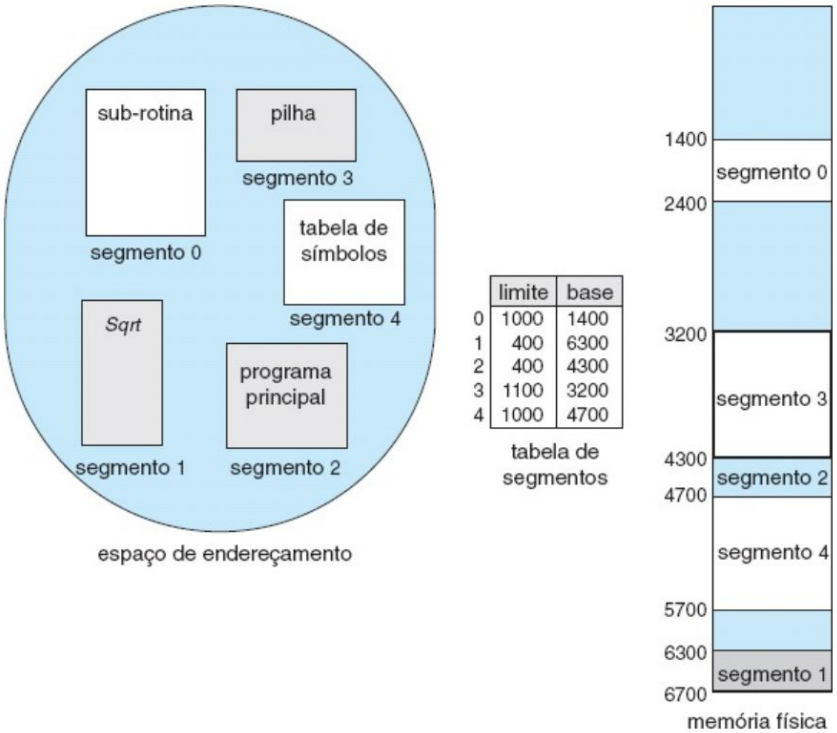
Sistemas Operacionais I

Aula passada

- Segmentação;
- Paginação;

Segmentação

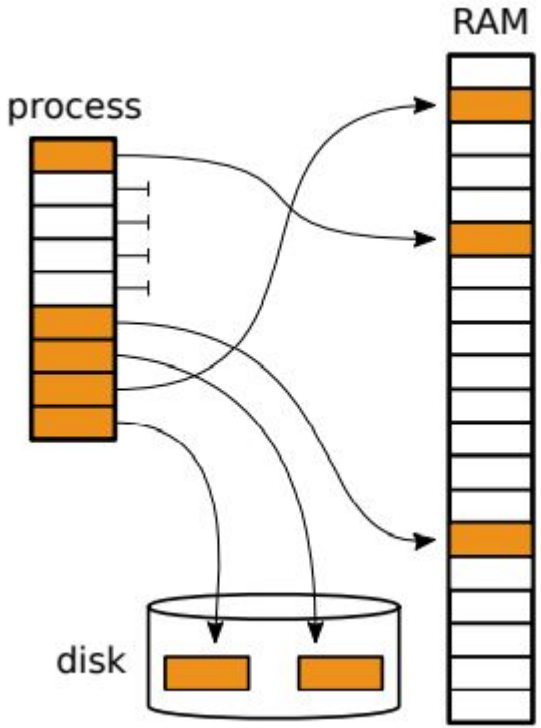


Exemplo de segmentação.

