Capítulo 3 – Gerenciamento de Memória

- Programas tendem a se expandir afim de ocupar toda a memória disponível.
- Hierarquia de memória: memória dividida em camadas.
- O sistema operacional abstrai as camadas.
- Gerenciador de memória é o responsável por organizar a memória.

3.1 – Sem abstração de memória.

- Considerava-se um sistema apenas com memória física, sem abstração de camadas.
- Não era possível executar dois programas na memória ao mesmo tempo.
- O SO pode estar na parte inferior da RAM, modelo usado antigamente; o SO pode estar em uma ROM, computadores portáteis e embarcados; drivers de dispositivos em ROM, empregado nos primeiros computadores pessoais. A porção que fica na ROM é o BIOS. O primeiro e terceiro modelos tem a desvantagem de o programa de usuário poder alterar o disco.
- Utiliza-se a troca de processos para obter "paralelismo"
- "Realocação" estática ou relocação: Adiciona-se uma constante a todos os endereços de um programa durante o seu carregamento.

3.2 – Abstração de memória: Espaços de endereçamento.

- A exposição da memória física aso processos tem várias desvantagens.

3.2.1 – A noção de espaço de endereçamento.

- Para rodar múltiplas aplicações ao mesmo tempo deve-se resolver 2 problemas: proteção e realocação.
- Um espaço de endereçamento é uma abstração da memória, é um conjunto de endereços que um processo pode utilizar para referenciar a memória.
- Realocação dinâmica: Mapeia cada espaço de endereçamento do processo em uma parte diferente da memória física.
- Equipa-se a CPU com dois registradores: registrador-base e registrador-limite.
- Programas são contínuos na memória.
- Ao acessar um endereço somamos esse endereço ao endereço base no registrador.
- Uma desvantagem é o número de adições necessárias.

3.2.2 – Troca de memória

- Dois métodos para lidar com a sobrecarga de memória: swapping e memória virtual.
- Troca de processos consiste em trazer o processo todo para a memória, executá-lo por alguns instantes e manda-lo para o disco para executar outro processo.
- Memória virtual permite que processos sejam executados mesmo não estando todo na memória principal.
- Quando a troca de processo deixa muitos "buracos" na memória é possível juntá-los todos em um único espaço contiguo, isso se chama compactação de memória. Mas geralmente não é usada pelo tempo necessário.
- Processo que crescem dinamicamente podem apresentar problemas se deixados crescer livremente. Começa-se tentando alocar as redondezas e caso não seja possível faz-se o swapping até não ser mais possível.
- Noção de heap e stakc que crescem um na direção do outro.

3.2.3 – Gerenciando a memória livre

- Dois métodos existem: mapa de bits e listas livres.
- Mapa de bits: divide-se a memória em pequenos blocos e atribui 0 ao bloco livre e 1 o ocupado no mapa.
- Quanto menor a unidade de alocação, maior será o mapa de bits. E o mapa ocupará muita memória, já uma unidade muito pequena pode proporcionar desperdício.
- A desvantagem do mapa de bits é a busca por espaços livres.
- Utiliza-se também listas encadeadas que guardam informações sobre blocos alocados e não alocados (endereço inicial e final).
- Algoritmos de alocação: first fit, mais simples e mais rápido. Next fit, alteração do first fit que memoriza onde foi colocado oultimo

segmento e o tamanho. Best fit, pesquisa a lista inteira e localiza a posição com menos desperdício, é mais lento pois precisa ler a lista inteira, também há mais desperdício de memória pois deixa muitos espaços pequenos na memória. Worst fit procura o espaço que gera mais desperdício.

- Uma solução mais adequada é ter duas listas, uma com segmentos disponíveis e outra com ocupados.
- Existe o quickfit que serve para listas separadas por tamanho.

3.3 – Memória Virtual

- O tamanho da memória aumenta rápido, mas dos software mais rápido ainda.
- Sobreposições.
- Cada programa tem seu próprio espaço de endereçamento dividido em páginas. Parte das páginas estão na memória física. Funciona bem na multiprogramação.

3.3.1 – Paginação

- Endereços gerados pelo programa são endereços virtuais.
- MMU mapeia endereços físicos em endereços virtuais.
- Um cópia completa do programa estará na memória virtual enquanto que só parte dele estará na física.
- Memória virtual = páginas. Memória física = frames.
- A memória desconhece a existência da MMU.
- Existe um bit presente/ausente.
- Se a página não está presente, ocorre uma falta de página. O SO escolhe um frame através de um certo algoritmo e o salva e trás a página faltosa.

3.3.2 – Tabela de páginas

- Matematicamente uma tabela de paginas recebe o endereço virtual e retorna o endereço físico.
- Contem numa tabela de páginas: Número do frame; Bit presente/ausente; proteção; modificada (frames modificados devem ser salvos quando retirados da memória); referenciada (páginas que não estão sendo utilizadas são as melhores candidatas para substituição);
- A memória virtual faz uma nova abstração (o espaço de endereçamento).

3.4 – Algoritmos de substituição de páginas

- Com a necessidade de uma página, uma das paginas deve ser escolhida, se tiver sido modificada deve ser salva, caso contrário pode-se sobrescrever a nova página diretamente sobre esta. O desempenho será melhor se a página escolhida for alguma que não esteja sendo muito usada.

3.4.1- O algoritmo ótimo de substituição de páginas

- Fácil de descrever, mas impossível de implementar.
- A página que será usada por último deve ser retirada.
- Adia a ocorrência da próxima falta de página o máximo possível.
- O SO não tem como saber quando as páginas serão utilizadas.

