Sistemas Operacionais

Gerência de Memória Virtual

- Memória virtual é uma técnica sofisticada e poderosa de gerência de memória, onde as memórias principal e secundária são combinadas, dando ao usuário a ilusão de existir uma memória muito maior que a capacidade real da memória principal.

- A primeira implementação foi realizada no início da década de 1960.
- Existe um forte relacionamento entre a gerência da memória virtual e a arquitetura de hardware do sistema computacional.
- Por motivos de desempenho, é comum que algumas funções da gerência de memória virtual sejam implementadas diretamente no hardware.

Espaço de Endereçamento Virtual

- Um programa no ambiente de memória virtual não faz referência a endereços físicos de memória (endereços reais), mas apenas a “endereços virtuais”.
- No momento da execução de uma instrução, o endereço virtual referenciado é traduzido para um endereço físico, pois o processador manipula apenas posições da memória principal.

Espaço de Endereçamento Virtual

- O mecanismo de tradução do endereço virtual para endereço físico é denominado mapeamento.
- Em ambientes que implementam memória virtual, o espaço de endereçamento do processo é conhecido como “espaço de endereçamento virtual” e representa o conjunto de endereços virtuais que o processo pode endereçar.

Espaço de Endereçamento Virtual

- Como o espaço de endereçamento virtual não tem nenhuma relação direta com os endereços no espaço real, um programa pode fazer referência a endereços virtuais que estejam fora dos limites da memória principal, ou seja, os programas e suas estruturas de dados não estão mais limitados ao tamanho da memória física disponível.

Espaço de Endereçamento Virtual

- Quando um programa é executado, somente uma parte do seu código fica residente na memória principal, permanecendo o restante na memória secundária até o momento de ser referenciado. Esta condição permite aumentar o compartilhamento da memória principal entre muitos processos.

Espaço de Endereçamento Virtual

- No desenvolvimento de aplicações, a existência dos endereços virtuais é ignorada pelo programador. Os compiladores e linkers se encarregam de gerar o código executável em função do espaço de endereçamento virtual, e o SO cuida dos detalhes durante sua execução.

Mapeamento

- O processador apenas executa instruções e referencia dados residentes no espaço de endereçamento real; portanto, deve existir um mecanismo que transforme os endereços virtuais em endereços reais.
- Esse mecanismo, conhecido como “mapeamento”, permite traduzir um endereço localizado no espaço virtual para um associado no espaço real.

Mapeamento

- Como consequência do mapeamento, um programa não mais precisa estar necessariamente em endereços contíguos na memória principal para ser executado.
- Nos sistemas modernos, a tarefa de tradução de endereços virtuais é realizada por hardware juntamente com o SO, de forma a não comprometer seu desempenho e torná-lo transparente a usuários e suas aplicações.

Mapeamento

- O dispositivo de hardware responsável por esta tradução é conhecido como unidade de gerência de memória (Memory Management Unit – MMU), sendo acionado sempre que se faz referência a um endereço virtual. Depois de traduzido, o endereço real pode ser utilizado pelo processador para o acesso a memória principal.

Mapeamento

- Cada processo tem o seu próprio espaço de endereçamento virtual como se possuísse sua própria memória. O mecanismo de tradução se encarrega, então, de manter “tabelas de mapeamento” exclusivas para cada processo, relacionando os endereços virtuais do processo às posições na memória real.

Mapeamento

- A tabela de mapeamento é uma estrutura de dados existente para cada processo. Quando um determinado processo está sendo executado, o sistema utiliza a tabela de mapeamento do processo em execução para realizar a tradução de seus endereços virtuais.

Mapeamento

- Se um outro processo vai ser executado, o sistema deve passar a referenciar a tabela de mapeamento do novo processo.
- A troca de tabelas de mapeamento é realizada através de um registrador, que indica a posição inicial da tabela corrente, onde, toda vez que há mudança de contexto, o registrador é atualizado com o endereço da nova tabela.

Memória Virtual por Paginação

- É a técnica de gerência de memória onde o espaço de endereçamento virtual e o espaço de endereçamento real são divididos em blocos de mesmo tamanho chamado “páginas”.
- As páginas no espaço virtual são denominadas “páginas virtuais”, enquanto as páginas no espaço real são chamadas de “páginas reais” ou “frames”.

Memória Virtual por Paginação

- Todo o mapeamento de endereço virtual em real é realizado através de “tabelas de páginas”. Cada processo possui sua própria tabela de páginas e cada página virtual do processo possui uma entrada na tabela, com informações de mapeamento que permitem ao sistema localizar a página real correspondente.

Memória Virtual por Paginação

- Quando um programa é executado, as páginas virtuais são transferidas da memória secundária para a memória principal e colocadas nos frames. Sempre que um programa fizer referência a um endereço virtual, o mecanismo de mapeamento localizará na “entrada na tabela de páginas (ETP)” da tabela do processo o endereço físico do frame no qual se encontra o endereço real correspondente.

Memória Virtual por Paginação

- Nessa técnica, o endereço é formado pelo “número da página virtual (NPV)” e por um “deslocamento”. O NPV identifica unicamente a página virtual que contém o endereço, funcionando como um índice na tabela de páginas.