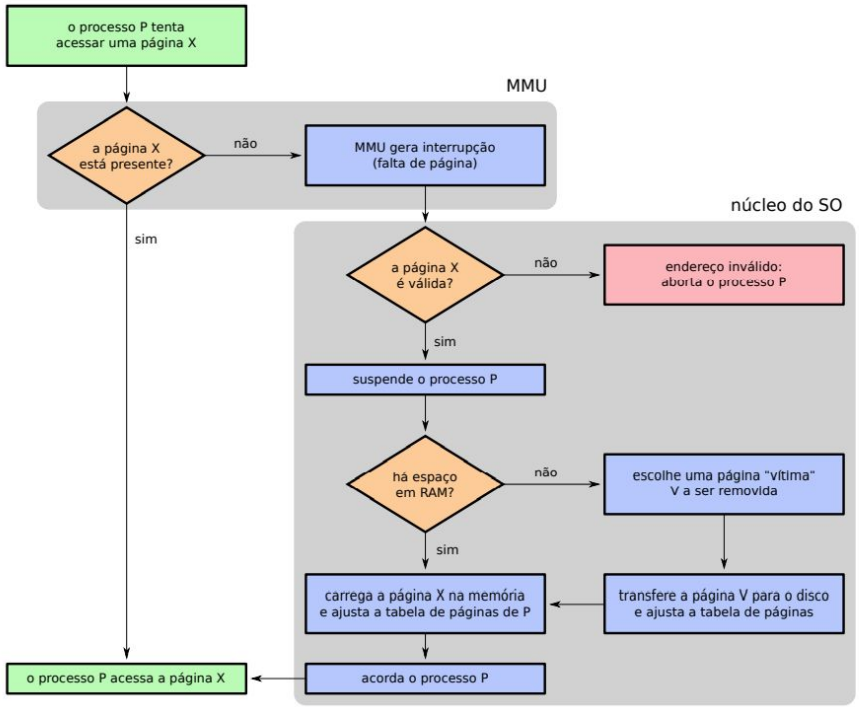
Paginação

- ➔ Usar dispositivos de **armazenamento secundário** como extensão da memória RAM;
- ➔ Partes **ociosas** da memória podem ser **transferidas** para um **disco**, liberando a memória RAM para outros usos;
- ➔ Caso algum **processo** tente **acessar** esse **conteúdo** posteriormente, ele deverá ser trazido de **volta** à **memória RAM**;
- ➔ A **transferência** dos **dados** entre **memória** e **disco** é feita pelo **sistema operacional**, de forma **transparente** para os **processos**;

A ideia central da paginação em disco consiste em transferir páginas ociosas da memória RAM para uma área em disco, liberando memória para outras páginas.



Suporte de Hardware à Paginação



Ações do mecanismo de **paginação** em disco

Critérios de Seleção

Idade da página: Há quanto tempo a página está na memória; páginas muito antigas talvez sejam pouco usadas.

Frequência de acessos à página: páginas mais acessadas em um passado recente possivelmente ainda o serão em um futuro próximo.

Data do último acesso: páginas há mais tempo sem acessar possivelmente serão pouco acessadas em um futuro próximo (sobretudo se os processos respeitarem o princípio da localidade de referências).

Prioridade do processo proprietário: processos de alta prioridade, ou de tempo real, podem precisar de suas páginas de memória rapidamente; se elas estiverem no disco, seu desempenho ou tempo de resposta poderão ser prejudicados.

Conteúdo da página: páginas cujo conteúdo seja código executável exigem menos esforço do mecanismo de paginação, porque seu conteúdo já está mapeado no disco (dentro do arquivo executável correspondente ao processo). Por outro lado, páginas de dados que tenham sido alteradas precisam ser salvas na área de troca.

Páginas especiais: páginas contendo buffers de operações de entrada/saída podem ocasionar dificuldades ao núcleo caso não estejam na memória no momento em que ocorrer a transferência de dados entre o processo e o dispositivo físico. O processo também pode solicitar que certas páginas contendo informações sensíveis (como senhas ou chaves criptográficas) não sejam copiadas na área de troca, por segurança.

Agenda

- Memória Virtual (Fundamentos)
  - Paginação por Demanda;
  - Substituição de Páginas;

Memória Virtual

A **memória virtual** é uma técnica que permite a execução de processos que **não estão** totalmente na memória.

Uma grande vantagem desse esquema é que os programas podem ser maiores do que a memória física.

A memória virtual abstrai a memória principal em um array de armazenamento uniforme extremamente grande, separando a memória lógica, conforme vista pelo usuário, da memória física.

Essa técnica deixa os programadores livres de preocupações com as **limitações de armazenamento da memória**.

