



Estratégia
CONCURSOS

Aula

Arquitetura e Sistemas Operacionais pf Prefeitura de Betim (Analista de Sistemas) - Pós-Edital

Professor: Equipe Informática e TI, Evandro Dalla Vecchia Pereira

Gerência de Memória	2
<i>Multiprogramação com Partições Fixas</i>	<i>2</i>
<i>Swapping</i>	<i>3</i>
<i>Memória Virtual</i>	<i>6</i>
<i>Algoritmos de Substituição em Segmentação</i>	<i>9</i>
<i>Algoritmos de Substituição de Página</i>	<i>9</i>
<i>Questões Comentadas</i>	<i>11</i>
Lista de Questões	21
Gabarito	25



PROF. EVANDRO DALLA VECCHIA

Autor do livro "Perícia Digital - Da investigação à análise forense", Mestre em Ciência da Computação (UFRGS), Bacharel em Ciência da Computação (PUCRS), Técnico em Redes de Computadores (Ecom/UFRGS) e em Processamento de Dados (Urcamp). Perito Criminal na área de Perícia Digital desde 2004 no Instituto-Geral de Perícias/RS. Professor de pós-graduação em diversas instituições, nas áreas de Perícia Digital, Perícia Criminal e Auditoria de Sistemas. Lecionou na graduação de 2006 a 2017, nas instituições PUCRS, Unisinos, entre outras. Professor em cursos de formação e aperfeiçoamento de Peritos Criminais, Delegados, Inspetores, Escrivães e Policiais Militares.

Áreas de cursos ministrados pelo professor no Estratégia: Computação Forense, Arquitetura de Computadores e Sistemas Operacionais.

Entre em contato:   profevandrodallavecchia





GERÊNCIA DE MEMÓRIA

O componente central de um sistema operacional que determina o local da memória onde deverá ser colocado o código de um novo processo, lido de um arquivo previamente armazenado em um dispositivo de entrada e saída (HD, por exemplo) é denominado **gerenciador de memória**.

Antes de avançar no assunto é importante conhecer o termo **memory leak** (vazamento de memória) que pode ocorrer em duas situações:

1. Blocos de memória estão alocados e disponíveis para serem utilizados pelo programa, mas não são acessíveis porque a aplicação não tem nenhum ponteiro apontando para essa área de memória. Ou seja, tais blocos de memória não podem ser utilizados nem pelo programa nem por outros processos (ficam alocados e sem acesso!);
2. Blocos de memória possuem dados que poderiam ser liberados por estarem inacessíveis e que, por algum “esquecimento”, ainda são referenciados no código. Ou seja, mesmo sem estarem sendo usados, não podem ser liberados.

Os gerenciadores de memória podem ser divididos em duas classes: os que alternam os processos entre a memória principal e o disco durante a execução (*swapping*) e os que não alternam. Nosso foco será na primeira classe, visto que é basicamente o que é utilizado há um bom tempo e o que é cobrado em concurso!

MULTIPROGRAMAÇÃO COM PARTIÇÕES FIXAS

Quando existe mais de um processo na memória por vez (o que é o mais comum), alguma forma de organização da memória deve ser estabelecida. A mais simples é a divisão da memória em N partições (possivelmente com tamanhos diferentes, porém de tamanho fixo). Tais partições podem ser estabelecidas na configuração do sistema operacional, por exemplo.

