NOVA SCHOOL OF SCIENCE & TECHNOLOGY

MÁTICA

segunda paginação

ORK3

A submeter para avaliação em mooshak.di.fct.unl.pt até final de 9 de novembro de 2023

Este trabalho é individual e todas as soluções serão comparadas. Não olhe para o código de outros alunos e não mostre o seu para outras pessoas.

Objetivo

Conclua um demonstrador de um gerenciador de memória de sistema operacional de brinquedo em uma arquitetura de paginação. Esta demonstração simula apenas a tradução de endereços virtuais (feita pelo hardware MMU) e o tratamento de falhas de página para apenas um processo, feito por um sistema operacional. Iremos ignorar os acessos reais à memória, seu conteúdo, caches de memória e cache de tabela de páginas (TLB), etc. O objetivo principal é entender o tratamento de falhas de página e simular as políticas de carregamento de página sob demanda e substituição de página em um SO.

Programa Simulador

O código-fonte fornecido já implementa a tradução de endereços usando uma tabela de páginas que deve ser atualizada a cada falha de página pelo simulador do SO, após a alocação de um novo quadro de memória física. A leitura e gravação de páginas de memória (page-in e page-out) são omitidas neste simulador.

No código-fonte você pode encontrar a arquitetura de paginação de hardware definida pelo seguinte:

TAMANHO DA PÁGINA–tamanho da página de arquitetura;

MAX_VIRTUAL_PAGES–o tamanho máximo da tabela de páginas. Isso corresponde ao espaço máximo de endereço possível de um processo;

nomeFrames–tamanho da memória física em número de frames (páginas físicas);

tabela de páginas–a tabela da página do processo. O índice desta matriz é o número da página. O SO inicializa-o com todas as páginas 'não presentes' na memória e apenas duas 'páginas inválidas' (não utilizadas pelo processo): a página 0 e a última página do espaço de endereço virtual. Esses casos especiais são representados pelos valores especiais -1 e - 2. Quando uma página é mapeada para a memória real (um quadro é alocado), o número do quadro é salvo aqui.

O SO também mantém a seguinte tabela:

quadroTable–representação do status real da memória. O índice desta matriz representa o número do quadro. O conteúdo de cada posição pode ser -1, representando que este quadro não está em uso, ou o número da página virtual neste quadro (observe que existe apenas um processo).

Durante a simulação, *simularAllSteps* chamadas *traduzirOneAddr* para traduzir cada endereço virtual em um endereço físico, como se fosse feito pelo hardware MMU da CPU. A tradução usa o *tabela de páginas* quando um quadro é encontrado para a página de endereço, mas também pode "gerar exceções", chamando o sistema operacional para lidar com referências de páginas inválidas e falhas de página.

Durante a simulação, algumas estatísticas devem ser coletadas incrementando alguns contadores. Estude todo esse código e seus comentários.

Plano de trabalho

Implemente a paginação sob demanda com uma política de substituição. Para iniciar sua implementação, considere que quando uma referência de memória produz uma falha de página, o SO deve verificar se ela não é válida ou se simplesmente não está em

memória. Se for válido, mas não estiver na memória, aloque um novo quadro vazio para esta página usando uma política sequencial, começando no quadro 0, e atualize o *tabela de páginas* e *quadroTable*. Nesta fase, o simulador pode "executar" programas que cabem completamente na memória real.

Quando toda a memória física é usada (todos os quadros estão em uso), uma política de substituição deve ser usada para remover uma página (possivelmente trocá-la) e usar esse quadro para a página necessária. O *tabela de páginas* e *quadroTable* deve ser atualizado conforme necessário. Implemente uma solução de substituição de página baseada na política FIFO. Nas condições deste simulador, basta começar de novo, sequencialmente, a partir do quadro 0. Leia o capítulo 22.3 do livro OSTEP. Nesta fase, o simulador pode "executar" qualquer programa com qualquer tamanho de memória física.

Para a solução final, altere a política de substituição para dar uma segunda chance às páginas usadas recentemente, semelhante ao algoritmo de clock no OSTEP 22.8. A versão a implementar deverá ser a seguinte:

1. a cada novo acesso à memória, a MMU deverá marcar aquela página como acessada (pode-se usar uma nova tabela de quadros para simular este bit);
2. quando for necessária uma substituição de página, comece de uma posição "atual" procurando um quadro vazio, um quadro em uso mas sem bit acessado e, se todas as últimas escolhas falharam, escolha o quadro "atual" (este quadro não será ser marcado como acessado para este primeiro acesso)
3. A posição "atual" é incrementada para o próximo quadro.

Exemplos e envio

Teste em seu computador. O simulador precisa de dois argumentos: primeiro é o tamanho da memória física dado em número de quadros de memória; segundo, um arquivo de texto com uma sequência de endereços de memória, por linha.

Alguns exemplos de arquivos de acesso à memória são fornecidos e a saída esperada é a seguinte:

para mmu 2 ex1.trace:
swapin página 8 para o quadro 0
swapin página 10 para o quadro 1
swapout página 10 do quadro 1
swapin página 18 para o quadro 1
swapout página 18 do quadro 1
swapin página 20 para o quadro 1
swapout página 20 do quadro 1
swapin página 28 para o quadro 1
Acessos à memória : 9
Falhas de página: 5 (55,56% dos acessos à memória)
Swap outs: 3

para mmu 3 ex1.trace:
swapin página 8 para o quadro 0
swapin página 10 para o quadro 1
swapin página 18 para o quadro 2
swapout página 10 do quadro 1
swapin página 20 para o quadro 1
swapout página 18 do quadro 2
swapin página 28 para o quadro 2
Acessos à memória: 9
Falhas de página: 5 (55,56% dos acessos à memória)
Swap outs: 2

para mmu 3 ex2.trace:
trocar página 8 para o quadro 0
trocar página 10 para o quadro 1
trocar página 18 para o quadro 2
trocar página 18 do quadro 2 trocar
página 20 para o quadro 2 trocar
página 20 do quadro 2 trocar página
18 para o quadro 2

troca da página 18 do quadro 2 troca
da página 20 para o quadro 2 Acessos
à memória: 14
Falhas de página: 6 (42,86% dos acessos à memória)
Swap outs: 3

para mmu 100 ex3.trace:
swapin página 7fff para quadro 0 ERRO: endereço
inválido: 0 (página 0) Processo simulado Falha de
segmentação! Acessos à memória: 3

Falhas de página: 1 (33,33% dos acessos à memória)
Swap outs: 0

para mmu 200 bzip.trace
swapin página 3d92 para o quadro 0
swapin página 114 para o quadro 1
swapin página 89 para o quadro 2
swapin página ff para o quadro 3
swapin página 116 para o quadro 4
swapin página 105 para o quadro 5
swapin página ca para o quadro 6
swapin página 2ff5 para o quadro 7
[. . .]
swapin página 7dec para o quadro 5
swapout página 105 do quadro 7
swapin página 7e90 para o quadro 7
swapout página ca do quadro 8 swapin
página 6f0e para o quadro 8 Acessos à
memória: 1000000
Falhas de página: 311 (0,03% dos acessos à memória)
Trocas: 111

Deverá submeter o ficheiro C com a sua solução final para mooshak.di.fct.unl.pt, como anteriormente. Observe que todas as mensagens informando o que está acontecendo devem aparecer em uma ordem predefinida.

Alguns pontos serão atribuídos a uma solução com substituição de páginas usando ordem FIFO sem Segunda Chance.

Bibliografia

OS Three Ease Pieces, capítulo 22 e capítulo 23 (exemplos de VAX/VMS e Linux)

Apresentações do FOE