Simulação de Algoritmos de Substituição de Páginas

Gabriel de Souza Ferreira – 17250447
Jean Marcelo Mira Junior – 16102369

Resumo. *O presente trabalho da disciplina de sistemas operacionais tem por finalidade explorar o gerenciamento de memória através da simulação de algoritmos de substituição de páginas usado pelos mecanismos de paginação dos sistemas operacionais modernos.*

Palavras chave: substituição, páginas, simulador, cpp.

Introdução

Esse trabalho tem como objetivo o desenvolvimento da simulação de um algoritmo de substituição de páginas. Existem várias soluções para algoritmos de substituição de páginas, entretanto este trabalho terá foco no desenvolvimento dos seguintes métodos: First In, First Out (FIFO), Algoritmo Ótimo (OPT) e Least Recently Used (LRU). Cada um deles tem uma estratégia para solucionar a retirada de uma página da memória do sistema operacional quando o mesmo se encontra com falta de página, assim dando espaço para nova página requerida.

Discussão

A seguir será demonstrado a lógica por trás de cada algoritmo de substituição de páginas, apresentando a lógica, desenvolvimento e discussões do trabalho. Vale a pena ressaltar que todo código foi devidamente comentado para o melhor entendimento. A seguir será exemplificado o funcionamento das classes como pode se ver na figura 1.

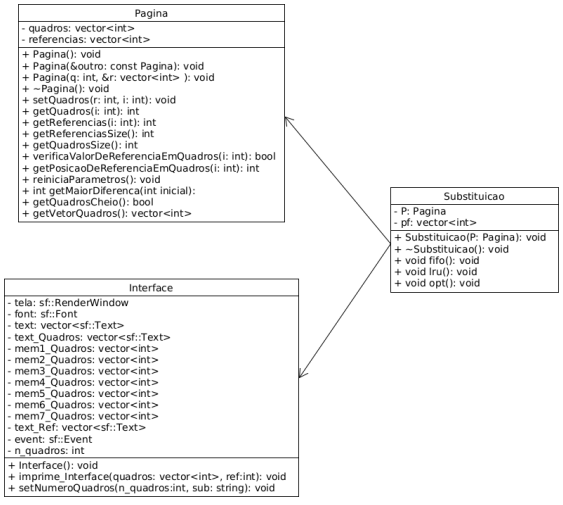

Figura 1: Diagrama UML.

Classe Main - explorando main.cpp

A main.cpp tem como principal função integrar as demais classes, coletando através do usuário via terminal o arquivo que será simulado. O executável já está disponível na pasta do programa sendo necessário que o usuário abra um terminal na pasta onde o mesmo se encontra e digite o comando, por exemplo: ./simulador 3 < exemplo.txt. Sendo o primeiro termo o ponto e barra seguido do nome do executável, depois um espaço seguido da quantidade de quadros da análise, seguido novamente de um espaço e terminando com o nome do arquivo de texto a ser analisado.

Classe Pagina - explorando pagina.h e pagina.cpp:

Esta classe é responsável por armazenar e criar os quadros, além de criar e armazenar o vetor de referências. Sua principal função é guardar os dados que irão ser analisados, contendo os métodos de construtor padrão, construtor por cópia, destrutor padrão, métodos que retornam os valores, métodos que alteram os valores parte padrão da linguagem orientada a objeto. Além disso há também métodos que verificam um valor de referência dentro do vetor de quadros com finalidade de verificar se o valor já está disposto dentro dos quadros, e uma função com a mesma condição operacional com a diferença que está retorna o valor da posição do quadro, entre outras funções que auxiliam o funcionamento do programa que estão devidamente explicadas no código. Podemos ver na figura 2.

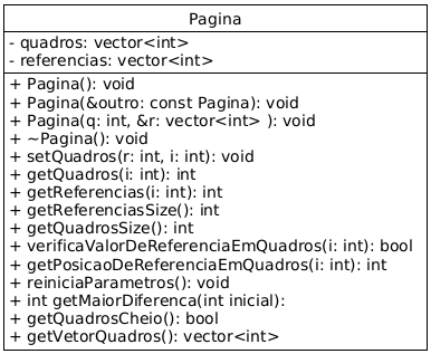

Figura 2: Diagrama UML classe Pagina.