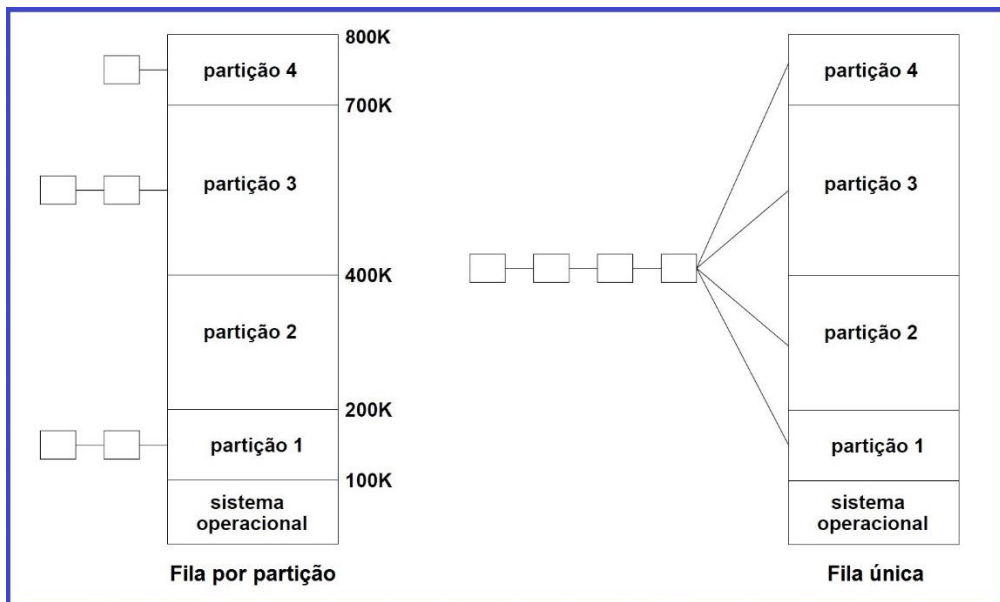
Ao ser inicializado, um processo pode ser colocado em uma fila de entrada para ocupar a menor partição de tamanho suficiente para carregá-lo. Como as partições são fixas, qualquer espaço em uma partição não utilizado pelo processo é desperdiçado, afinal de contas **apenas um processo pode ocupar uma partição**.

Basicamente pode-se adotar a estratégia de ter uma **fila para cada partição** (de acordo com o tamanho do processo, para evitar tanto desperdício) ou uma **fila única** (mais fácil de implementar, mas não evita desperdício). A figura a seguir ajuda a esclarecer (detalhe: se o processo tiver tamanho menor ou igual a 100 KB, ele pode ir para a fila da partição 1 ou 4, se for adotada a estratégia de fila por partição).

Essa estratégia com partições fixas definidas pelo operador foi usado pelo sistema operacional OS/360 (*mainframes* da IBM) e era chamado de MFT (*Multiprograming with a Fixed number of Task*).



Como vimos ele é simples de se entender e de implementar e, embora seja considerado obsoleto, cai em prova de concurso!

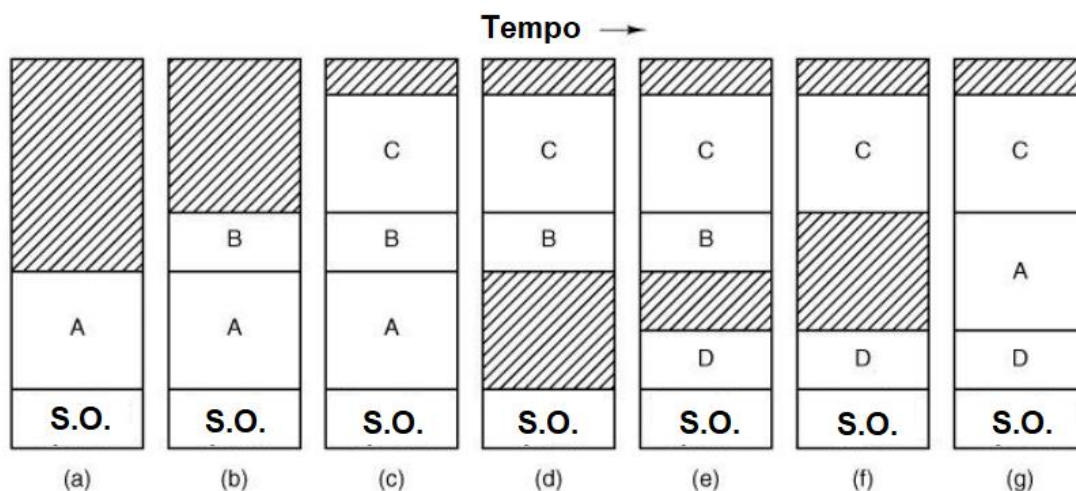


SWAPPING

Há momentos em que não há memória principal suficiente para todos os processos ativos, de maneira que os processos excedentes devem ser mantidos em disco e trazidos para a memória dinamicamente. Para isso pode ser usado uma das seguintes estratégias:

- **Swapping:** traz cada processo em sua totalidade, executa-o por algum tempo e o coloca novamente no disco;
- **Memória virtual:** os programas podem ser executados mesmo estando apenas parcialmente na memória principal.