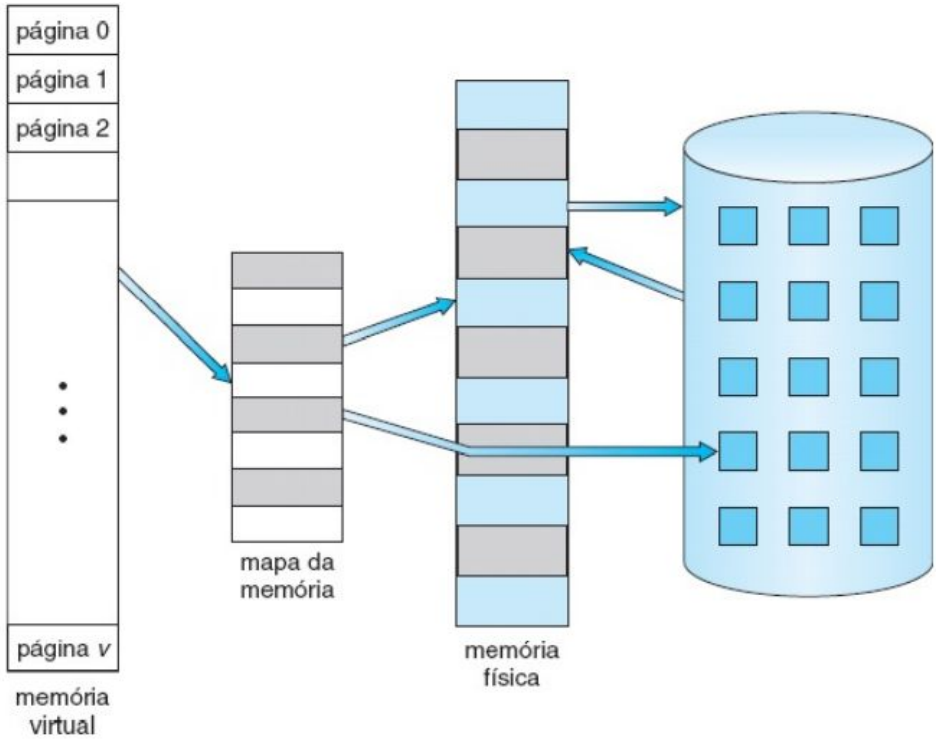
A memória virtual também permite que os processos compartilhem arquivos facilmente e implementem a memória compartilhada. Além disso, ela fornece um mecanismo eficiente para a criação de processos. **No entanto, a memória virtual não é fácil de implementar e pode piorar substancialmente o desempenho, se for usada sem cuidado**.

Memória Virtual

A **memória virtual** envolve a **separação** entre a **memória lógica** como percebida pelos usuários e a **memória física**.

Essa separação permite que uma **memória virtual extremamente grande** seja fornecida aos programadores quando apenas uma **memória física menor está disponível**.

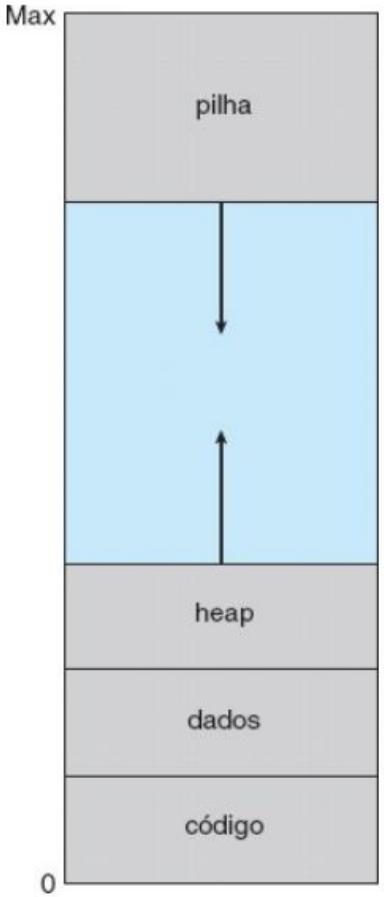
A **memória virtual** torna a tarefa de **programar muito mais fácil** porque o programador não precisa mais se preocupar com o montante de memória física disponível; em vez disso, ele pode se concentrar no problema a ser programado



Memória Virtual

O **espaço de endereçamento virtual** de um processo diz respeito à visão lógica (ou virtual) de como um processo é armazenado na memória. Normalmente, de acordo com essa visão, um processo começa em determinado endereço lógico — digamos, endereço 0 — e existe em memória contígua.

- Permitimos ao **heap** crescer para cima na memória conforme ele é usado na **alocação de memória dinâmica**;
- Permitimos que a **pilha** cresça para baixo na memória por meio de chamadas de função sucessivas.

