridade (b) Tempo de conclusão de SO

Figura 5 - Distribuição dos voluntários

5. Resultados do Experimento

Cada uma das duas atividades práticas usando MOSS possui 5 questões, e cada questão com valor de 2 pontos, de modo que 100% de acerto corresponde a 10 pontos. Os gráficos Figura 6 apresentam a frequência do total de pontos obtidos pelos voluntários na realização das Atividade 1 (GP) e Atividade 2 (GM). Nota-se que, em ambas as atividades a maioria dos participantes obtiveram notas iguais ou superiores a 6. Além disso, é visto que a maior frequência de notas em ambas as atividades foram em notas consideradas altas (8 no GP e 10 no GM). Com isso, o objetivo de aplicar uma metodologia (texto, ferramenta, exercício), sem supervisão, para ensino/aprendizado de SO, foi atingido, mesmo que mais de 40% dos participantes tenham concluído a disciplina há no mínimo 2 anos, conforme perfil apresentado na Figura 5

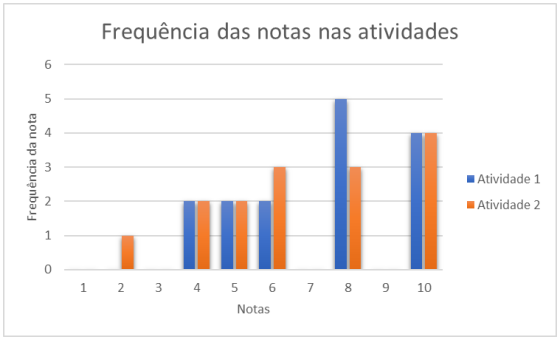


Figura 6: Frequência das notas das atividades propostas.

A média de pontos da Atividade 1 foi igual a 7,33 com desvio padrão 2,1. A Atividade 2 teve média igual a 6,8 e desvio padrão de 2,4. Isso corresponde a uma compreensão das atividades pela maioria, uma razoável dispersão das notas.

Com relação a avaliação da metodologia, foram realizadas 6 perguntas sobre a ferramenta e 6 sobre as atividades propostas. As perguntas foram objetiva, com 5 opções de resposta, usando escala de Likert, ou seja, variando do mínimo ao máximo em qualidade. A Tabela 1 apresenta as perguntas da avaliação.

QUESTÕES SOBRE MOSS	QUESTÕES SOBRE AS ATIVIDADES
1) Quão amigável é a ferramenta MOSS?	1) Quão auto-explicativas são as atividades em geral?
2) Quão fácil de configurar MOSS?	2) Como classifica o feedback do formulário das atividades práticas após enviar suas respostas?

3) Quão útil é o MOSS para aprender sobre Gerenciamento de Processos?	3) Quão útil você achou as atividades para o ensino/aprendizados para Gerenciamento de Processos?
4) Quão útil é o MOSS para aprender sobre Gerenciamento de Memória?	4) Quão útil você achou as atividades para o ensino/aprendizados sobre Gerenciamento de Memória?
5) Como você classifica a MOSS para relacionar os assunto de Gerenciamento de Processos e Gerenciamento de Memória em SO?	5) Qual a classificação geral das atividades para utilização da ferramenta MOSS?
6) Como você classifica a documentação da ferramenta MOSS?	6) Quão aplicável são as atividades que você realizou para alunos que estão cursando SO?

A Figura 7 resume as opiniões dos voluntários sobre o MOSS por meio da avaliação. Em média cada questão está acima de 4, o que indica que os participantes aprovaram a ferramenta nas questões avaliadas. As menores notas (2,0 pontos) nas questões 2, 3 e 4, pode ser interpretada como a falta de um professor para orientação na utilização da ferramenta. Ainda assim, em todas as questões também foram obtidas notas máximas.

