Atividade Experimental 3

Algoritmos de Substituição de Páginas

1 Descrição Geral

O objetivo desta atividade é estudar e implementar as políticas globais de substituição de páginas, comparando o desempenho de cada uma delas ao aplicar um mesmo *string* de referência. O *string* de referência deve ser gerado de forma aleatória para cada execução do programa. Os algoritmos a serem implementados são os seguintes:

- FIFO
- LRU
- Segunda chance (*clock algorithm*)
- Segunda chance melhorado (*clock algorithm* com bit de modificação)

Observações:

- O programa deve ser implementado segundo as especificações deste documento.
- Essa é uma atividade INDIVIDUAL e OPCIONAL. Lembra-se que para utilizar as notas das atividades experimentais no cálculo da média final dos trabalhos, é necessário entregar TODAS as atividades experimentais dentro do prazo estabelecido.
- As implementações entregues serão corrigidas por um CORRETOR AUTOMÁTICO!
- NÃO PODEM SER UTILIZADAS OUTRAS BIBLIOTECAS, EXCETO AS BIBLIOTECAS PADRÃO DO "C".

2 Algoritmos globais para substituição de página

Os algoritmos globais de substituição de página consideram a existência de uma quantidade fixa e conhecida de quadros na memória física (RAM). O programa a ser desenvolvido nesta atividade considerará a existência de um único processo e as seguintes condições:

- A memória virtual do processo é composta por *P* páginas, numeradas de *0* a *P-1*.
- O *string* de referência (*SR*) representa a sequência de páginas do processo que foram referenciadas ao longo do tempo. Esses valores podem variar entre *0* e *P-1*. No caso do algoritmo "Segunda Chance Melhorado" é necessário diferenciar as referências de leitura e escrita. Para isso, deve-se gerar um número aleatório entre 0 e 15: gerar um número aleatório e calcular o resto da divisão por 16. Então, considerar que a referência será de LEITURA sempre que esse resto for menor ou igual a 12, e a referência será de ESCRITA sempre que esse resto for maior do que 12. Com isso, está sendo considerados que, em média, 80% das referências são de leitura enquanto que 20% são de escrita;
- A memória física (RAM) é composta por *Q* quadros numerados de *0* a *Q-1*. Ela será representada por um vetor *M* com *Q* elementos. O elemento *M[i]* representa o *i*-ésimo quadro de memória e armazenará o identificador (número inteiro) da página cujo conteúdo está residindo nesse quadro, em um dado instante de tempo.

O programa, denominado de *subpag*, deverá ser implementado e executado na máquina virtual GNU/Linux disponível no Moodle da disciplina. O programa será disparado a partir da linha de comando, conforme a seguinte especificação:

```
subpag [opções]
```

As `[opções]` podem aparecer em qualquer ordem e, se não forem fornecidas, deve-se usar seus valores *default*. São as seguintes:

- *-v <quantidade de páginas>*, informa a quantidade de páginas *P* de um processo. Esse parâmetro pode variar entre 1 e 2^{20} (valor *default = 1024*);
- *-s <tamanho do String>*, fornece a quantidade de elementos que comporão o *string* de referência. Esse parâmetro pode variar entre 1 e 2^{20} (valor *default = 5000*);
- *-f <quantidade. de quadros>*, indica a quantidade de quadros *Q* que compõe a memória física (RAM). Esse valor pode variar entre 1 e 2^{20} (valor *default = 256*).

O programa deverá realizar as seguintes tarefas:

1. Inicialmente, o programa deverá gerar a sequência de referências às páginas do processo e coloca-las no vetor *SR[]*. Para isso, o programa deverá usar um gerador de números aleatórios. Notar que o vetor *SR[]* deverá ser dimensionado conforme o valor fornecido na opção "*-s*".
2. Então, para cada elemento do vetor *SR[]*, em ordem, o programa deve verificar se a página referenciada está presente na memória física e ligar, convenientemente, os bits de referência e de modificação da página (*dirty bit*). Se a página estiver na memória, então existirá um elemento do vetor *M[]* com o identificador da página e o algoritmo deve continuar com o próximo elemento de *SR[]*.
3. Se, por outro lado, a página não for encontrada na memória, ocorre uma falta de página. Nesse caso, o algoritmo deve buscar um quadro livre na memória. Se for encontrado um quadro livre, aquela página deverá ser alocada ao quadro livre. Não esquecer de ligar, adequadamente, os bits de referência e de modificação da página.
4. Se não houver quadro livre, o programa deverá selecionar, de acordo com a política do algoritmo de substituição de páginas que estiver rodando, um quadro ocupado e substituir seu conteúdo pelo conteúdo da página que está sendo referenciada. Não esquecer de ligar, adequadamente, os bits de referência e de modificação da página.