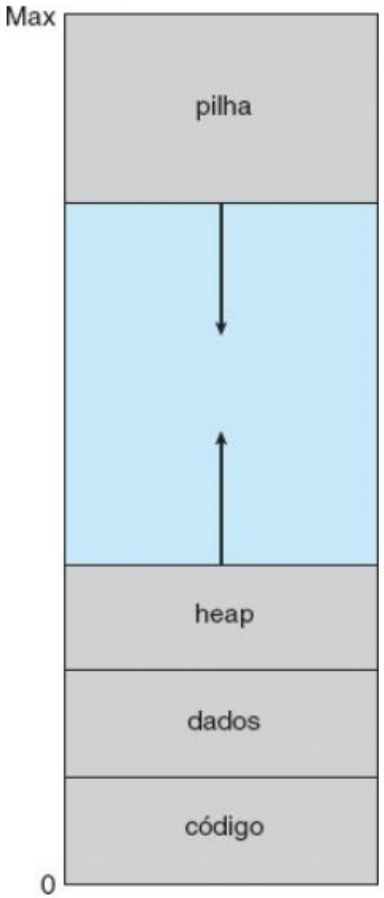


Memória Virtual

O grande espaço vazio (ou **brecha**) entre o **heap** e a **pilha** faz parte do **espaço de endereçamento virtual**, mas precisará de páginas físicas reais somente se o **heap** ou a **pilha** crescerem.

Os espaços de endereçamento virtuais que incluem **brechas** são conhecidos como espaços de endereçamento **esparsos**.

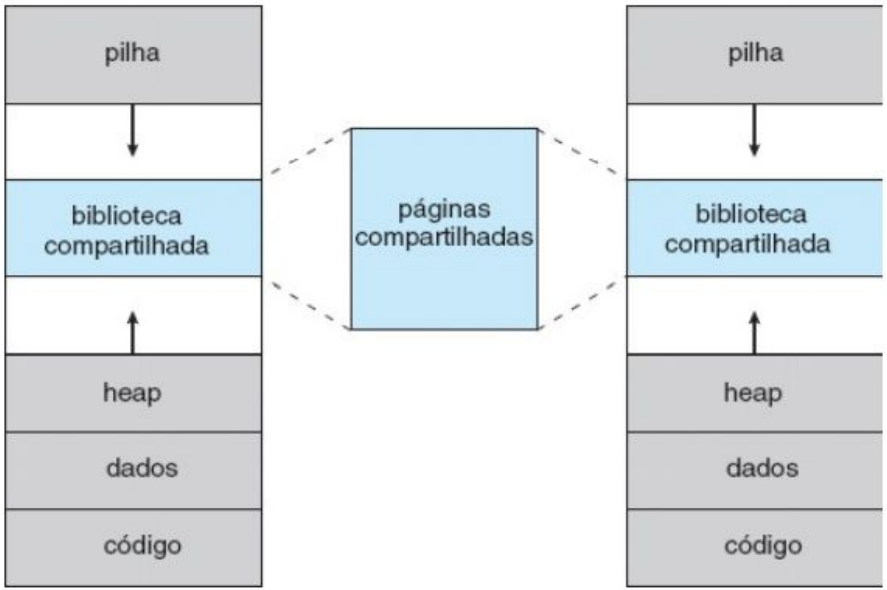
O uso de um espaço de endereçamento **esparso** é benéfico porque as **brechas** podem ser preenchidas conforme os segmentos da **pilha** ou do **heap** cresçam, ou se quisermos vincular bibliotecas dinamicamente.



Memória Virtual

Além de separar a memória lógica da memória física, a **memória virtual** permite que arquivos e memória sejam compartilhados por dois ou mais processos através do compartilhamento de páginas. Isso traz os benefícios a seguir:

- As bibliotecas do sistema podem ser compartilhadas por vários processos através do mapeamento do objeto compartilhado para um espaço de endereçamento virtual;
- Da mesma forma, processos podem compartilhar memória.



Paginação por Demanda

Paginação por Demanda

Considere como um programa executável pode ser carregado de disco para a memória. Uma opção é carregar **o programa inteiro na memória física** em tempo de execução do programa. No entanto, um problema dessa abordagem é que inicialmente podemos não **precisar** do programa inteiro na memória.

Uma estratégia alternativa é carregar páginas somente à medida que elas sejam necessárias. Essa técnica é conhecida como **paginação por demanda** e é comumente usada em sistemas de **memória virtual**.

Na **memória virtual paginada por demanda**, as páginas são carregadas somente quando são necessárias durante a execução do programa. Portanto, páginas que nunca são acessadas nunca são carregadas na memória física

Paginação por Demanda

Quando um **processo** está para ser **inserido** na memória, o **paginador** avalia as **páginas** que serão usadas, antes que o processo seja removido novamente.