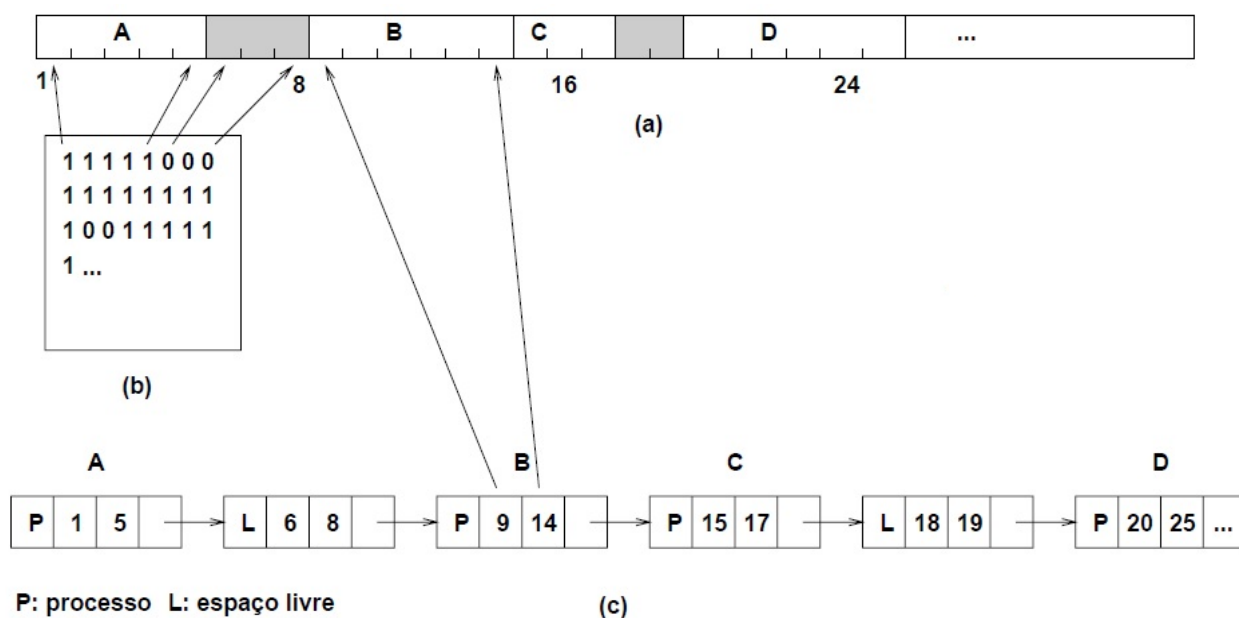
Vamos focar agora no *swapping*...analise a figura abaixo e em seguida vamos comentá-la.



Podemos ver a memória em diferentes instantes de tempo, sendo que o S.O. ocupa uma pequena parte (embaixo) e o restante é destinado para processos do usuário. Inicialmente apenas o processo A foi carregado, em seguida o processo B e depois o C. Quando falamos “carregado” pode ser porque acabou de ser criado ou foi recuperado do disco. Digamos que o processo D precisa ser carregado, porém não há espaço para ele. Então A foi colocado em disco e D foi colocado em seu lugar (na verdade sobrou espaço). Na sequência foi necessário carregar A, e para isso B teve que ser retirado da memória e colocado em disco.

Na figura podemos ver que fragmentos são criados na medida que processos são carregados ou retirados da memória. Para evitar isso é possível utilizar a técnica de **compactação de memória**, que move os processos “para baixo” quando há algum espaço livre, deixando a parte “de cima” sem fragmentos. Por exemplo, na parte (d) da figura os processos B e C seriam deslocados para baixo, “encostando” no S.O. Porém, essa técnica exige muito tempo de CPU e geralmente não é utilizada.