Classe Substituicao - Explorando substituicao.h e substituicao.cpp:

Esta classe tem como finalidade aplicar a simulação dos algoritmos de substituição de página. Contendo uma variável do tipo Pagina e um vetor de inteiros que guarda a quantidade de page fault, além de um construtor padrão e os métodos FIFO, LRU E OPT.

O conceito para desenvolver do algoritmo First In, First Out (FIFO) para o algoritmo de substituição de páginas consiste em criar uma fila do tamanho da quantidade de quadros e adicionar as referências à mesma, ou seja, a fila faz o controle da prioridade de acesso em quadros, sempre adicionando a cabeça da fila e retirando a mesma.

O algoritmo Least Recently Used (LRU) tem o mesmo princípio do FIFO, só que foi utilizado uma lista que quando uma referência já existente nos quadros é adicionada novamente faz com a referência volte para o final da lista, e a referência na antiga posição da lista seja apagada. Por fim o Algoritmo Ótimo (OPT) tem por fundamento a análise da distância das referências que já estão dentro dos quadros, a partir de uma posição do vetor de referências o algoritmo verifica quando o valor dentro do quadro vai ser utilizado novamente e faz a troca baseado no valor de referência que vai demorar mais para ser manipulado. As funções da classe podem ser vistas na figura 3.

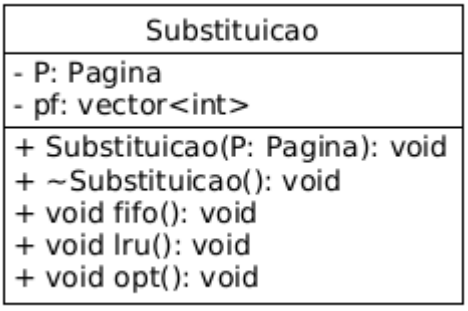

Figura 3: Diagrama UML classe Substituicao.

Classe Interface - Explorando interface.h e interface.cpp:

Esta classe tem como finalidade apresentar o desenvolvimento da mudança de quadro em tempo de execução, sendo que o usuário deve habilitar ou não a funcionalidade no arquivo interface.h. Para habilitar a inteface o usuario deve colocar 1 no `#define INTERFACE 1` e 1 na variável `#define DELAY 1` caso o usuário não queira executar a interface deve colocar o valor zero para as variáveis, como na figura 4. O modelo de interface pode ser visto na figura 6, onde o valor -1 significa que o quadro está vazio.

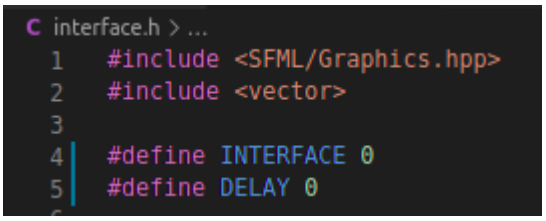

Figura 4: Desabilitar a interface gráfica.

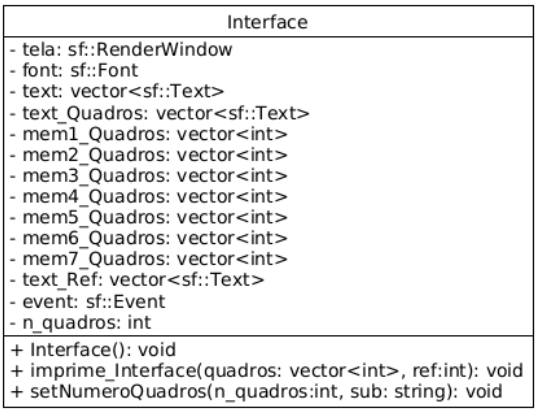

Figura 5: Diagrama UML classe Interface.