Em vez de **inserir** um processo **inteiro**, o **paginador** traz apenas essas páginas para a memória.

Portanto, ele **evita** que sejam **transferidas** para a memória **páginas** que não serão **usadas**, **diminuindo** o **tempo** de **permuta** e o **montante** de **memória física** necessária.

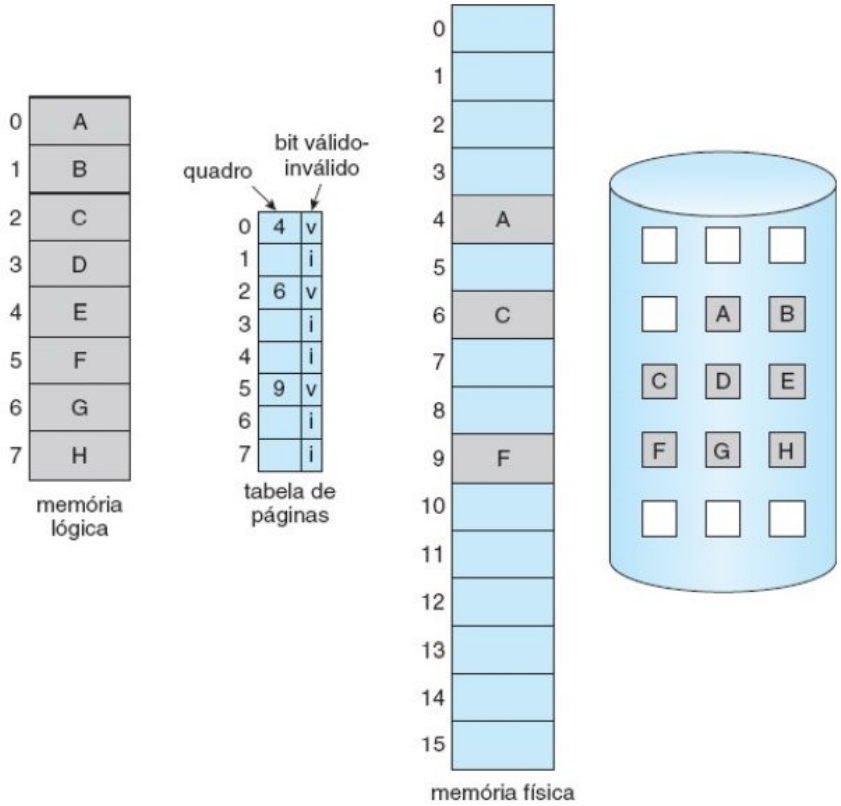
Nesse esquema, precisamos de algum tipo de suporte de hardware para diferenciar as páginas que estão na memória das páginas que estão em disco

Paginação por Demanda

Tabela de páginas: essa tabela pode marcar uma entrada como inválida por meio de um bit válido-inválido ou de um valor especial de bits de proteção.

Memória secundária: essa memória mantém as páginas que não estão presentes na memória principal.

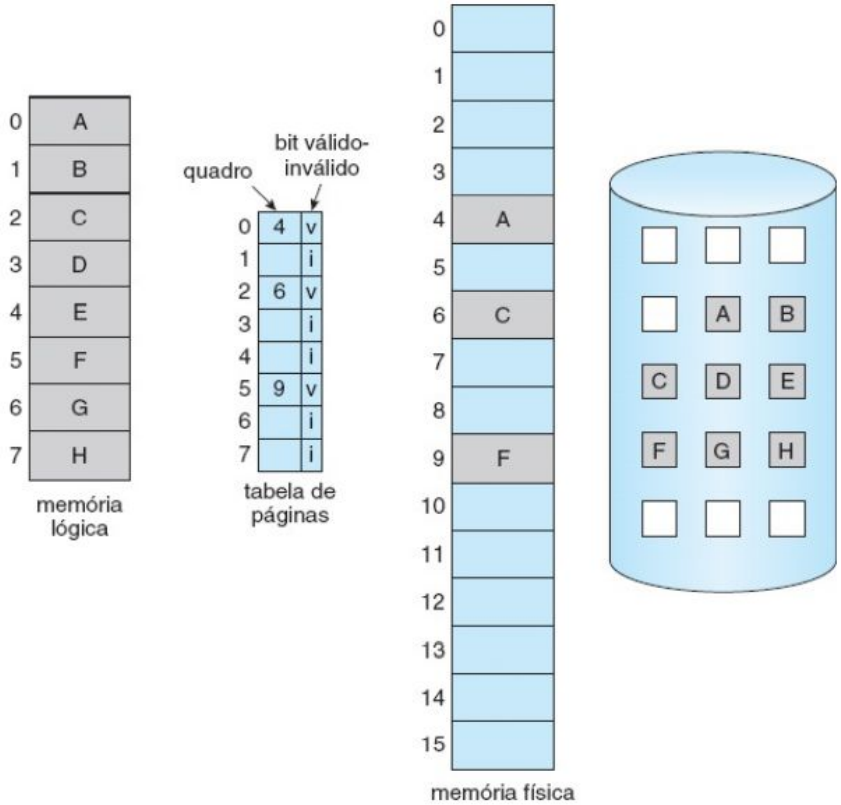
A memória secundária é, usualmente, um disco de alta velocidade. Ela é conhecida como **dispositivo de permuta**, e a seção de disco usada para esse fim é conhecida como **espaço de permuta**.