Gerenciamento de memória com mapa de bits: a memória é dividida em unidades de alocação, desde um pequeno número de palavras até muitos Kilobytes (ex.: 4KB). Para cada unidade de alocação existe um bit no mapa de bits, que é 0 se a unidade estiver livre e 1 caso esteja ocupada (ou vice-versa, dependendo da implementação). Fica mais fácil entender olhando a figura abaixo:



Na figura vemos:

- (a) um pedaço da memória contendo 4 processos (A, B, C e D). Observamos também quantas unidades de alocação cada processo ocupa: A = 5, B = 6, C = 3 e D = 6. Se a unidade de alocação tiver 4KB, por exemplo, o processo A teria o tamanho entre (16KB + 1 byte) e 20KB, pois não há garantia que o processo ocupará toda a última unidade de alocação (seria muita “sorte”);
- (b) um mapa de bits correspondente ao pedaço de memória mostrado em (a), ou seja, para cada unidade de alocação ocupada há um bit 1 e para cada unidade de alocação livre existe um bit 0. Você pode contar cinco bits 0 apenas, pois são correspondentes àqueles “quadrados

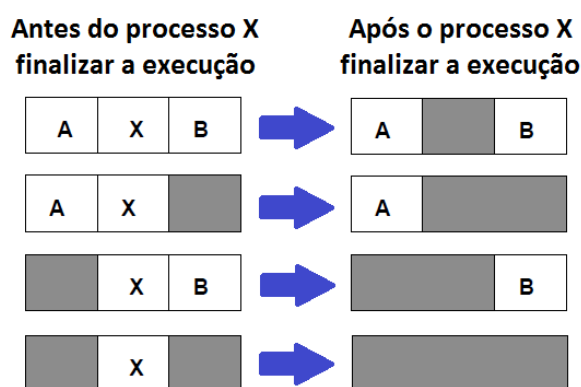
pintados” e ainda pode ver que eles aparecem nas posições 6, 7, 8, 18 e 19. O resto das unidades de alocação está ocupado (bit 1);

- (c) uma lista encadeada reapresentando as mesmas informações mostradas no item (b), da seguinte forma: “P” representa que está ocupado por um processo, o campo seguinte informa a posição inicial, o seguinte a posição final e o último campo é um ponteiro para o próximo nodo da lista. Se estiver livre, no lugar “P” é colocado um “L”.

O tamanho de cada unidade de alocação é uma importante característica de projeto. Para unidades de alocação menores tem-se um mapa de bits maior. Entretanto, mesmo com uma unidade de alocação tão pequena como com 4 bytes, 32 bits de memória irão requerer somente 1 bit no mapa ($1/32 = 3,1\%$ da memória). Se a unidade de alocação for grande, o mapa de bits será pequeno, mas uma porção considerável da memória pode ser desperdiçada (lembre daquele finalzinho do processo na última unidade de alocação, afinal de contas é difícil ter um processo com um tamanho exatamente múltiplo exato da unidade de alocação).

O problema principal do mapa de bits é que quando se decide trazer um processo de N unidades de alocação para a memória, o gerenciador de memória deve pesquisar no mapa de bits uma sequência de N bits 0 consecutivos no mapa. Tal operação é lenta, motivo pelo qual essa estratégia ser evitada na prática, mas...é cobrada em concurso!

Gerenciamento de memória com listas encadeadas: uma outra maneira de gerenciar a memória é manter uma lista de alocações e segmentos de memória livre, onde um segmento é um processo ou um espaço livre entre dois processos. Podemos voltar à figura mostrada no mapa de bits e verificar que no item (c) há uma lista encadeada de segmentos e em cada nodo podemos ver se aquele “pedaço” está ocupado por um processo (P) ou livre (L). Quando um processo é terminado, ao seu lado podemos ter outros processos ou espaços livres. Quatro situações podem ocorrer (“X” é o processo que vai finalizar sua execução):



MEMÓRIA VIRTUAL

Um fato notório na evolução da informática é que o tamanho dos programas vem superando a quantidade de memória disponível para carregá-los. Uma solução que geralmente era adotada no passado era a divisão do programa em partes (**overlays**). O **overlay** 0 começa a ser executado primeiro. Quando finaliza, o próximo **overlay** é executado, e assim sucessivamente. Os **overlays** eram mantidos em disco e permutados para a memória pelo sistema operacional.

O grande problema é que a divisão do programa era deixada a cargo do programador, então dependia da habilidade desse profissional. Surgiu então um método que deixa esse particionamento do programa a cargo do sistema operacional: a **memória virtual**. A ideia básica é que a combinação do tamanho do programa, dados e pilha, pode exceder a quantidade de memória física disponível. Por exemplo, você pode ter uma máquina com 8GB de memória RAM e ter diversos processos em execução, totalizando 10GB (2GB a mais que a memória física).