Memória Virtual por Paginação

- Além da informação sobre a localização da página virtual, a ETP possui outras informações, como o “bit de validade (valid bit)” que indica se uma página está ou não na memória principal. Se o bit tem o valor 0, isto indica que a página virtual não está na memória principal, mas se é igual a 1, a página está localizada na memória.

Memória Virtual por Paginação

- Sempre que o processo referencia um endereço virtual, a unidade de gerência de memória verifica, através do bit de validade, se a página que contém o endereço referenciado está ou não na memória principal. Caso a página não esteja na memória, dizemos que ocorreu um “page fault”.

Memória Virtual por Paginação

- Neste caso, o sistema transfere a página da memória secundária para a memória principal, realizando uma operação de E/S conhecida como “page in” ou “paginação”.
- O número de page faults gerado por um processo depende de como o programa foi desenvolvido, além da política de gerência de memória implementada pelo SO.

Memória Virtual por Paginação

- O número de page faults gerados por cada processo em um determinado intervalo de tempo é definido como “taxa de paginação” do processo. O overhead gerado pelo mecanismo de paginação é inerente da gerência de memória virtual, porém se a taxa de paginação dos processos atingir valores elevados, o excesso de operações de E/S poderá comprometer o desempenho do sistema.

Memória Virtual por Paginação

- Quando um processo referencia um endereço e ocorre um page fault, o processo em questão passa do estado de execução para o estado de espera, até que a página seja transferida do disco para a memória principal. Na troca de contexto, as informações sobre a tabela de mapeamento são salvas e as informações do novo processo escalonado são restauradas. Após a transferência da página para a memória principal, o processo é recolocado na fila de processos no estado de pronto, e quando for reescalonado poderá continuar sua execução.

Políticas de Busca de Páginas

- O mecanismo de memória virtual permite a execução de um programa sem que seu código esteja completamente residente na memória principal.
- A política de busca de páginas determina quando uma página deve ser carregada para a memória.

Políticas de Busca de Páginas

- Basicamente, existem duas estratégias para este propósito:
  - Paginação por demanda
  - Paginação antecipada

Políticas de Busca de Páginas

- **Paginação por demanda**: as páginas dos processos são transferidas da memória secundária para a principal apenas quando são referenciadas. Este mecanismo é conveniente, na medida em que leva para a memória principal apenas as páginas realmente necessárias à execução do programa. Desse modo, é possível que partes não executadas do programa, como rotinas de tratamento de erros, nunca sejam carregadas para a memória.

Políticas de Busca de Páginas

- **Paginação antecipada**: o sistema carrega para a memória principal, além da página referenciada, outras páginas que podem ou não ser necessárias ao processo ao longo do processamento. Se imaginarmos que o programa está armazenado sequencialmente no disco, existe uma grande economia de tempo em levar um conjunto de páginas da memória secundária, ao contrário de carregar uma de cada vez. Por outro lado, caso o processo não precise das páginas carregadas antecipadamente, o sistema terá perdido tempo e ocupado memória principal desnecessariamente.

Políticas de Busca de Páginas

- A técnica de paginação antecipada pode ser empregada no momento da criação de um processo ou na ocorrência de um page fault.

Políticas de Alocação de Páginas

- Determina quantos frames cada processo pode manter na memória principal.
- Existem, basicamente, duas alternativas:
  - Alocação fixa
  - Alocação variável

Políticas de Alocação de Páginas

- **Política de alocação fixa**: cada processo tem um número máximo de frames que pode ser utilizado durante a execução do programa. Caso o número de páginas reais seja insuficiente, uma página do processo deve ser descartada para que uma nova seja carregada. O limite de páginas reais pode ser igual para todos os processos ou definido individualmente.

Políticas de Alocação de Páginas

- Apesar de parecer justo, alocar o mesmo número de páginas para todos os processos, pode não ser uma boa opção, pois a necessidade de memória de cada processo raramente é a mesma. O limite de páginas deve ser definido no momento da criação do processo, com base no tipo da aplicação que será executada. Essa informação faz parte do contexto de software do processo.

Políticas de Alocação de Páginas

- Apesar da simplicidade, a política de alocação fixa de páginas apresenta dois problemas. Se o número máximo de páginas alocadas for muito pequeno, o processo tenderá a ter um elevado número de page faults, o que pode impactar no desempenho de todo o sistema.

Políticas de Alocação de Páginas

- Por outro lado, caso o número de páginas seja muito grande, cada processo irá ocupar na memória principal um espaço maior do que o necessário, reduzindo o número de processos residentes e o grau de multiprogramação. Nesse caso, o sistema pode implementar a técnica de swapping, retirando e carregando processos da/para a memória principal.

Políticas de Alocação de Páginas

- **Política de alocação variável**: o número máximo de páginas alocadas ao processo pode variar durante sua execução em função de sua taxa de paginação e da ocupação da memória principal. Nesse modelo, processos com elevadas taxas de paginação podem ampliar o limite máximo e frames, a fim de reduzir o número de page faults.

Políticas de Alocação de Páginas

- Da mesma forma, processos com baixas taxas de paginação podem ter páginas realocadas para outros processos. Este mecanismo apesar de mais flexível, exige que o SO monitore constantemente o comportamento dos processos, gerando maior overhead.

Políticas de Substituição de Páginas

- Em algumas situações, quando um processo atinge o seu limite de alocação de frames e necessita alocar novas páginas na memória principal, o SO deve selecionar, dentre diversas páginas alocadas, qual deverá ser liberada. Este mecanismo é chamado de “política de substituição de páginas”.