Paginação por Demanda

Mas o que acontece se o processo tentar acessar uma **página** que não foi trazida para a memória?

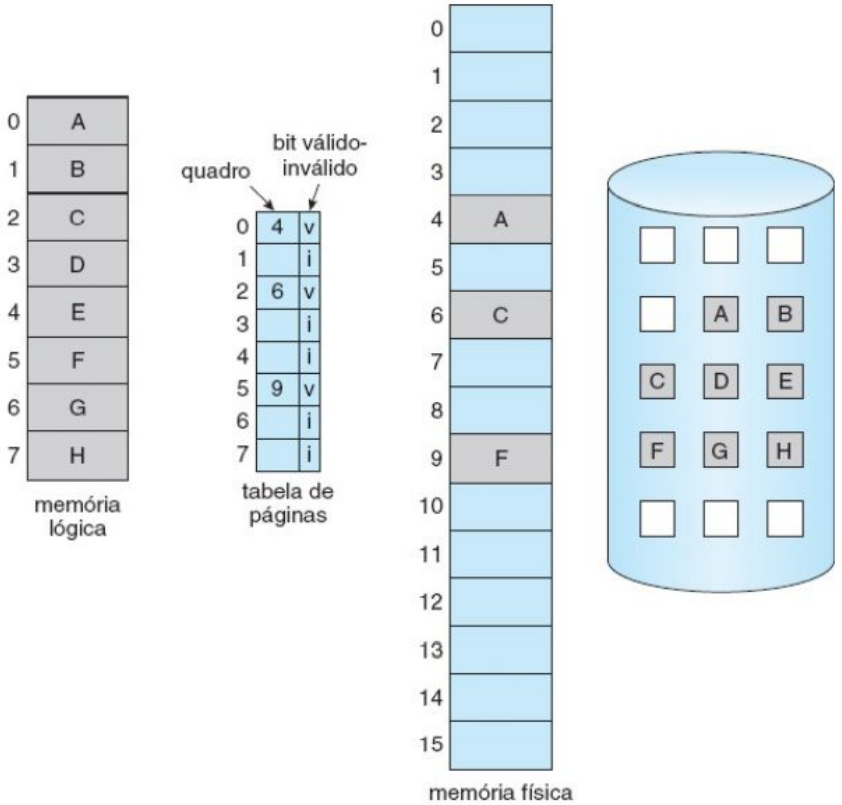
O acesso a uma página marcada como inválida causa um **erro de página**.



Paginação por Demanda

O procedimento para manipulação desse erro de página é:

1. Verificamos uma tabela interna desse processo para determinar se a referência foi um acesso válido ou inválido à memória;
2. Se a referência foi inválida, encerramos o processo. Se ela foi válida, mas ainda não trouxemos a página para a memória, a traremos agora;
3. Encontramos um quadro livre (usando um da lista de quadros livres, por exemplo);
4. Incluímos uma operação de disco no schedule para ler a página desejada para o quadro recém alocado;
5. Quando a leitura em disco é concluída, modificamos a tabela interna mantida com o processo e a tabela de páginas para indicar que agora a página está na memória;
6. Reiniciamos a instrução que foi interrompida pela interceptação. Agora o processo pode acessar a página como se ela sempre tivesse estado na memória.