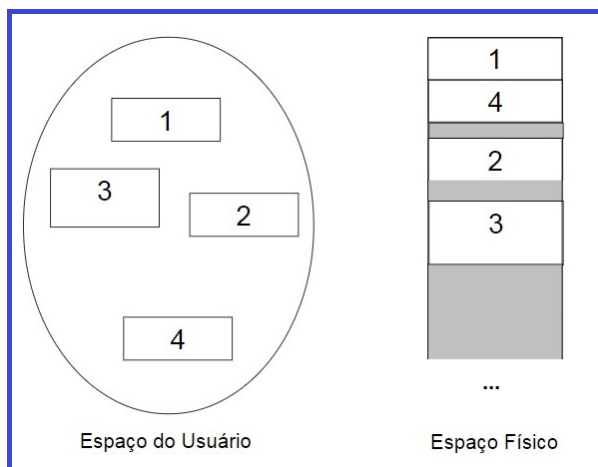
Qual a mágica? O S.O. mantém aquelas partes do programa correntemente em uso na memória principal e o resto no disco. Digamos que um programa de tamanho 50MB esteja executando em uma máquina que aloca 5MB de memória por processo, escolhendo-se cuidadosamente quais dos 5MB será mantido na memória a cada instante, com segmentos sendo permutados entre disco e memória assim que forem necessários. Vamos ver a seguir duas formas de implementar a memória virtual: a segmentação e a paginação.

Segmentação: os programas são divididos em segmentos de tamanhos diversos, cada um com o seu próprio espaço de endereçamento. A alocação da memória ocorre de maneira não fixa, ou seja, cada segmento pode ter um tamanho diferente (a alocação depende da lógica do programa). O mapeamento é feito através das tabelas de mapeamento de segmentos e os endereços são compostos pelo número do segmento e um deslocamento dentro do segmento (ex.: segmento 0, deslocamento = 500 bytes).

Cada entrada na tabela mantém o endereço físico do segmento, o tamanho do segmento, se ele está ou não na memória e sua proteção. Para isso, o S.O. mantém uma tabela com as áreas livres e ocupadas da memória e somente segmentos referenciados são transferidos para a memória principal. Nesse modelo ocorre **fragmentação externa**, pois internamente não “sobra espaço”, mas externamente sempre pode ficar um “pedaço” pequeno sem uso, entre dois segmentos.

Imagine a seguinte situação: o usuário executa diversos programas, encerra alguns, abre outros, e assim por diante. Em algum momento existem os processos 1, 2, 3 e 4 no “espaço do usuário”, os primeiros três com tamanho semelhante (3,9 KB, 4 KB e 4,2 KB) e o quarto com tamanho 6 KB, além de outros processos. Esses outros processos foram terminados e apenas os quatro ficaram na memória (fora os processos do S.O.). Uma configuração possível seria a seguinte:



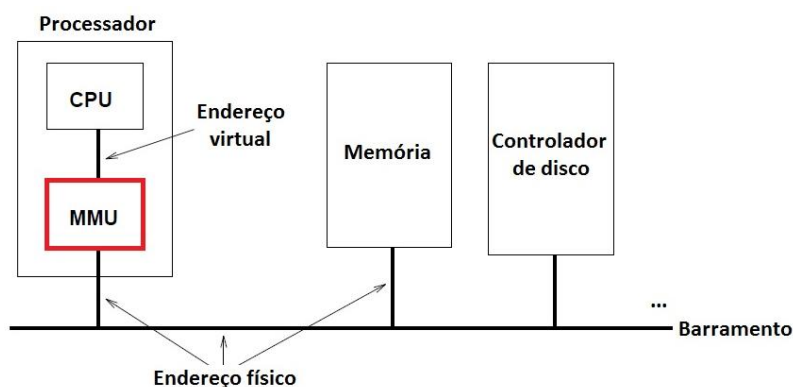


É possível verificar que os processos ocupam segmentos de tamanhos variáveis e existe uma fragmentação externa (fora do segmento), digamos que uma com cerca de 1KB (entre os processos 4 e 2) e uma com cerca de 1,5 KB (entre os processos 2 e 3).