Políticas de Substituição de Páginas

- Uma página real, quando liberada por um processo, está livre para ser utilizada por qualquer outro processo. A partir dessa situação, qualquer estratégia de substituição de páginas deve considerar se uma página foi ou não modificada antes de liberá-la; caso contrário, os dados armazenados na página podem ser perdidos.

Políticas de Substituição de Páginas

- No caso de páginas contendo código executável, que não sofrem alterações, não existe essa preocupação, pois existe uma cópia do código no arquivo executável em disco. As páginas modificáveis, que armazenam variáveis e estruturas de dados, podem sofrer alterações. Neste caso, o sistema deverá gravá-la na memória secundária antes do descarte, preservando seu conteúdo para uso em futuras referências.

Políticas de Substituição de Páginas

- Este mecanismo é conhecido como “page out”. Com este propósito, o sistema mantém um “arquivo de paginação (page file)” onde todas as páginas modificadas e descartadas são armazenadas. Sempre que uma página modificada for novamente referenciada, ocorrerá um “page in”, carregando-a para a memória principal a partir do arquivo de paginação.

Políticas de Substituição de Páginas

- O SO consegue identificar as páginas modificadas através de um bit que existe em cada entrada da tabela de páginas, chamado “bit de modificação (dirty bit ou modify bit)”. Sempre que uma página sofre uma lateração, o valor do bit de modificação é alterado, indicando que a página foi modificada.

Políticas de Substituição de Páginas

- A política de substituição pode ser definida como:
  - Local
  - Global

Políticas de Substituição de Páginas

- **Política de substituição local**: apenas as páginas do processo que gerou o page fault são candidatas a realocação. Nesse modelo, sempre que um processo precisar de uma nova página, o sistema deverá selecionar, dentre os frames alocados pelo processo, a página a ser substituída. Os frames dos demais processos não são avaliados para substituição.

Políticas de Substituição de Páginas

- **Política de substituição global**: todas as páginas alocadas na memória principal são candidatas a substituição, independente do processo que gerou o page fault. Como qualquer processo pode ser escolhido, é possível que o processo selecionado sofra uma aumento na sua taxa de paginação, em função da redução do número de páginas alocadas na memória.

Políticas de Substituição de Páginas

- Na verdade, nem todas as páginas podem ser candidatas a substituição. Algumas páginas, como as do núcleo do sistemas, são marcadas como bloqueadas e não podem ser realocadas.

Políticas de Substituição de Páginas

- A política de alocação fixa permite apenas a utilização de uma política de substituição local.
- A política de alocação variável permite uma política de substituição tanto local quanto global.

Working Set

- Apesar de suas diversas vantagens, o mecanismo de memória virtual introduz um sério problema. Como cada processo possui na memória principal apenas algumas páginas alocadas, o sistema deve manter um conjunto mínimo de frames buscando uma baixa taxa de paginação. Ao mesmo tempo, o SO deve impedir que os processos tenham um número excessivo de páginas na memória, de forma a aumentar o grau de compartilhamento da memória principal.

Working Set

- Caso os processos tenham na memória principal um número insuficiente de páginas para a execução do programa, é provável que diversos frames referenciados ao longo do seu processamento não estejam na memória. Esta situação provoca a ocorrência de um número elevado de page faults e, consequentemente, inúmeras operações de E/S. Neste caso, ocorre um problema conhecido como “thrashing”, provocando sérias consequências ao desempenho do sistema.

Working Set

- O conceito de “working set” surgiu com o objetivo de reduzir o problema do thrashing e está relacionado ao “princípio da localidade”.
- Existem dois tipo de localidade que são observados durante a execução da maioria dos programas.

Working Set

- A “localidade espacial” é a tendência de que após uma referência a uma posição de memória sejam realizadas novas referências a endereços próximos.
- A “localidade temporal” é a tendência de que após a referência a uma posição de memória esta mesma posição seja novamente referenciada em um curto espaço intervalo de tempo.

Working Set

- O princípio da localidade significa, na prática, que o processador tenderá a concentrar suas referências a um conjunto de páginas do processo durante um determinado período de tempo. Imaginando um loop, cujo código ocupe três páginas, a tendência de essas três páginas serem referenciadas diversas vezes é muito alta.

Working Set

- O princípio da localidade é indispensável para que a gerência de memória virtual funcione eficientemente. Como as referências aos endereços de um processo concentram-se em um determinado conjunto de páginas, é possível manter apenas parte do código de cada um dos diversos programas ne memória principal, sem prejudicar a execução dos processos.

Working Set

- Caso contrário, o sistema teria que manter integralmente o código de todos os programas na memória para evitar o problemas de thrashing.

Working Set

- A partir da observação do princípio da localidade, Peter Denning formulou o “modelo de working set”.
- O conceito de working set é definido como sendo o conjunto das páginas referenciadas por um processo durante determinado intervalo de tempo.

Working Set

- O modelo de working set proposto por Denning possibilita prever quais páginas são necessárias à execução de um programa de forma eficiente.
- Caso a janela do working set seja apropriadamente selecionada, em função da localidade do programa, o SO deverá manter as páginas do working set de cada processo residentes na memória principal.