Paginação por Demanda

No caso extremo, podemos iniciar a execução de um processo **sem** páginas na memória. Quando o sistema operacional posiciona o ponteiro de instruções para a primeira instrução do processo que está em uma página **não residente na memória**, o processo é interrompido imediatamente pelo **erro de página**.

Após essa página ser trazida para a memória, o processo continua a ser executado, sendo interrompido, se necessário, até que cada página de que ele precise esteja na memória.

Nesse ponto, ele pode ser executado sem mais erros. Esse esquema é a **paginação por demanda pura**: **nunca trazer uma página para a memória antes que ela seja necessária**.

Paginação por Demanda

Um **requisito crucial** da **paginação por demanda** é a capacidade de reiniciar qualquer instrução após um **erro de página**.

Já que salvamos o estado (registradores, código de condição, contador de instruções) do processo interrompido quando o **erro de página** ocorre, devemos ser capazes de reiniciar o processo exatamente no mesmo local e estado, exceto pelo fato de a página desejada estar agora na memória e poder ser acessada.

Na maioria dos casos, esse requisito é fácil de alcançar. Um erro de página pode ocorrer em qualquer referência à memória. Se o erro de página ocorrer na busca da instrução, poderemos reiniciar buscando a instrução novamente.

Substituição de Páginas

Substituição de Páginas

Sobre a taxa de **erros de página**, consideramos que cada página falha no máximo uma vez, quando é referenciada inicialmente.

Se um processo de **10 páginas** utiliza realmente apenas metade delas, então a paginação por demanda economiza o I/O necessário à carga das cinco páginas que nunca são usadas.

Também poderíamos aumentar nosso nível de multiprogramação executando o **dobro de processos**.

Essa representação não é muito precisa.

Precisamos estabelecer uma Substituição de Páginas Básica

Substituição de Páginas

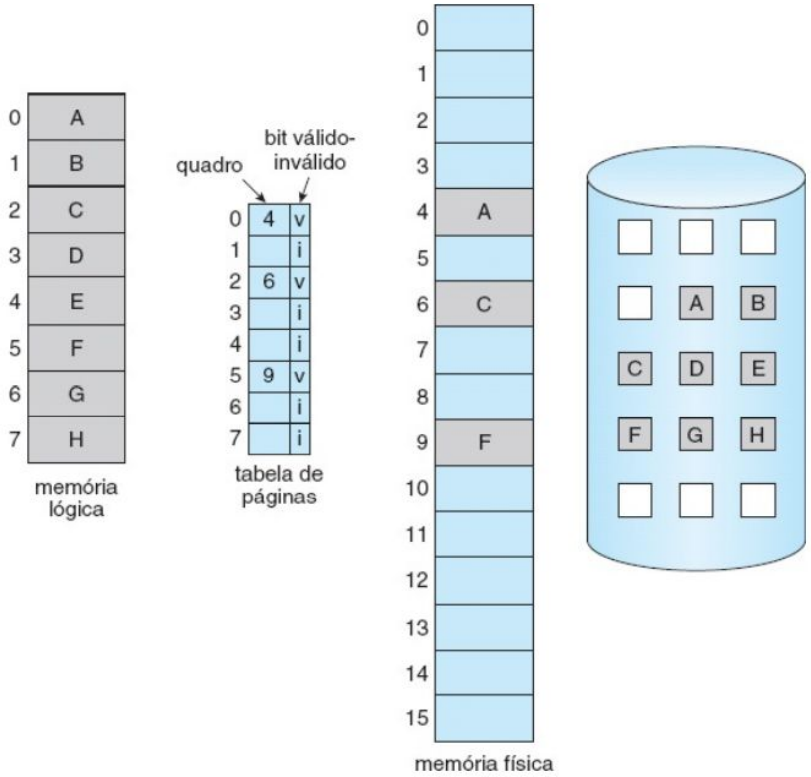
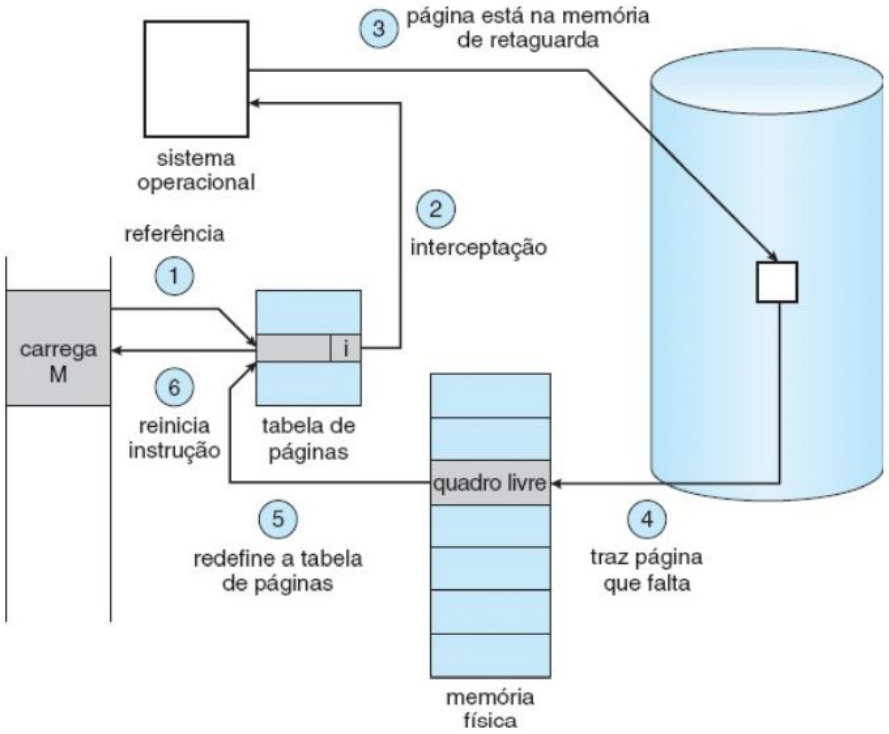


