Segundo Trabalho[1] de INF 1019 – 2013.1

Data de entrega: domingo, 23 de junho de 2013

1- Introdução e Objetivo

Nesse trabalho, você deverá implementar um simulador de memória virtual, **sim-virtual**, em linguagem C. Neste simulador você criará as estruturas de dados e os mecanismos de mapeamento de memoria (lógico -> físcico) necessários para realizar a paginação, e deverá implementar três *algoritmos de Substituição de Páginas:* o Not-Recently-Used (NRU), o Least-Recently-Used (LRU) e o de Segunda Chance (SEG).

O simulador receberá como entrada um arquivo que conterá a sequência de endereços de memória acessados por um programa real (de fato, apenas uma parte da sequência completa de acessos a memoria). Esses endereços estarão escritos como números hexadecimais, seguidos por uma letra R ou W, para indicar se o acesso foi de leitura ou escrita. Ao iniciar o simulador, será definido o tamanho da memória (em quadros) para aquele programa e qual o algoritmo de substituição de páginas que será utilizado. O simulador deve, então, processar cada acesso à memória para atualizar os bits de controle de cada página/quadro, detectar falhas de páginas (*page faults*) e simular o processo de substituição de páginas e carga de uma nova página em um quadro disponibilizado. Durante a simulacrao, estatísticas devem ser coletadas, para gerar um relatório curto ao final da execução.

O simulador deverá ter os seguintes quatro argumentos de linha de commando:

a) o algoritmo de susbtituição de página sendo simulado (LRU/NRU/SEG)
b) o arquivo contenxo a sequência de endereços de memoria acessados (arq.log)
c) o tamanho de cada página/quadro de página em KB (faixa de valores a ser suportada 4 a 64 KB)
d) o tamanho total de memoria física hipoteticamente disponível em KB (faixa de valores 128 a 16384 = 16 MB).

Ou seja, uma invocação do simulador:

```
sim-virtual  LRU arquivo.log 4 128
```

indicaria um tamanho de página de 4KB e uma memória fisica de 128 KB.

2- Formato do arquivo de entrada

[1] O enunciado foi adaptado de um trabalho prático do prof. Dorgival Guedes Neto (DCC/UFMG)

Cada linha do arquivo (.log) conterá um endereço de memória acessado (em representação hexa-decimal), seguido das letras R ou W, indicando um acesso de leitura ou escrita, respectivamente. Por exemplo:

```
0044e4f8 R
0044e500 R
0044e520 R
0700ff10 R
2f6965a0 W
0044e5c0 R
00062368 R
```

Portanto, a leitura de cada linha do arquivo poderá ser feita usando o procedimento `da stdio: fscanf(file, "%x %c ", &addr, &rw),` onde o tipo de `addr é unsigned.`

Serão fornecidos quarto arquivos de entrada, que correspondem a sequências de acesso à memoria de uma execução real dos seguintes quarto programas considerados significativos: **compilador.log, matrix.log, compressor.log** e **simulador.log**, ou seja, um compilador, um programa científico, um prompressor de arquivos e um simulador de partículas.

Estes arquivos estão disponíveis em www.inf.puc-rio.br/~endler/courses/inf1019/13.1/P2-arquivos-log, estão compactados (.zip) e devem ser descompactados antes de serem usados como entrada do simuador.

Obtendo a página a partir do endereço lógico

Para de obter o índice da página que corresponde a um endereçp lógico, `addr`, basta descartar os *s* bits menos significativos, ou seja, executar um shift para a direita: **`page= add >> s`**. Por exemplo, se o tamamnho da página/quadro for 4 KB, obtém-se o índice da página com a instrução: **`page= addr>>12`**. No entanto, o seu simulador deve funcionar para diferentes tamanhos de página (vide parâmetro c de **sim-virtual, n**a seção 2), ou seja, voce precisará implementar um código que calcula *s* a partir do valor do parâmetro c).