Working Set

- Considerando que a localidade de um programa varia ao longo da sua execução, o tamanho do working set do processo também varia, ou seja, o seu limite de páginas reais deve acompanhar esta variação.
- O working set refletirá a localidade do programa, reduzindo a taxa de paginação dos processos e evitando, consequentemente, o thrashing.

Working Set

- Apesar de o conceito de working set ser bastante intuitivo, sua implementação não é simples, por questão de desempenho. Para implantar esse modelo, o SO deve garantir que o working set de cada processo permaneça na memória principal, determinando quais páginas devem ser mantidas e retiradas em função da última janela de tempo. Em função disso, o modelo de working set deve ser implementado somente em sistemas que utilizam a política de alocação de páginas variável, onde o limite de páginas reais não é fixo.

Algoritmos de Substituição de Páginas

- O maior problema na gerência de memória virtual por paginação não é decidir quais páginas carregar para a memória principal, mas quais liberar.
- Quando um processo necessita de uma nova página e não existem frames disponíveis, o sistema deverá selecionar, dentre as diversas páginas alocadas na memória, qual deverá ser liberada pelo processo.

Algoritmos de Substituição de Páginas

- Os “algoritmos de substituição de páginas” têm o objetivo de selecionar os frames que tenham as menores chances de serem referenciados em um futuro próximo; caso contrário, o frame poderia retornar diversas vezes para a memória principal, gerando vários page faults e acessos à memória secundária.

Algoritmos de Substituição de Páginas

- A partir do princípio da localidade, a maioria dos algoritmos tenta prever o comportamento futuro das aplicações em função do comportamento passado, avaliando o número de vezes que uma página foi referenciada, o momento em que foi carregada para a memória principal e o intervalo de tempo da última referência.

Algoritmos de Substituição de Páginas

- A melhor estratégia de substituição de páginas seria aquela que escolhesse um frame que não fosse mais utilizado no futuro ou levasse mais tempo para ser novamente referenciado. Porém, quanto mais sofisticado o algoritmo de substituição, maior overhead para o SO implementá-lo.

Algoritmos de Substituição de Páginas

- O algoritmo de substituição deve tentar manter o working set dos processos na memória principal e, ao mesmo tempo, não comprometer o desempenho do sistema.

Algoritmos de Substituição de Páginas

- Principais algoritmos existentes para a substituição de páginas:
  - **Ótimo:** o algoritmo ótimo seleciona para substituição uma página que não será mais referenciada no futuro ou aquela que levará o maior intervalo de tempo para ser novamente utilizada. Apesar de este algoritmo garantir as menores taxas de paginação para os processos, na prática é impossível de ser implementado, pois o SO não tem como conhecer o comportamento futuro das aplicações. Essa estratégia é utilizada apenas como modelo comparativo na análise de outros algoritmos de substituição.

Algoritmos de Substituição de Páginas

- **Aleatório:** O algoritmo aleatório, como o nome já sugere, não utiliza critério algum de seleção. Todas as páginas alocadas na memória principal têm a mesma chance de serem selecionadas, inclusive os frames que são frequentemente referenciados. Apesar de ser uma estratégia que consome pouco recursos do sistema, é raramente implementada, em função de sua baixa eficiência.

Algoritmos de Substituição de Páginas

- **FIFO (First-In-First-Out):** No algoritmo FIFO, a página que primeiro foi utilizada será a primeira a ser escolhida, ou seja, o algoritmo seleciona a página que está a mais tempo na memória principal. O algoritmo pode ser implementado associando-se a cada página o momento em que foi carregada para a memória ou utilizando-se uma estrutura de fila, onde as páginas mais antigas estão no início e as mais recentes no final.

Algoritmos de Substituição de Páginas

– Parece razoável pensar que uma página que esteja há mais tempo na memória seja justamente aquela que deve ser selecionada, porém isto nem sempre é verdadeiro. No caso de uma página que seja constantemente referenciada, como é o caso de páginas que contém dados, o fator tempo torna-se irrelevante e o sistema tem que referenciar a mesma página diversas vezes ao longo do processamento.

Algoritmos de Substituição de Páginas

- O algoritmo FIFO é raramente implementado sem algum outro mecanismo que minimize o problema da seleção de páginas antigas que são constantemente referenciadas.

Algoritmos de Substituição de Páginas

- **LFU (Least-Frequently-Used):** O algoritmo LFU seleciona a página menos referenciada, ou seja, o frame menos utilizado. Para isso, é mantido um contador com o número de referências para cada página na memória principal. A página que possuir o contador com o menor número de referências será escolhida, ou seja, o algoritmo evita selecionar páginas que são bastante utilizadas.

Algoritmos de Substituição de Páginas

– Inicialmente, esta parece ser uma boa estratégia, porém as páginas que estão há pouco tempo na memória principal podem ser, justamente, aquelas selecionadas, pois seus contadores estarão com o menor número de referências. É possível também que uma página muito utilizada no passado não seja mais referenciada no futuro.

Algoritmos de Substituição de Páginas

- Nesse caso, como o contador possuiria um número elevado de referências, a página não seria selecionada para substituição. Este esquema, como apresentado, é raramente implementado, servindo apenas de base para outros algoritmos de substituição.