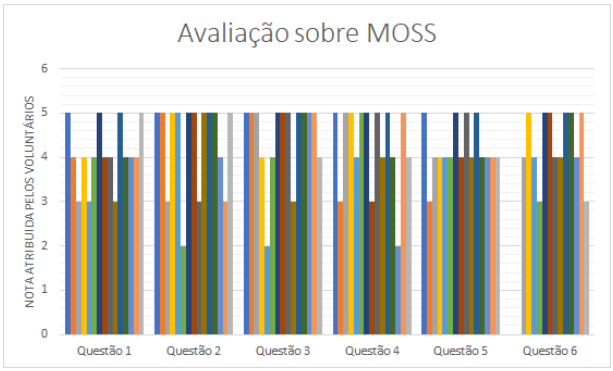


Figura 7 - Avaliação do MOSS pelos voluntários

Na avaliação sobre as atividades (Figura 8), a média de notas é ainda maior para cada questão e nota mínima (3,0 pontos) nas questões de 2 a 6. Da mesma forma que a anterior, nessa avaliação todas as questões também obtiveram nota máxima.

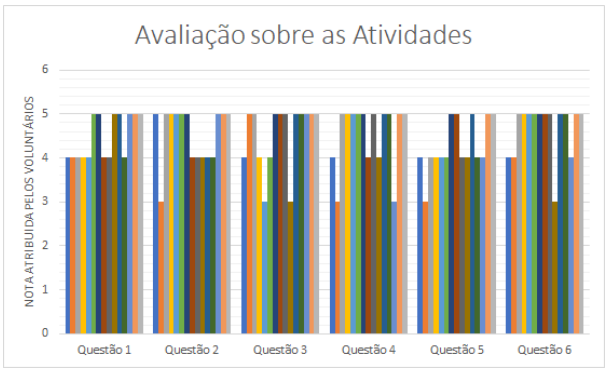


Figura 8 - Avaliação das atividades pelos voluntários

6. Conclusões e trabalhos futuros

Este artigo apresentou MOSS, uma ferramenta didática para Sistemas Operacionais (SO) integrada ao MARS, a qual estende e cria novos *syscalls* e *tools*.

MOSS inclui, de forma parametrizável, os gerenciadores de processo, de memória e de arquivos, os quais são utilizados a partir do código em *assembly* do usuário executado no MARS. Adicionalmente, desenvolvemos atividades práticas para uso desses gerenciadores, as quais foram aplicados a voluntários que já cursaram a disciplina de SO, por meio de um tutorial. Esse tutorial guia seus leitores, em uma ordem adequada, para aprender sobre a ferramenta, configurar e usá-la de forma incremental. Além disso, também disponibiliza um texto de ajuda sobre a ferramenta com acesso fácil ao longo do tutorial. Por fim, o tutorial leva o leitor para uma avaliação sobre a ferramenta e em seguida sobre as atividades práticas.

A partir da avaliação feita pelos voluntários foi colhido bons indicadores de estado atual da ferramenta. Pode-se perceber uma grande aceitação por parte dos que a utilizaram e compreensão dos conceitos e atividades propostas. Tanto as atividades (a partir das questões com maior frequência de erros) quanto às avaliações feita por eles sobre a metodologia são bons indicativos para melhorias necessárias no MOSS e nas atividades propostas.

Como trabalhos futuros, podemos usar a infraestrutura do ger. de processos para orientar outros alunos a desenvolverem soluções de mutexes, monitores, semáforos e implementação dos problemas clássicos de IPC para testá-los. É possível implementar novos algoritmos de escalonamento, bem como modificar os critérios de alteração de prioridade dinâmica já implementado. O ger. de memória atualmente monitora apenas os endereços do segmento de código, assim, também é possível estenderem para o segmento de dados, onde existe permissão de leitura e escrita. Ainda é possível fazer uma integração maior entre o ger. de memória e de arquivos, durante o processo de substituição de página, envolvendo o bloqueio o processo, acesso às informações de localização no disco e etc. E com o ger. de arquivos seria interessante exibir na interface gráfica uma forma mais clara e didática do acesso aos *inodes* e blocos do disco. Com exceção do *timer*, todo um ger. de entrada/saída pode ser desenvolvido. E ainda há espaço para integração com um compilador de uma linguagem em alto nível para *assembly* do MIPS, o qual pode aumentar a interdisciplinaridade. Outro trabalho é o desenvolvimento de uma nova *tool* que funcione como um interpretador de comandos para o MOSS. Outros planos incluem aprimorar o suporte à depuração e realce do conteúdo dos processos, memórias e arquivos modificados passo a passo na execução, como a exibição dos nós alocados por arquivos em uma matriz, a referência de blocos alocados pelo arquivo separados por cores em uma matriz de pixels personalizada para tal e a inclusão de uma tabela no gerenciador de processo com informações de cada processo. Após todos os aprimoramentos pretende-se submeter a ferramenta para o grupo que mantém o MARS para que ela possa fazer parte das *tools* oficiais desse ambiente.