Para cada algoritmo de substituição de páginas, o programa deverá aplicar todo o *string* de referência gerado e medir três valores de desempenho. Os conjuntos de valores medidos para os algoritmos devem ser colocados na tela, um conjunto por linha, na seguinte ordem:

- FIFO;
- LRU;
- Segunda Chance;
- Segunda Chance Melhorado.

O conjunto de valores de desempenho de cada algoritmo deve ser colocado na tela em uma única linha, na seguinte ordem:

- Número de operações de *page-in* de páginas;
- Número de operações de *page-out* de páginas;
- Quantidade de faltas de páginas geradas.

3 Dicas de Implementação

Cada algoritmo exigirá estruturas de dados adicionais, que deverão ser manipuladas segundo suas políticas específicas. A seguir são listados alguns aspectos da implementação de cada algoritmo.

FIFO

Deve-se manter uma fila com os identificadores das páginas, ordenada segundo **a ordem de carga** na memória. Com isso, é possível identificar a página que foi carregada a mais tempo na memória;

LRU

Deve-se manter uma fila com os identificadores das páginas, ordenada segundo **o tempo em que foram referenciadas**. Com isso, é possível identifica a página que ficou mais tempo sem ser referenciada. Esse controle de tempo pode ser feito com contadores ou com um mecanismo de fila duplamente encadeada (uma "pilha", na nomenclatura do livro do Silbershatz)

Segunda Chance

Deve-se manter uma fila circular por onde navega o ponteiro de *reset* e de verificação do bit de referência. O critério para escolha da página vítima baseia-se, apenas, no bit de referência;

Segunda Chance Melhorado

O mecanismo é semelhante ao da "Segunda Chance", exceto que agora as páginas podem ser classificadas pelos bits de referência e de modificação, em quatro possibilidades. Agora, a escolha da página vítima baseia-se nos dois bit usados. Então, levando-se em consideração o valor dos bits de referência e de modificação, o algoritmo deve substituir as páginas ou dar-lhes uma segunda chance, conforme a seguinte tabela:

Bit de Referência	Bit de Modificação	Ação
0	0	Substituição imediata.
0	1	Substituir apenas se não houver nenhuma outra

		página para substituição imediata.
1	0	Deve-se dar uma segunda chance, fazendo BR=0.
1	1	Deve-se dar uma segunda chance, fazendo BR=0.

4 Entregáveis

Devem ser entregues:

- Todos os arquivos fonte (arquivos ".c" e ".h") usados para implementar o programa;
- Arquivo *makefile* com pelo menos duas regras: "*all*" (para gerar o programa executável, que deve ter o nome de "*subpag*") e "*clean*" (para apagar todos os arquivos intermediários e o executável.

Os arquivos do programa devem ser compilador com a opção "*-Wall*" e não devem gerar nenhum erro ou *warning* de compilação.

Os arquivos devem ser entregues em um *tar.gz* (NÃO USAR OUTROS FORMATOS), seguindo, **obrigatoriamente**, a seguinte estrutura de diretórios e arquivos:

\subpag		
	include	DIRETÓRIO: local onde devem estar todos os arquivo ".h".
	src	DIRETÓRIO: local onde devem estar todos os arquivos ".c" (códigos fonte) usados na implementação do programa.
	obj	DIRETÓRIO: local onde colocar todos os arquivos do tipo ".o"
	bin	DIRETÓRIO: local onde colocar o arquivo executável, gerado no final do processo de compilação.
	makefile	ARQUIVO: arquivo makefile com as regras para gerar o programa. Deve possuir as regras "*all*" e "*clean*".

5 Avaliação

Para que um trabalho possa ser avaliado ele deverá cumprir com as seguintes condições:

- Entrega dentro dos prazos estabelecidos (INFORMADO NO MOODLE). NÃO serão aceitos programas entregues após a data final de entrega.
- Obediência à especificação da linha de comando. Se não for obedecida essa determinação, o corretor automático indicará erro na implementação;
- Compilação e geração do executável, sem erros ou *warnings*;
- Fornecimento de todos os arquivos solicitados;

O programa será corrigido de forma automática, e sua valoração será proporcional ao número de casos de teste para os quais programa operar corretamente.

6 Observações

Recomenda-se a troca de ideias entre os alunos. Entretanto, a identificação de cópias de trabalhos acarretará na aplicação do Código Disciplinar Discente e a tomada das medidas cabíveis para essa situação.