Algoritmos de Substituição de Páginas

- **LRU (Least-Recently-Used):** O algoritmo LRU seleciona a página na memória principal que está há mais tempo sem ser referenciada. Se considerarmos o princípio da localidade, uma página que não foi utilizada recentemente provavelmente não será referenciada novamente em um futuro próximo.

Algoritmos de Substituição de Páginas

- Para implementar esse algoritmo, é necessário que cada página tenha associado o momento do último acesso, que deve ser atualizado a cada referência a um frame. Quando for necessário substituir a página, o sistema fará uma busca por um frame que esteja há mais tempo sem ser referenciado. Outra maneira de implementar o LRU seria através de uma lista encadeada, onde todas as páginas estariam ordenadas pelo momento da última referencia. Nesta caso, cada acesso à memória exigiria um acesso à lista.

Algoritmos de Substituição de Páginas

– Apesar de ser uma estratégia com uma eficiência comparável ao algoritmo ótimo, é pouco empregada na prática, devido ao seu elevado custo de implementação.

Algoritmos de Substituição de Páginas

- **NRU (Not-Recently-Used):** O algoritmo NRU é bastante semelhante ao LRU, porém com menor sofisticação. Para a implementação deste algoritmo é necessário um bit adicional, conhecido como “bit de referência (BR)”. O bit indica se a página foi utilizada recentemente e está presente em cada entrada da tabela de páginas.

Algoritmos de Substituição de Páginas

- Quando uma página é carregada para a memória principal, o bit de referência é alterado pelo hardware, indicando que a página foi referenciada (BR=1). Periodicamente, o sistema altera o valor do bit de referência (BR=0), e a medida que as páginas são utilizadas, o bit associado a cada frame retorna para 1. Dessa forma, é possível distinguir quais frames foram recentemente referenciados. No momento da substituição de uma página, o sistema seleciona um dos frames que não tenha ido utilizado recentemente, ou seja, com o bit de referência igual a zero.

Algoritmos de Substituição de Páginas

– O algoritmo NRU torna-se mais eficiente se o bit de modificação for utilizado em conjunto com o bit de referência. Neste caso, é possível classificar as páginas em quatro categorias.

Algoritmos de Substituição de Páginas

Categorias	Bits avaliados	Resultado
1	BR=0 e BM=0	Página não referenciada e não modificada
2	BR=0 e BM=1	Página não referenciada e modificada
3	BR=1 e BM=0	Página referenciada e não modificada
4	BR=1 e BM=1	Página referenciada e modificada

Algoritmos de Substituição de Páginas

- **FIFO com buffer de páginas:** combina uma lista de páginas alocadas (LPA) com uma lista de páginas livres (LPL). A LPA organiza todas as páginas que estão sendo utilizadas na memória principal, podendo ser implementada como uma lista única para todos os processos ou uma lista individual para cada processo. Independentemente da política utilizada, a LPA organiza as páginas alocadas há mais tempo na memória no início da lista, enquanto as páginas mais recentes no seu final.

Algoritmos de Substituição de Páginas

- Da mesma forma, a LPL organiza todos os frames livres da memória principal, sendo que as páginas livres a mais tempo estão no início e as mais recentes no final. Sempre que um processo necessita alocar uma nova página, o sistema utiliza a primeira página da LPL, colocando-a no final da LPA. Caso o processo tenha que liberar uma página, o mecanismo de substituição seleciona o frame em uso há mais tempo na memória, isto é, o primeiro LPA, colocando-o no final da LPL.

Algoritmos de Substituição de Páginas

- É importante notar que a página selecionada e que entrou na LPL continua disponível na memória principal por um determinado intervalo de tempo. Caso esta página seja novamente referenciada e ainda não tenhasido alocada, basta retirá-la da LPL e devolvê-la ao processo. Nesse caso, a LPL funciona como um buffer de páginas, evitando o acesso à memória secundária. Por outro lado, se a página não for mais referenciada, como o passar do tempo irá chegar ao início da LPL, quando será utilizada para outro processo. Caso a página seja posteriormente referenciada, o sistema terá que carregá-la novamente da memória secundária.

Algoritmos de Substituição de Páginas

- O buffer de páginas permite criar um algoritmo de substituição de páginas simples e eficiente, sem o custo de outras implementações. O SO Mach utiliza o esuqema de buffer de páginas único, enquanto o OpenVMS e o Windows 2000 utilizam dois buffers: uma para páginas livres (free page list) e outro para páginas modificadas (modified page list).

Algoritmos de Substituição de Páginas

- **FIFO circular (clock):** utiliza como base o FIFO, porém as páginas alocadas na memória estão em uma estrutura de lista circular, semelhante a um relógio. Este algoritmo é implementado, com pequenas variações na maioria dos sistemas Unix.
- Para a implementação do algoritmo existe um ponteiro que guarda a posição da página mais antiga na lista.

Algoritmos de Substituição de Páginas

- Cada página possui associado um bit de referência, indicando se a página foi recentemente referenciada. Quando é necessário substituir uma página, o sistema verifica se o frame apontado tem o bit de referência desligado (BR=0). Nesse caso, a página é selecionada para descarte, pois, além de ser a mais antiga, não foi utilizada recentemente. Por outro lado, se a página apontada tem o bit de referência ligado (BR=1), o bit é desligado e o ponteiro incrementado, pois, apesar de ser a página mais antiga, foi utilizada recentemente. O processo se repete até ser encontrado uma página com bit de referência igual a zero.