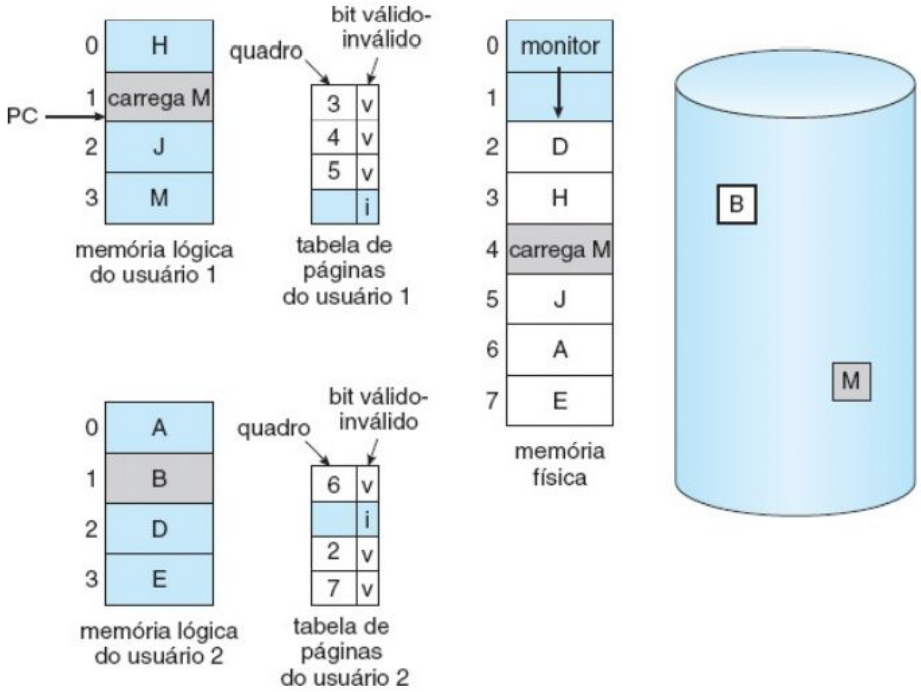
Tabela de páginas quando algumas páginas não estão na memória principal

Substituição de Páginas



Passos para a manipulação de um **erro de página**.

Substituição de Páginas



Necessidade de substituição de páginas.

Substituição de Páginas

1. Encontre a locação da página desejada no disco.
2. Procure um quadro livre:
    a. Se houver um quadro livre, use-o.
    b. Se não houver um quadro livre, use um **algoritmo de substituição de páginas** para selecionar um quadro vítima.
    c. Grave o quadro vítima no disco e promova as devidas alterações nas tabelas de páginas e de quadros.
3. Transfira a página desejada para o quadro recém-liberado e altere as tabelas de páginas e quadros.
4. Retome o processo do usuário a partir de onde ocorreu o erro de página.