Universidade do Vale do Itajaí
Disciplina: Sistemas Operacionais
Professor: Felipe Viel
Trabalho M2 – Paginação e Page Fault

Instruções

1. Esta avaliação deve ser feita individualmente ou até em trio.

2. Data de entrega: 28/05/2024 até 19:00. Não serão aceitos trabalhos entregues em atraso.

3. Esta avaliação tem por objetivo consolidar o aprendizado sobre escalonamento.

4. A implementação deverá ser desenvolvida utilizando as bibliotecas e arquivos cedidos. Poderá ser usado o equivalente no Windows e Linux. O uso de bibliotecas de terceiros deverá ser explicado com uma justificativa válida que ficará ao critério do professor aceitar ou não. Você também poderá aumentar a análise com algoritmos e programas próprios.

5. Link para os arquivos: MemoryCost e FDA

6. O sistema deve ser entregue funcionando corretamente.

7. Deve ser apresentado um relatório eletrônico em formato PDF (em outro formato é descontado 1,5 ponto) que contenha:
   a. Identificação do autor e do trabalho.
   b. Enunciado do projeto.
   c. Explicação e contexto da aplicação para compreensão do problema tratado pela solução.
   d. Resultados obtidos com as simulações.
   e. Códigos importantes da implementação.
   f. Resultados obtidos com a implementação (tabelas, gráficos e etc).
   g. Análise e discussão sobre os resultados finais.

8. Deve ser disponibilizado os códigos da implementação juntamente com o relatório (salvo o caso da disponibilidade em repositório aberto do aluno, que deve ser fornecido o link). Também deve ser apresentado o trabalho em aula para o professor. Na apresentação do trabalho, não é necessário utilizar slides, apenas apresentar o código e sua execução (compilar na apresentação).

Descrição do projeto a ser desenvolvido

Projeto 1

A análise de *page faults* em sistemas operacionais refere-se ao processo de monitoramento, identificação e análise das ocorrências de *page faults* em um sistema computacional. Um *page fault* ocorre quando um programa tenta acessar uma página de memória que não está atualmente carregada na memória principal. Esses eventos são de extrema importância para entender o desempenho do sistema operacional e das aplicações em execução, pois podem indicar problemas de otimização de memória, como a necessidade de ajustes na política de paginação ou a presença de vazamentos de memória. A análise de *page faults* envolve a coleta de dados sobre quando e onde esses *page faults* ocorrem, bem como a investigação das causas subjacentes e a implementação de estratégias para otimizar o desempenho do sistema.

Como forma de aplicar conceitos vistos em aula, esse trabalho tem como objetivo avaliar *page faults* em sistemas operacionais de propósito geral, como Windows e Linux. Sendo assim, você deverá:

- Utilizar o código fornecido de teste de alocação de memória para avaliar acesso de paginação de memória. Você deverá, além do padrão oferecido no código original, fornecer diferentes tamanhos de alocação modificando o código fonte.
- Você deverá utilizar o algoritmo do Filtro de Difusão Anisotrópica fornecido para processar imagens em escala de cinza para analisar a demanda de páginas no processamento. São disponibilizadas duas imagens em tamanhos diferentes. Você também pode usar imagens maiores para a análise, porém precisará deixar a imagem em escala de cinza e no formato PGM tipo ASCII (Sugestão: o GIMP faz isso). Uma referência para explicar o FDA está disponível neste ARTIGO. Para executar o algoritmo compilado, você deve passar como argumento o nome da imagem de entrada e da imagem de saída:
    - Exemplo Windows: fda.exe lena_ruido.pgm imagem_out.pgm
    - Exemplo Linux: ./fda lena_ruido.pgm imagem_out.pgm
    - Além disso você deverá informar o parâmetro lambda e quantidade de iterações.
- Você deverá escolher um software, desenvolvido/compilado por você, de sua preferência e multiplataforma (para executar em Windows e Linux) e executar. Não há restrição de linguagem (sim, pode ser em Java).
- Utilizar monitores de desempenho para memória no Windows e Linux para observar o desempenho. Para Windows recomendo o Gerenciador de Processos (está no repositório) e no Linux os comandos (exemplos), mas podem ser utilizados outros:
    - ps -o min_flt, maj_flt <Nº PID>
    - top
- Para quem não utiliza Linux (e até para quem tem), pode utilizar o Codespace ou Cocalc, já que ambas as máquinas têm pouca memória principal, o que ocasionará bastante busca e substituição.
- Você também deverá avaliar a plataforma com poucas aplicações e com vários processos abertos para demandar boa parte da memória (exemplo: 2 ambas do Chrome).

Dica: Adicione evidências visuais no relatório (dos analisadores) e também apresente uma análise em gráfico para cada código com diferentes parâmetros.

Pontuação Extra na prova (máximo 1,0 ponto):

- Utilizar o mesmo algoritmo em duas linguagens diferentes (devem ser 1:1), assumindo que os dois gerem uma alta demanda de memória e compara o desempenho das duas linguagens.