Até pode ser utilizado algum algoritmo para compactação, para unir o 2 com o 4 e o 3 com o 2, eliminando os fragmentos, mas é um custo computacional elevado!

Paginação: técnica utilizada pela maioria dos sistemas com memória virtual. Em qualquer computador existe um determinado conjunto de endereços de memória que programas podem referenciar. Trata-se de endereços virtuais que formam o espaço virtual de endereçamento do processo. Em computadores sem memória virtual, o endereço virtual é colocado diretamente no barramento de memória uma palavra da memória física com mesmo endereço é lida ou escrita.

Com o uso da memória virtual, os endereços de memória não vão diretamente para o barramento de memória, eles vão à unidade de gerenciamento de memória (**MMU - Memory Management Unit**), um hardware específico que mapeia os endereços virtuais em endereços da memória física:

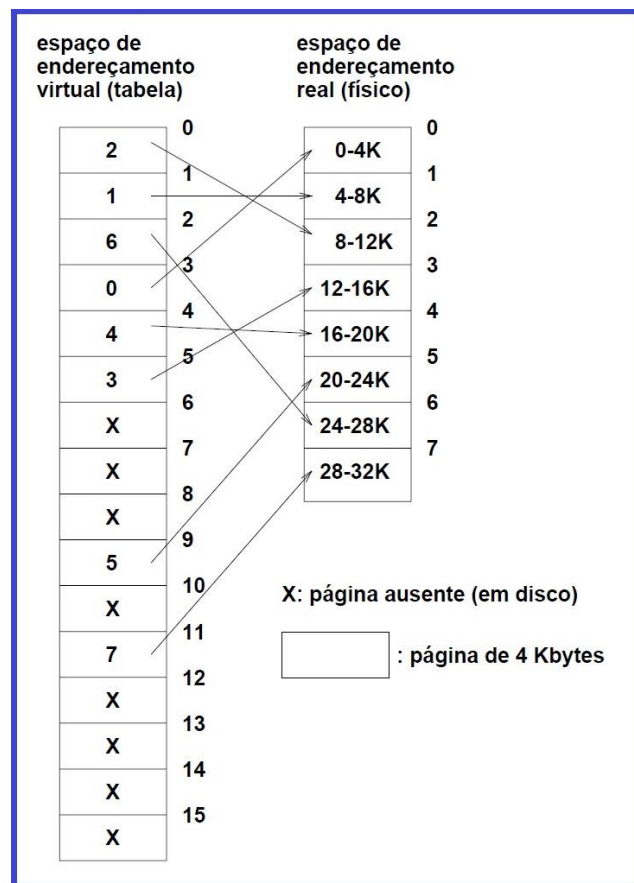


Vamos ver um exemplo de um computador que pode gerar endereços de 16 bits ($2^{16} = 65536 = 64\text{KB}$). Esses são os endereços virtuais. Agora imagine que tal computador tenha somente 32KB de memória física, então embora programas de 64KB possam ser escritos, eles não podem ser carregados totalmente para a memória para serem executados. Uma cópia completa do programa deve estar presente no disco e segmentos desse programa podem ser carregados para a memória pelo sistema na medida em que se tornem necessários.

O espaço de endereçamento virtual é dividido em unidades chamadas **páginas**. As unidades correspondentes na memória física são chamadas **quadros (page frames)**. Para não ter erro pense no seguinte: uma página deve ser “encaixada” em um quadro, uma “moldura física”. Assim fica mais fácil para não se confundir. ☺

As páginas e quadros são sempre do mesmo tamanho. No exemplo ao lado elas são de 4KB. Com 64KB de espaço de endereço virtual e 32KB de memória física, temos 16 páginas e 8 quadros. As transferências entre a memória e o disco sempre são realizadas em unidades de páginas (4KB, no nosso exemplo).

Quando um programa tenta acessar o endereço 5000, por exemplo, o endereço virtual 9000 é enviado para a MMU. Como temos páginas de 4KB (4096 bytes), sabemos que a página 0 possui os endereços 0-4095, a página 1 os endereços 4096-8191, a **página 2 os endereços 8192-12287**, e assim por diante.