Algoritmos de Substituição de Páginas

- Neste algoritmo existe a possibilidade de todos os frames possuírem o bit de referência ligado. Nesse caso, o ponteiro percorrerá toda a lista, desligando o bit de referência de cada página. Ao final, a página mais antiga é selecionada, A utilização do bit de referência permite conceder a cada página uma segunda chance antes de ser substituída.

Algoritmos de Substituição de Páginas

- É possível melhorar a eficiência do algoritmo utilizando o bit de modificãção, juntamente com o bit de referência, como apresentado no esquema NRU.

Tamanho de Página

- A definição do “tamanho de página” é um fator importante no projeto de sistemas que implementam memória virtual por paginação.
- O tamanho da página está associado à arquitetura do hardware e varia de acordo com o processador, mas normalmente está entre 512 e 16 M endereços. Algumas arquiteturas permitem a configuração do tamanho da página, oferecendo assim maior flexibilidade.

Tamanho de Página

- O tamanho da página tem impacto direto sobre o número de entradas na tabela de páginas e, consequentemente, no tamanho da tabela e no espaço ocupado na memória principal.
- Por exemplo, em uma arquitetura de 32 bits para endereçamento e páginas de 4 K endereços, teríamos tabelas de páginas de até 2^{20} entradas. Se cada entrada ocupasse 4 bytes, poderíamos ter tabelas de páginas de 4 MB por processo.

Tamanho de Página

- Logo, páginas pequenas necessitam de tabelas de mapeamento maiores, provocam maior taxa de paginação e aumentam o número de acessos à memória secundária.
- Apesar de páginas grandes tornarem menor o tamanho das tabelas de páginas, ocorre o problema da “fragmentação interna”. A fragmentação só é encontrada, realmente, na última página, quando o código não ocupa o frame por completo.

Tamanho de Página

- O principal argumento a favor do uso de páginas pequenas é a melhor utilização da memória principal. A partir do princípio da localidade, com páginas pequenas teríamos na memória apenas as partes dos programas com maiores chances de serem executadas. Quanto maior o tamanho de página, maiores são as chances de ter na memória código pouco referenciado, ocupando espaço desnecessariamente. Além disso, páginas pequenas reduzem o problema da fragmentação interna.

Tamanho de Página

- Outra preocupação quanto ao tamanho da página é a relacionada aos tempos de leitura e gravação na memória secundária. Devido ao modo de funcionamento dos discos, o tempo de operações de E/S com duas páginas de 512 bytes e muito maior do que uma página de 1024 bytes.

Tamanho de Página

- Com o aumento do espaço de endereçamento e da velocidade de acesso à memória principal, a tendência no projeto de SO com memória virtual por paginação é a adoção de páginas maiores, apesar dos problemas citados.

Paginação em Múltiplos Níveis

- Em sistemas que implementam apenas um nível de paginação, o tamanho das tabelas de páginas pode ser um problema. Em uma arquitetura de 32 bits para endereçamento e páginas com 4 K endereços por processo, onde cada entrada na tabela de páginas ocupe 4 bytes, a tabela de páginas poderia ter mais de um milhão de entradas e ocuparia 4 MB de espaço.

Paginação em Múltiplos Níveis

- Imaginando vários processos residentes na memória principal, manter tabelas desse tamanho para cada processo certamente seria de difícil gerenciamento.

Paginação em Múltiplos Níveis

- Uma boa solução para contornar o problema apresentado é a utilização de “tabelas de páginas em múltiplos níveis”. A idéia é que o princípio da localidade seja aplicado também às tabelas de mapeamento, apenas as informações sobre páginas realmente necessárias aos processos estariam residentes na memória principal.

Translation Lookaside Buffer

- A gerência de memória virtual utiliza a técnica de mapeamento para traduzir endereços virtuais em endereços reais, porém, o mapeamento implica em pelo menos dois acessos à memória principal: o primeiro à tabela de páginas e o outro à própria página. Sempre que um endereço virtual precisa ser traduzido, a tabela de mapeamento deve ser consultada para se obter o endereço do frame e, posteriormente, acessar o dado na memória principal.

Translation Lookaside Buffer

- Como a maioria das aplicações referencia um número reduzido de frames na memória principal, seguindo o princípio da localidade, somente uma pequena fração da tabela de mapeamento é realmente necessária. Com base neste princípio, foi introduzida uma memória especial chamada “translation lookside buffer (TLB)”, com o intuito de mapear endereços virtuais em endereços físicos sem a necessidade do acesso à tabela de páginas.

Translation Lookaside Buffer

- O TLB funciona como uma memória cache, mantendo apenas as traduções dos endereços virtuais das páginas mais recentemente referenciadas.
- A TLB é essencial para reduzir o número de operações de acesso à memória principal em sistemas que implementam memória virtual.

Proteção de Memória

- Em qualquer sistema multiprogramável, onde diversas aplicações compartilham a memória principal, devem existir mecanismos para preservar as áreas de memória do SO e dos diversos processos dos usuários. O SO deve impedir que um processo tenha acesso ou modifique uma página do sistema sem autorização.

Proteção de Memória

- Caso uma página do SO seja indevidamente alterada, é possível que, como consequência, haja uma instabilidade no funcionamento do sistema ou sua parada completa. Até mesmo páginas dos processos de usuários necessitam ser protegidas contra alteração, como no caso de frames contendo código executável.