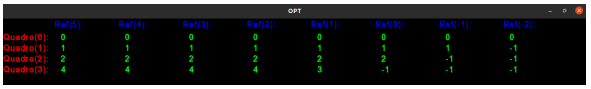

Figura 6: Exemplo de interface gráfica.

Conclusão

Claramente o melhor algoritmo de substituição de páginas dentre First In, First Out (FIFO), Algoritmo Ótimo (OPT) e Least Recently Used (LRU) é o OPT, em segundo lugar temos o LRU e por fim o FIFO. Como esperado o algoritmo ótimo é impossível de ser implementado pois necessita do conhecimento prévio de todas as referências que possam vir a requisitar acesso a memória, assim se tornando um parâmetro para averiguar o quão otimizado são as demais lógicas de substituição. Com base nisso torna o algoritmo Least Recently Used (LRU) o mais eficaz dentre os desenvolvidos neste trabalho.

A principal diferença entre o algoritmo LRU e o FIFO é a prioridade que cada método dá para estabelecer a troca da referência nos quadros. Imagine uma fila, quando uma referência solicita acesso a memória essa referência entra na fila e todas as demais também criando assim uma prioridade de saída da fila, esse é o funcionamento da FIFO onde quando uma referência solicita o acesso a memória continua no mesmo lugar na fila tornando a imutável. Já a fila de prioridade de acesso à memória do método LRU insere o valor da referência no final sempre que

esta solicita acesso a memória, assim mudando a prioridade de saída da memória.

Portanto pode se atestar que dentre as estratégias adotadas em cada algoritmo de substituição de páginas o mais eficaz é o OPT mas impossível de se implementar realmente, o LRU é implementável e relativamente eficaz e por fim o FIFO é relativamente ineficaz mas fácil de implementar. Por fim, a figura 7 mostra os resultados finais de cada exemplo proposto com 4 quadros. Sendo que na figura 8 é mostrado o tempo de execução de cada quantidade de quadros e referências especificadas na imagem, onde a simulação do algoritmo OPT detém o maior tempo de execução.

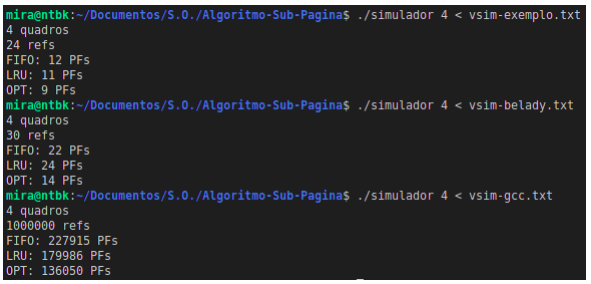

Figura 7: Resultado final com 4 quadros.

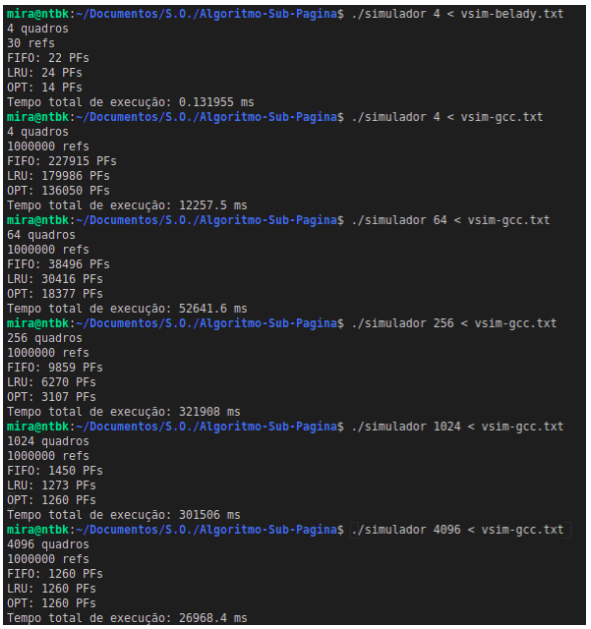

Figura 8: Resultado com tempo de execução.