7. Referências

Carvalho, D. S, Balthazar, G. R. , Dias, C. R. , Araújo, M. A. P. and Monteiro, P. H. R. (2006) “Simulador para a Prática de Sistemas Operacionais”. Revista Eletrônica da FMG 3.

Christopher, Wayne A., Procter, S. J., and Anderson T. E. (1992) “The Nachos Instructional Operating System”. EECS Department, University of California, Berkeley, novembro. http://www2.eecs.berkeley.edu/Pubs/TechRpts/1992/6022.html.

Du, Wenliang, and Wang, R. (2008) "SEED: A Suite of Instructional Laboratories for Computer Security Education". J. Educ. Resour. Comput. 8, no 1 (março): 3:1–3:24. https://doi.org/10.1145/1348713.1348716.

Fernandes, Silvio and Silva, Ivan Saraiva. (2017). Relato de Experiência Interdisciplinar Usando MIPS. In *International Journal of Computer Architecture Education*, V.6, n. 1, pág. 52, SBC.

Hill, J., Ray, C. K., Blair, J. R., and Carver Jr, C. A. (2003) "Puzzles and games: addressing different learning styles in teaching operating systems concepts," in ACM SIGCSE Bulletin, vol. 35, pp. 182–186.

Jones, D. and Newman, A. (2002). A Constructivist-based Tool for Operating Systems Education. In P. Barker & S. Rebelsky (Eds.), Proceedings of ED-MEDIA 2002--World Conference on Educational Multimedia, Hypermedia & Telecommunications (pp. 882-883). Denver, Colorado, USA: Association for the Advancement of Computing in Education (AACE). Retrieved March 29, 2018 from https://www.learntechlib.org/p/10276/.

Jones, David, e Newman, A. (2001) "RCOS.java: a Simulated Operating System with Animations". In Proceedings of the Computer-Based Learning in Science Conference. Rep. Tcheca.

Machado, Berenger, F., and Maia, L. P. (2007), Arquitetura de Sistemas Operacionais. 4o ed. LTC.

Otero, R., Rafael, and Aravind, A. A. (2015) "MiniOS: An Instructional Platform for Teaching Operating Systems Projects". In Proceedings of the 46th ACM Technical Symposium on Computer Science Education, 430–435. SIGCSE '15. New York, NY, USA: ACM. https://doi.org/10.1145/2676723.2677299.

Sanderson, P. and Vollmar, K. (2007) "An Assembly Language I.D.E. To Engage Students Of All Levels," presented at the 2007 CCSC.

Tanenbaum, A. S. (2008), Sistemas Operacionais. Projeto e Implementação, 3rd ed. Bookman.

Tanenbaum, A. S. (2009) Sistemas Operacionais Modernos, 3a. ed. São Paulo: Pearson.

Tanenbaum, A. S. "MINIX 3" (2006) [Online]. Disponível em: http://www.minix3.org/.

Tonini, Gustavo Alexssandro, and Lunardi, S. C. (2006) "Simulador para o Aprendizado de Sistemas Operacionais". UFN.

Vollmar, K. and Sanderson, P. (2006) "MARS: An Education-Oriented MIPS Assembly Language Simulator," in ACM SIGCSE.

Yile, F. (2016) "Utilizing the Virtualization Technology in Computer Operating System Teaching". In 2016 Eighth International Conference on Measuring Technology and Mechatronics Automation (ICMTMA), 885–88, https://doi.org/10.1109/ICMTMA.2016.213.

Yi-Ran, H., Cheng, Z., Feng, Y., and Meng-Xiao, Y. (2010) "Research on teaching operating systems course using problem-based learning," in 5th International Conference on Computer Science & Education, 2010, pp. 691–694.