A MMU reconhece que o endereço 9000 fica na página 2, que é mapeada para o quadro 6 (veja a figura). A transformação do endereço é realizada e colocada no barramento. A **tabela de memória** nem sabe da existência da MMU! Apenas sabe de uma requisição para leitura ou escrita no endereço 9000. Então, a MMU mapeou todo endereço virtual entre 8192 e 12287 (página 2) em endereço físico de 24576 a 28671 (quadro 6).

Agora digamos que o programa queira acessar o endereço 24576 (exatamente o 1º byte da página 6). Olhando na figura verificamos que se trata de uma página ausente! A MMU verifica que a página não está mapeada e força a CPU a causar uma interrupção (*trap*) no sistema operacional. Essa **interrupção** é chamada **falta de página (page fault)**. O S.O. remove o quadro menos usado e escreve seu conteúdo de volta no disco. Ele então busca a página referenciada para o quadro liberado, atualiza o mapa, e retorna à instrução interrompida.

Como as páginas são do mesmo tamanho (ex.: 4KB), é muito provável que a última página de um processo tenha um espaço não utilizado. Imagine um processo com tamanho 9KB, ele precisa de 3 páginas (2 “completas” + 1 usando apenas 1KB dos 4KB). Isso é conhecido como **fragmentação interna**, pois existem as páginas de tamanho fixo e algumas ficarão sem ser preenchidas (e nenhum outro processo pode utilizá-las).

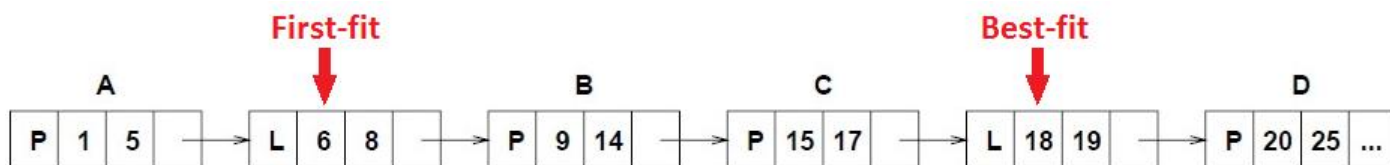
ALGORITMOS DE SUBSTITUIÇÃO EM SEGMENTAÇÃO

A partir do momento que processos (P) e espaços livres (L) são mantidos na lista ordenada por endereços, algoritmos podem ser utilizados para alocar memória, com o objetivo de criar ou permutar processos. Esses algoritmos são chamados quando o gerenciador de memória precisa de um segmento de memória de N bytes. Vamos analisar esses algoritmos a seguir.

First-fit: procura ao longo da lista de segmentos até encontrar um espaço livre de tamanho maior ou igual a N. Caso o espaço livre seja maior que N, o espaço livre é quebrado em dois segmentos: um para o processo (N bytes) e o outro para a memória que não foi utilizada. É um algoritmo rápido pois termina a busca o mais cedo possível.

Next-fit: funciona da mesma forma que o *First-fit*, com uma exceção: guarda a posição da lista onde o último espaço livre foi alocado. Quando for chamado novamente, o algoritmo começa a procurar a partir desse ponto.

Best-fit: procura pela **lista inteira** e “pega” o espaço livre de tamanho mais próximo de N. É lento e cria na memória espaços livres pequenos que dificilmente serão alocados. Porém, para N grande, esse algoritmo aumenta as chances de se encontrar na lista um espaço livre de tamanho adequado, visto que minimiza o uso de espaços livres grandes para atender requisições pequenas. Imagine que um bloco de tamanho 2 seja solicitado, o algoritmo *First-fit* alocaria o espaço livre 6-7, e o *Best-fit* o espaço livre 18-19:



Quick-fit: mantém listas separadas para tamanhos comumente solicitados, como por exemplo: uma lista para espaços livres de tamanho 4KB, outra para espaços de 8KB, e assim sucessivamente. Com esse algoritmo encontra-se um espaço livre de tamanho requerido muito rapidamente, mas com a desvantagem de todos os esquemas de classificar os espaços livres por tamanho. Ou seja, quando um processo termina ou é permutado para o disco, determinar seus vizinhos para uma possível fusão é uma operação custosa. Se fusões não forem realizadas, a memória rapidamente se fragmentará em um grande número de pequenos espaços livres não utilizáveis.

