Universidade Federal de Minas Gerais
Departamento de Ciência da Computação
Algoritmos e Estruturas de Dados III
2º Semestre de 2012

Trabalho Prático 3 - Memória Virtual

Esse trabalho tem como objetivo a implementação de um simulador de sistema de memória virtual (SMV). Sistemas de memória virtual fornecem uma abstração para a hierarquia de memória de uma dada arquitetura de modo que programas clientes enxerguem um único espaço de endereçamento sequencialmente acessível. Nesse trabalho o aluno deve implementar uma versão simplificada de um SMV de modo a exercitar os conceitos de gerenciamento de memória e localidade de referência.

Problema

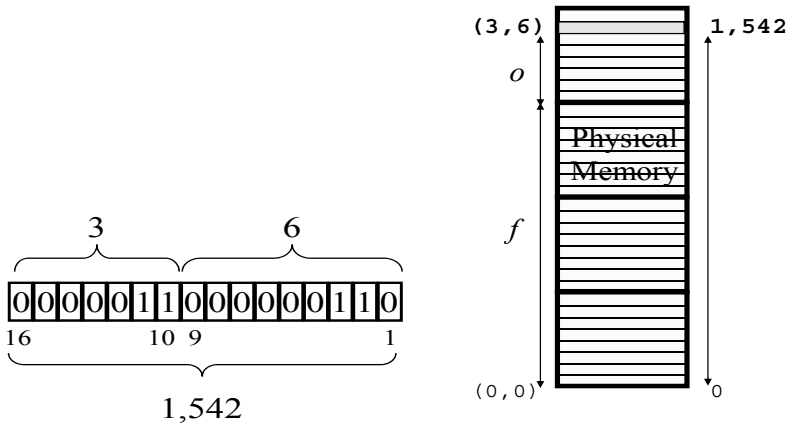
O SMV a ser implementado consiste basicamente em apenas um nível de paginação, sem caches ou otimizações. A entrada do problema consiste em uma sequência de acessos à posições na memória visível do processo. Os dados que residem nessas posições podem estar armazenados em memória primária (*page hit*) ou armazenados em disco (*page miss*, *page fault* ou falha). Em sistemas reais uma verificação deve ser feita para garantir que essa posição pertence ao processo requerente, caso contrário um erro deve ser emitido (*segmentation fault*). Além disso, diferentes tipos de acessos existem, como escrita, leitura, alocação, etc. Para propósito de simplificação, considere que todos os acessos são legais e de leitura.

Uma vez feito um acesso o sistema implementado deve primeiro verificar se os dados requeridos residem em memória primária. SMVs utilizam o conceito de página, que representa um bloco de dados e funciona como uma unidade indivisível de movimentação de dados. Se um dado foi acessado e precisa ser carregado em memória primária, toda a página onde o dado reside será carregada também. A página é carregada no que é chamada moldura de página (*page frame*). Portanto, quando um dado é acessado, deve-se saber em qual página esse dado está e se tal página reside em memória primária. Caso sim, o dado é retornado normalmente, caso contrário, a página deve ser carregada em memória. No evento de uma falha (*page miss*), se houver espaço em memória a página acessada é carregada e o registro de onde a página está é atualizado. Se não houver espaço ocorre algo chamado reposição de página. Alguma página deve ser retirada para que a nova página seja carregada em seu lugar. A escolha de qual página será removida pode ser feita de inúmeras formas. Nesse trabalho você deve implementar e comparar diferentes políticas de reposição:

1. **FIFO:** A página que está residente a mais tempo é escolhida para remoção/.
2. **LRU:** A página acessada a mais tempo deve ser escolhida para remoção.
3. **LFU:** A página com a menor quantidade de acessos deve ser escolhida para remoção.

Os acessos serão representados por inteiros indicando a posição da memória virtual requerida. Para mapear essas posições para suas respectivas páginas os inteiros devem ser interpretados como dois blocos de bits. Seja o espaço de endereçamento da memória física de N bits e o tamanho da página de P bits, cada acesso pode ser lido como os P bits da direita sendo a posição daquele byte

dentro da página e os N-P bits da esquerda sendo o identificador da página onde o byte reside. Por exemplo, em uma configuração de memória física de 2^{16} bytes e páginas de 512 bytes de tamanho, a posição 1542 reside no byte 6 da página 3, como ilustrado pelas imagens abaixo.



O sistema implementado deve permitir a configuração do tamanho da memória física disponível, assim como o tamanho de cada página. A memória virtual sempre será endereçável por 32 bits e o espaço em disco pode ser considerad0 como arbitrariamente grande. Considere que o tamanho da memória física é sempre um múltiplo do tamanho da página e que no início da simulação a memória primária está vazia, portanto qualquer acesso provoca uma falha.

Entrada e Saída

Seu programa deverá ler de um arquivo de entrada um conjunto de configurações e sequências de acessos a serem simuladas. O programa deverá então escrever em um arquivo de saída o número de falhas ocorridas em cada sequência para cada uma das três políticas de reposição implementadas. Um exemplo de execução é dado abaixo.

```
./tp3 input.txt output.txt
```

O arquivo de entrada possui um inteiro K na primeira linha onde K é o número de instâncias a serem simuladas. Em seguida as K instâncias são definidas da seguinte forma. A primeira linha possui três inteiros, o tamanho em bytes da memória física, o tamanho em bytes de cada página e o número N de acessos. A linha seguinte contém N inteiros representando as N posições da memória virtual acessadas sequencialmente.