3.4.2 – O algoritmo de substituição de página Não Usado Recentemente NRU

- Usa os bits de status R e M
- Atualização dos bits é por hardware.
- Separa as páginas em 4 classes de acordo com os valores de R e M.
- 00 – Não referenciada e não modificada
- 01 – Não referenciada e modificada
- 10 – Referenciada e não modificada
- 11 – Referenciada e Modificada
- Remove aleatoriamente (ou em fifo) uma página de classe mais baixa.
- Fácil de entender e implementer, fornece um desempenho adequado.

3.4.3 – O algoritmo de substituição primeiro a entrar, primeiro a sair

- Baixo custo.
- O So mantém uma lista de todas as páginas atualmente na memória, com a página mais antiga na cabeça da lista e a que chegou mais recentemente no final. Quando ocorrer uma falta de página a primeira da fila é removida e a nova é adicionada no final.

- Não diferencia entre páginas mais usadas e menos usadas.
- Raramente usado na configuração original.

3.4.4 – O algoritmo de substituição de página segunda chance

- Usa-se FIFo, mas inspeciona-se o bit R da página antes de retirá-la.
- Se R for 1 coloca-se ele em 0 e manda a página para o fim da fila. Se R for 0, retira-se a página imediatamente.
- Se todas as páginas foram referenciadas recentemente ele se torna o FIFO puro.

3.4.6 – Algoritmo de substituição de página usada menos recentemente (LRU)

- Páginas que não estão sendo utilizadas por um longo período de tempo provavelmente permanecerão inutilizadas por muito tempo.
- Não é barato.
- Armazena-se um contador junto com cada entrada da tabela de página e procura-se o menor valor dele para trocar as páginas.
- Usando matrizes, NxN.A linha da moldura k é setada e a da coluna k é zerada. A linha com menor valor é a LRU.

3.4.10 – Resumo dos algoritmos de substituição de página.

- Ótimo : Não implementável, mas útil como padrão de desempenho.
- NRU: Aproximação mais rudimentar do LRU.
- FIFO: Pode descartar páginas importantes.
- Segunda chance: FIFO melhorado.
- LRU: Ótimo algoritmo, mas difícil de implementar.
- OS melhores são o envelhecimento e o WSClock.

3.5 – Questões de projeto para sistemas de paginação

3.5.1 – Política de alocação local versus global.

- Substituição local e suubstiução global.
- Dividir as molduras igualmente entre os processos.
- AlgoritmoPFF. Controla somenteo tamanho do conjunto de alocação.
- A escolha global vesus local é independente do algoritmo.

3.5.3 – Tamanho de página

- Escolhido pelo SO
- Tamanho de páginas pegueno geram menos fragmentação interna.
- Páginas grande geram desperdício.
- Páginas pequenas implicam em muitas páginas e uma grande tabela de páginas.
- Gasta muito tempo para ler do disco páginas pequenas.
- Tempo de chaveamento (troca de contexto) aumenta com a diminuição do tamanho da pagina. O espaço ocupado pela tabela de páginas também aumenta.

3.5.4 – Espaços separados de instruções e dados

- Não causa nenhuma complicação especial e dublica o espaço de endereçamento disponível.

3.5.8 – Política de limpeza

- Daemon de paginação, dorme na maior parte do tempo, mas as vezes acorda para salvar páginas.
- “Limpa as molduras”

3.6 – Questões de implementação

3.6.1 – Envolvimento do sistema operacional com a paginação

- Na criação de um processo, no tempo de execução do processo, na ocorrência de falta de página e na finalização do processo.
- Quando um processo é criado: SO deve determinar o tamanho do programa e dos dados e criar uma tabela de páginas para ele.
- Área de trocas deve ser inicializada.
- Quando um processo é escalonado: MMu reinicializada e TLB esvaziada para limpar o processo anterior.
- Quando ocorre uma falta de página: o So deve carregar a página para uma moldura.
- Quando um processo termina: deve liberar suas estruturas de dados.

3.6.2 – Tratamento da falta de página

- 1. Hardware gera interrupção. E informações são salvas.
- 2. Rotina de hardware salva registradores e chama o SO.
- 3.SO tenta descobrir qual a pagina faltante.
- 4.SO verifica se existe moldura disponível, se não inicia algoritmo de substituição.
- 5.Se a moldura foi modificada ela será salva no disco e outro processo será executado enquanto isso.
- 6.SO busca a página faltante, enquanto isso o processo permanece suspenso.
- 7.Atualiza-se as tabelas de páginas com novos valores.
- 8. Roda a instrução causadora da falta de pagina novamente.
- 9.O processo causador é escalonado e o SO retorna o controle para ele.
- 10. Registradores são recarregados e funcionamento segue o fluxo normal.

3.6.6 – Separação de política e mecanismo

- Permite um código mais modular e com maior flexibilidade.
- Desvantagem de sobrecarga adicional causada pelos diversos chaveamentos entre núcleo e usuário.

3.7 – Segmentação

- Partes da memória crescem ou diminuem continuamente.
- Utiliza-se segementos.
- Segemento é entidade lógica.
- Segmento cresce bidireccionalmente.
- Cada segmento contém um tipo de objeto.

Paginação X Segmentação

- O programador sabe que a segmentação esta sendo implementada, mas não a paginação.
- Na paginação só há um espaço de endereçamento linear, na segmentação há vários.
- Rotinas e dados podem ser distinguidos e protegidos separadamente na segmentação, mas não na paginação.
- As tabelas cujo tamanho flutuam podem ser facilmente acomodadas na segmentação, mas não na paginação.
- O compartilhamento de rotinas entre outros processos é facilitado na segmentação, mas não na paginação.
- A paginação foi criada para obter um grande espaço de endereçamento linear sem a necessidade de comprar mais memória física.
- A segmentação foi criada para permitir que programas e dados sejam divididos em espaços de endereçamento logicamente independentes e para auxiliar o compartilhamento e proteção.