Proteção de Memória

- A proteção de acesso é realizada individualmente em cada página da memória principal, utilizando-se as entradas da tabela de mapeamento, onde alguns bits especificam os acessos permitidos.

Proteção de Memória

- Sempre que uma página é referenciada, o SO verifica na tabela de mapeamento do processo a proteção do frame e determina se a operação é permitida. Caso a tentativa de uma operação de gravação seja realizada em uma página com proteção apenas de leitura, o sistema gera um erro indicando a ocorrência do problema.

Compartilhamento de Memória

- Em sistemas que implementam memória virtual, é bastante simples a implementação da reentrância, possibilitando compartilhamento de código entre os diversos processos. Para isso, basta que as entradas das tabelas de mapeamento dos processos apontem para os mesmos frames na memória principal, evitando, assim, várias cópias de um mesmo programa. Apesar de os processos compartilharem as mesmas páginas de código, cada um possui sua própria área de dados em páginas independentes.

Compartilhamento de Memória

- O compartilhamento de memória também é extremamente importante em aplicações que precisam compartilhar dados na memória principal. Similar ao compartilhamento de código, o mecanismo de paginação permite que processos façam o mapeamento de uma mesma área na memória e, consequentemente, tenham acesso compartilhado de leitura e gravação. A única preocupação da aplicação é garantir o sincronismo no acesso à região compartilhada, evitando problemas de inconsistência.

Memória Virtual por Segmentação

- É a técnica de gerência de memória onde o espaço de endereçamento virtual é dividido em blocos de tamanhos diferentes chamados “segmentos”. Na técnica de segmentação, um programa é dividido logicamente em sub-rotinas e estruturas de dados que são alocadas em segmentos na memória principal.

Memória Virtual por Segmentação

- Enquanto na técnica de paginação o programa é dividido em páginas de tamanho fixo, sem qualquer ligação com a sua estrutura, na segmentação existe uma relação entre a lógica do programa e sua alocação na memória principal. Normalmente, a definição dos segmentos é realizada pelo compilador, a partir do código fonte do programa, e cada segmento pode representar um procedimento, função, vetor ou pilha.

Memória Virtual por Segmentação

- O mecanismo de mapeamento é muito semelhante ao de paginação. Os segmentos são mapeados através de “tabelas de mapeamento de segmentos”, e os endereços são compostos pelo “número do segmento virtual” e por um “deslocamento”.

Memória Virtual por Segmentação

- Uma grande vantagem da segmentação em relação à paginação é a sua facilidade em lidar com estruturas de dados dinâmicas.
- Na técnica de segmentação, apenas os segmentos referenciados são transferidos da memória secundária para a memória principal. Se as aplicações não forem desenvolvidas em módulos, grandes segmentos estarão na memória desnecessariamente, reduzindo o compartilhamento de memória e o grau de multiprogramação.

Memória Virtual por Segmentação

- Logo, para que a segmentação funcione de forma eficiente, os programas devem estar bem modularizados.
- Enquanto na paginação existe o problema da fragmentação interna, na segmentação externa surge o problema da fragmentação externa.

Memória Virtual por Segmentação

- Este problema ocorre sempre que há diversas áreas livres na memória principal, mas nenhuma é grande o suficiente para alocar um novo segmento. Neste caso, é necessário que os segmentos sejam realocados na memória de forma que os espaços livres sejam agrupados em uma única área maior.

Memória Virtual por Segmentação

- Em sistemas com segmentação, a proteção de memória é mais simples de ser implementada do que em sistemas com paginação.
- Na segmentação é mais simples o compartilhamento de memória do que na paginação, pois a tabela de segmentos mapeia estruturas lógicas e não páginas.

Memória Virtual por Segmentação com Paginação

- É a técnica de gerência de memória onde o espaço de endereçamento é dividido em segmentos e, por sua vez, cada segmento dividido em páginas. Esse esquema de gerência de memória tem o objetivo de oferecer as vantagens tanto da técnica de paginação quanto da técnica de segmentação.

Memória Virtual por Segmentação com Paginação

- Na visão do programador, sua aplicação continua sendo mapeada em segmentos de tamanhos diferentes, em função das sub-rotinas e estruturas de dados definidas no programa. Por outro lado, o sistema trata cada segmento como um conjunto de páginas de mesmo tamanho, mapeadas por uma tabela de páginas associadas ao segmento. Dessa forma, um segmento não precisa estar contíguo na memória principal, eliminando o problema da fragmentação externa encontrado na segmentação pura.

Swapping em Memória Virtual

- A técnica de swapping também pode ser aplicada em sistemas com memória virtual, permitindo aumentar o número de processos que compartilham a memória principal e, consequentemente, o grau de multiprogramação do sistema.

Swapping em Memória Virtual

- Quando existem novos processos para serem executados e não há memória principal livre suficiente para alocação, o sistema utiliza o swapping, selecionando um ou mais processos para saírem da memória e oferecer espaço para novos processos. Depois de escolhidos, o sistema retira os processos da memória principal para uma memória secundária (swap out), onde as páginas ou segmentos são gravados em um “arquivo de swap (swap file)”.

Swapping em Memória Virtual

- Com os processos salvos na memória secundária, os frames ou segmentos alocados são liberados para novos processos. Posteriormente, os processos que foram retirados da memória devem retornar para a memória principal (swap in) para serem novamente executados.