O arquivo de saída possui K linhas, uma para cada instância de entrada. Cada linha deverá conter 3 inteiros: o número de falhas utilizando FIFO, o número de falhas utilizando LRU e o número de falhas utilizando LFU. Um exemplo de arquivo de entrada e saída esperados é dado abaixo.

Entrada:

```
1
8 4 10
0 2 4 2 10 1 0 0 6 8
```

Saída:

```
6 5 5
```

Análises

O funcionamento da política de reposicão adotada depende fortemente dos parâmetros de configuração do sistema de memória virtual e do perfil de acesso aos dados. Neste trabalho você deve analisar e discutir como estes fatores influenciam no desempenho de cada uma das políticas implementadas. Serão fornecidos no fórum da disciplina arquivos de entrada com diferentes configurações e sequências de acesso. Para cada um destes, você deve realizar as seguintes análises:

1. Calcular a localidade de referência temporal.
2. Calcular a localidade de referência espacial.
3. Gerar o histograma das distâncias de acessos.
4. Gerar o histograma das distâncias de pilha.
5. Gerar um gráfico "Tamanho da página" × "Bytes movimentados".
6. Gerar um gráfico "Tamanho da memória" × "Falhas".

Consulte o material da disciplina para entender como obter as análises acima.

O item 5 tem como objetivo estudar o *trade-off* entre o tamanho da página e o volume de dados transferidos entre disco e memória (falhas × tamanho da página). Páginas maiores irão certamente diminuir o número de falhas, porém vão causar o carregamento desnecessário de muitos dados.

O item 6 ajuda na tarefa de estimar o tamanho ideal da memória física disponível. É natural que quanto mais memória física disponível menos falhas ocorrerão, porém esse aumento acarreta um custo financeiro de se ter maiores memórias. O gráfico pedido permite a identificação de pontos onde a diminuição das falhas é mais acentuada.

Uma vez gerados todos os gráficos, estime os valores ideais do tamanho da memória física e do tamanho das páginas tal que o número de falhas seja o menor possível para cada uma das sequências de acesso. Confirme suas suposições com experimentos. Por fim, compare as políticas de reposição de páginas implementadas e discuta como os diferentes perfis de acesso à memória influenciam no desempenho destas.

Entrega

- A data de entrega desse trabalho é **4 de Dezembro**.
- A penalização por atraso obedece à seguinte fórmula $2^{d-1}/0.32\%$, onde d são os dias úteis de atraso.
- Submeta apenas um arquivo chamado <numero_matricula>_<nome>.zip. Não utilize espaços no nome do arquivo. Ao invés disso utilize o caractere '_'.
- Não inclua arquivos compilados ou gerados por IDEs. **Apenas** os arquivos abaixos devem estar presentes no arquivo zip.
  - Makefile
  - Arquivos fonte (*.c e *.h)
  - Documentacao.pdf

- Não inclua **nenhuma pasta**. Coloque todos os arquivos na raiz do zip.
- Siga rigorosamente o formato do arquivo de saida descrito na especificação. Tome cuidado com whitespaces e formatação dos dados de saída
- NÃO SERÁ NECESSÁRIO ENTREGAR DOCUMENTAÇÃO IMPRESSA!
- Será adotada **média harmônica** entre as notas da **documentação e da execução**, o que implica que a nota final será 0 se uma das partes não for apresentada.

Documentação

A documentação não deve exceder 10 páginas e deve conter pelo menos os seguintes itens:

- Uma **introdução** do problema em questão.
- **Modelagem e solução proposta** para o problema. O algoritmo deve ser explicado de forma clara, possivelmente através de pseudo-código e esquemas ilustrativos.
- **Análise de complexidade** de tempo e espaço da solução implementada.
- **Experimentos** variando-se o tamanho da entrada e quaisquer outros parâmetros que afetem significavelmente a execução.
- Especificação da(s) **máquina(s) utilizada(s)** nos experimentos realizados.
- Uma breve **conclusão** do trabalho implementado.

Código

- O código deve ser obrigatoriamente escrito na **linguagem C**. Ele deve compilar e executar corretamente nas máquinas Linux dos laboratórios de graduação.
- O utilitário ***make*** deve ser utilizado para auxiliar a compilação, um arquivo *Makefile* deve portanto ser incluído no código submetido.
- As estruturas de dados devem ser **alocadas dinamicamente** e o código deve ser **modularizado** (divisão em múltiplos arquivos fonte e uso de arquivos cabeçalho .h)
- **Variavéis globais** devem ser evitadas.
- Parte da correção poderá ser feita de forma automatizada, portanta **siga rigorosamente os padrões de saída especificados**, caso contrário sua nota pode ser prejudicada.
- O arquivo executável deve ser chamado tp3.
- **Legibilidade e boas práticas** de programação serão avaliadas.