Estruturas de dados

Como os endereços nos arquivos.log, são de 32 bits então, para páginas de 4 KB (as menores a serem suportadas pelo seu simulador), pode haver 20 bits relativos ao índice de página, permitindo tabelas de página com 1 MB entradas! Isso não seria muito eficiente na prática, mas para fins de simulação, e como está-se considerando apenas um programa em execução, isso não é um problema para a simulação.
A estrutura de dados para representar cada quadro físico deve conter os flags **R** (de página referenciada), flag **M** (de página modificada), e o instante de último acesso de cada página em memória. Os demais campos dependerão de cada um dos algoritmos implementados, e ficam a seu critério.

Implementação da Substituição de páginas

Os algoritmos de substituição de páginas só entram em ação quando todos os quadros de página estiverem ocupados, e houver um Page-fault, ou seja, uma nova página precisar ser carregada na memória física. Nesse caso, a página contida no quadro correspondente será: (a) sobre-escrita, se só tiver acessada para leitura, ou (b) se tiver sido rescrita, precisará ser escrita de volta o disco – na partição de paginação. Tanto os Page-faults como essa escrita de páginas sujas deve ser registrado na simulação.

Utilize um contador **`time`** (valor inicial = 0) e que é incrementado a cada iteração do seu simulador, como forma de simular a passagem do tempo e registrar o momento de cada acesso a memória.

Formato de Saída da simulação

Como resultado, quando o ultimo acesso a memoria for processado, o simulador deverá gerar um pequeno relatório, contendo:

- a configuração utilizada (os quatro parâmetros de **sim-virtual**);
- o número total de acessos à memória contidos no arquivo;
- o número de page faults (páginas carregadas na memória);
- o número de páginas ``sujas'' que tiveram que ser escritas de volta no disco (lembre-se que páginas sujas que existam no final da execução não precisam ser escritas).

Um exemplo de saída poderia ser da forma (valores completamente fictícios):

```
prompt> sim-virtual lru arquivo.log 4 128
Executando o simulador...
Arquivo de entrada: arquivo.log
Tamanho da memoria: 128 KB
Tamanho das páginas: 4 KB
Alg de substituição: lru
Paginas lidas: 520
Paginas escritas: 352
```

Simulador em modo depuração

O programa (simulador) a ser entregue precisar gerar apenas o relatório final descrito anteriormente. Porém, recomenda-se fortemente que o simulador funcione também em um modo ``depuração'' onde o mesmo explicite o que é feito a cada acesso à memória. Para tal, pode-se prever um quinto parâmetro de **sim-virtual**, “debug” ou “-D”, onde D indica o grau de detalhamento do modo depurador. Por exemplo, para “D=1”, o simulador mostraria os bits R e M dos quadros de memoria a cada passo, enquanto que “D=2”, poderia mostrar também o contador “time” do ultimo acesso a cada quadro de memória, etc. Recomendo também que vocês implementem um modo passo-a-passo (modo interativo), para que o usuário possa acompanhar o funcionamento da simulação.

Entregáveis

Vove deve entregar um arquivo .zip ou .tgz contendo o(s) o código fonte do programa (.c e .h), um Makefile e um relatório sobre o seu trabalho. Não inclua os arquivos de teste no processo de entrega!

O relatório deve conter:

- Um resumo do projeto: alguns parágrafos que descrevam a estrutura geral do seu código e todas as estruturas de dados relevantes.
- Uma análise do desempenho dos 3 algoritmos de substituição de páginas para os 4 programas utilizados, comparando o numero de leituras e escritas de página quando o tamanho da memória física permanece fixo, mas o tamanho das páginas varia. Compare para pelo menos quatro tamanhos de página.

Esse relatório com a análise de desempenho é uma parte importante do trabalho e terá um peso significativo sobre a nota do trabalho.

Observações finais

- Os trabalhos devem ser feitos em grupos de dois. Indique claramente os integrantes do grupo no cabeçalho do simulador e no relatorio.
- Os grupos poderão ser chamados para apresentações orais/ demonstrações dos trabalhos entregues. (no dia 25/6)