Swapping em Memória Virtual

- O arquivo de swap é compartilhado por todos os processos que estão sendo executados no ambiente. Quando o processo é criado, o sistema reserva um espaço no arquivo de swap para o processo. Da mesma forma, quando um processo é eliminado o sistema libera a área alocada.

Swapping em Memória Virtual

- Em alguns SO, o arquivo de swap é, na verdade, uma área no disco reservada exclusivamente para esta função. Independentemente da implementação, o arquivo de swap deve oferecer o melhor desempenho possível para as operações de swapping.

Exercício

- Suponha que o esquema adotado para gerenciamento de memória de um determinado computador seja baseado na estratégia de working sets - W(t,), com política de re-alocação de página do tipo LRU – Least Recently Used e = 3. Nessas condições, se um determinado processo apresentar a seguinte seqüência de referências a páginas virtuais: 24, 15, 18, 23, 24, 18, 17, 18, 24, 17, 17, 15 o número de ocorrências de page faults será de:

a) 11
b) 9
c) 5
d) 13
e) 7

Exercício

- Suponha que o esquema adotado para gerenciamento de memória de um determinado computador seja baseado na estratégia de working sets - W(t,Δ), com política de re-alocação de página do tipo LRU – Least Recently Used e Δ = 3. Nessas condições, se um determinado processo apresentar a seguinte seqüência de referências a páginas virtuais: 24, 15, 18, 23, 24, 18, 17, 18, 24, 17, 17, 15 o número de ocorrências de page faults será de:

a) 11
b) 9
c) 5
d) 13
e) 7

	24	15	18	23	24	18	17	18	24	17	17	15
1	24	24	24	23	23	23	17	17	17	17	17	17
2		15	15	15	24	24	24	24	24	24	24	24
3			18	18	18	18	18	18	18	18	18	15
PF	1	1	1	1	1		1					1

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12
	24	15	18	23	24	18	17	18	24	17	17	15
1												
2												
3												
PF												

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12
	24	15	18	23	24	18	17	18	24	17	17	15
1	24											
2												
3												
PF	1											

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12
	24	15	18	23	24	18	17	18	24	17	17	15
1	24	24										
2		15										
3												
PF	1	1										

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12
	24	15	18	23	24	18	17	18	24	17	17	15
1	24	24	24									
2		15	15									
3			18									
PF	1	1	1									

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12
	24	15	18	23	24	18	17	18	24	17	17	15
1	24	24	24	23								
2		15	15	15								
3			18	18								
PF	1	1	1	1								

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12
	24	15	18	23	24	18	17	18	24	17	17	15
1	24	24	24	23	23							
2		15	15	15	24							
3			18	18	18							
PF	1	1	1	1	1							

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12
	24	15	18	23	24	18	17	18	24	17	17	15
1	24	24	24	23	23	23						
2		15	15	15	24	24						
3			18	18	18	18						
PF	1	1	1	1	1							

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12
	24	15	18	23	24	18	17	18	24	17	17	15
1	24	24	24	23	23	23	17					
2		15	15	15	24	24	24					
3			18	18	18	18	18					
PF	1	1	1	1	1		1					

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12
	24	15	18	23	24	18	17	18	24	17	17	15
1	24	24	24	23	23	23	17	17				
2		15	15	15	24	24	24	24				
3			18	18	18	18	18	18				
PF	1	1	1	1	1		1					

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12
	24	15	18	23	24	18	17	18	24	17	17	15
1	24	24	24	23	23	23	17	17	17			
2		15	15	15	24	24	24	24	24			
3			18	18	18	18	18	18	18			
PF	1	1	1	1	1		1					

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12
	24	15	18	23	24	18	17	18	24	17	17	15
1	24	24	24	23	23	23	17	17	17	17		
2		15	15	15	24	24	24	24	24	24		
3			18	18	18	18	18	18	18	18		
PF	1	1	1	1	1		1					

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12
	24	15	18	23	24	18	17	18	24	17	17	15
1	24	24	24	23	23	23	17	17	17	17	17	
2		15	15	15	24	24	24	24	24	24	24	
3			18	18	18	18	18	18	18	18	18	
PF	1	1	1	1	1		1					

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12
	24	15	18	23	24	18	17	18	24	17	17	15
1	24	24	24	23	23	23	17	17	17	17	17	17
2		15	15	15	24	24	24	24	24	24	24	24
3			18	18	18	18	18	18	18	18	18	15
PF	1	1	1	1	1		1					1

Exercício

- Suponha que o esquema adotado para gerenciamento de memória de um determinado computador seja baseado na estratégia de working sets - W(t,), com política de re-alocação de página do tipo LRU – Least Recently Used e = 3. Nessas condições, se um determinado processo apresentar a seguinte seqüência de referências a páginas virtuais: 24, 15, 18, 23, 24, 18, 17, 18, 24, 17, 17, 15 o número de ocorrências de page faults será de:

a) 11
b) 9
c) 5
d) 13
e) 7

	24	15	18	23	24	18	17	18	24	17	17	15
1	24	24	24	23	23	23	17	17	17	17	17	17
2		15	15	15	24	24	24	24	24	24	24	24
3			18	18	18	18	18	18	18	18	18	15
PF	1	1	1	1	1		1					1