Universidade Federal Rural do Semiarido
Departamento de Ciências Exatas e Naturais
Ciência da Computação
Prof. Sílvio Fernandes

Trabalho de Sistemas Operacionais – Unidade II

- O trabalho pode ser feito em dupla ou individual
- O código da implementação e um relatório técnico descrevendo a implementação deve ser entregue no dia: **11/10/2016**
- Corresponde a **2 pontos extra** na nota da II Unidade
- A implementação deve ser feita no MARS versão 4.5, disponível em <http://courses.missouristate.edu/kenvollmar/mars/>
- A implementação corresponde a uma infraestrutura de sistema operacional para gerenciamento de memória.
- Para se ter acesso ao código do MARS, após baixar o arquivo .jar utilize uma ferramenta de descompactação (ex: Winrar) e extraia seu conteúdo. Em seguida crie um projeto em uma IDE para Java (ex: Eclipse) com os arquivos extraídos.
- O "Apêndice A" do livro "PATTERSON, D. A. ; HENNESSY, J.L. Organização e projeto de computadores – a interface hardware software. 3. ed. Editora Campus, 2005" apresenta detalhes sobre tratamento de exceção no MIPS e chamadas de sistema.
- Esse trabalho deve ser uma continuação do trabalho da Unidade I, ou seja, tudo que foi implementado no primeiro trabalho servirá de base para este.

Descrição da Implementação: Gerenciador de memória principal

- Adicione atributos na sua classe PCB (*Process Control Block*) para manter informações a respeito do gerenciamento de memória do processo tais como:
  - Registrador de limite superior da memória do processo
  - Registrador de limite inferior da memória do processo
  - Ponteiro para segmento de código
  - Ponteiro para segmento de dados
  - Ponteiro para segmento de pilha
  - Entre outros que achar necessário
- Criar uma classe de gerenciador de memória com atributos globais para o todos os processos, tais como:
  - Tamanho de bloco de alocação de memória (página virtual)
  - Quantidade máxima de blocos de alocação por processo
  - Criar uma lista encadeada para gerenciamento de memória livre
  - Configuração do algoritmo de substituição de páginas da memória virtual
  - Entre outros que for necessário
- Criar uma classe "entrada da tabela virtual" com no mínimo os atributos

  - Referenciada
  - Modificada
  - Proteção (bits R, W e X)
  - Presente/ausente
  - Número da moldura
- Criar uma classe “tabela virtual” com uma coleção de “entradas da tabela virtual”
- O gerenciador de memória deve implementar memória virtual simulada por meio de uma ferramenta que deve ser conectada ao MIPS
  - Para implementar a ferramenta siga o tutorial disponível em: http://courses.missouristate.edu/KenVollmar/mars/tutorial.htm
  - Use como exemplo a ferramenta já existente “MemoryReferenceVisualization”
  - Essa ferramenta deve mostrar o estado atual da memória virtual de cada processo e as configurações escolhidas
  - Não há necessidade de modificações na memória do simulador, que do ponto de vista da ferramenta proposta funcionará como o “disco” onde sempre poderão ser obtidos os dados/instruções requisitadas. Assim, neste texto a memória do simulador será chamada de disco e a memória proposta simplesmente de memória.
  - É necessário implementar uma MMU (*Memory Manegment Unit*) que mantém a tabela de memória virtual para traduzir endereços virtuais em físicos
  - Quando um processo é criado (*fork*) o código e os dados do processo são transferidos do disco para memória na primeira página virtual
  - A cada acesso a memória a MMU verifica se a página está alocada na tabela
    - Se estiver traduz para o endereço físico e faz o acesso
    - Se não estiver
      - Verifica se a quantidade máxima de páginas foi atingida
        - Se sim faz a substituição pelo novo bloco
        - Se não aloca o novo bloco em um espaço vazio
  - Implementar os algoritmos de substituição de página: NRU, FIFO, Segunda chance, LRU
    - Fazer uma contagem dos passos realizados por cada algoritmo que pode representar o “esforço” ou eficiência de cada um
  - Considerar que um acesso a memória dura 1 unidade de tempo enquanto que acessar o disco dura 10 unidades
  - A ferramenta pode apresentar em tempo real ou um relatório no final de todas as estatísticas calculadas
    - Quantidade de hits na memória
    - Quantidade de miss na memória (acessos ao disco)
    - Quantidade de substituições de blocos
    - Tempo de “execução” do programa (soma dos tempos de acesso)
    - Tempo de acesso ao disco apenas
    - Tempo de acesso a memória apenas

        - Tempo de busca por espaço livre na memória
        - Tempo de busca pelo bloco a ser substituído
- Para testar o gerenciador de memória
    - Utiliza um código assembly do *main* que cria outros processos
    - Os demais processos podem ter laços de repetição e instruções quaisquer em quantidades variadas
    - Os parâmetros do gerenciador devem ser modificados e para cada combinação executar o mesmo teste e anotar o comportamento do simulador
        - Utilizar as informações de contagem dos algoritmos usados
    - Fazer comparação entre os dados obtidos para cada combinação de configuração e apontar a melhor configuração para este teste