ALGORITMOS DE SUBSTITUIÇÃO DE PÁGINA

Ótimo: quando ocorre uma falta de página, um determinado conjunto de páginas está na memória. Uma dessas páginas será referenciada em muitas das próximas instruções. Outras páginas não serão referenciadas antes de 10, 100, ou quem sabe 1000 instruções. Cada página pode ser “marcada” com o número de instruções que serão executadas antes que a página seja inicialmente referenciada.





O algoritmo simplesmente diz que a página com o maior rótulo deve ser removida, adiando-se o máximo possível a próxima falta de página. O **único problema é que ele não é realizável**. No momento da falta de página, o sistema operacional não tem como saber quando cada página será referenciada.

NRU (Not Recently Used) “Não Recentemente Usada”: para que o S.O. saiba quais páginas são e quais não são utilizadas recentemente, muitos computadores com memória virtual possuem 2 bits associados à cada página. Um bit R (Referência) é ativado pelo hardware em qualquer leitura ou escrita de página, ou seja, se houve uma referência, marca com o bit 1. O outro bit, M (Modificação), é ativado pelo hardware quando uma página é escrita (modificada). É importante que esses bits sejam atualizados em qualquer referência de memória, assim, é essencial que eles sejam ativados pelo hardware. Uma vez que um bit foi ativado, ele permanece ativado até que o sistema operacional o desative (por software).

Os bits R e M podem ser usados para construir o algoritmo NRU, da seguinte forma: quando um processo é iniciado, ambos os bits de página (para todas as páginas) são declarados 0 pelo sistema operacional. A cada interrupção de relógio o bit R é zerado, para distinguir páginas que não foram referenciadas recentemente daquelas que tenham sido.

Quando uma falta de página ocorre, o S.O. verifica todas as páginas e as classifica em 4 categorias baseado nos valores correntes de seus bits R e M:

- Classe 0: não referenciada, não modificada;
- Classe 1: não referenciada, modificada;
- Classe 2: referenciada, não modificada;
- Classe 3: referenciada, modificada.

Interrupções de relógio não zeram o bit M porque essa informação é necessária para determinar se uma página terá que ser reescrita no disco ou não. O algoritmo NRU remove uma página aleatória da classe não vazia de numeração mais baixa (classe 0). As características principais do NRU é que ele é fácil de entender, eficiente e gera um desempenho que é tido como adequado.

LRU (Least Recently Used) “Menos Usada Recentemente”: quando ocorre uma falta de página, retira-se a página que não tem sido usada por um tempo longo. É um algoritmo teoricamente realizável, porém **seu custo é alto**. Para sua implementação completa, é necessário manter uma lista encadeada de todas as páginas em memória, com a página mais recentemente usada no início e a menos recentemente usada no fim.

A dificuldade é que a lista deve ser atualizada em toda referência (leitura ou escrita) de memória. Encontrar a página na lista, removê-la de sua posição atual e movê-la para o início representa um grande esforço computacional!

FIFO (First In, First Out): o S.O. mantém uma lista de todas as páginas correntes na memória, sendo a página da “cabeça” da lista a mais antiga e a página do fim a carregada mais recentemente. Em uma falta de página, a página da cabeça é removida e a nova página carregada é acrescentada no fim da lista. É a tradicional fila que estamos acostumados em diversas situações na vida!

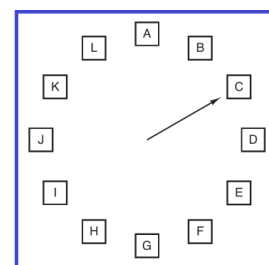


Segunda chance: no algoritmo FIFO pode ocorrer de uma página muito utilizada ser retirada, pois sua vez chegou no andamento da fila! Para dar uma “segunda chance” existe uma variação em relação ao FIFO. A ideia é primeiro examinar a página mais antiga como uma candidata a ser removida. Se seu bit R (Referência) for 0, a página é trocada imediatamente. Se o bit R for 1, o bit é zerado e a página é colocada no fim da lista de páginas (essa é a segunda chance da página!). Dessa forma a pesquisa continua. Resumindo: O algoritmo segunda chance busca por uma página antiga que não tem sido referenciada na interrupção de relógio anterior.

Relógio: utiliza uma lista ordenada de forma circular tal como um relógio. O ponteiro do relógio aponta para a página mais antiga e assim que ocorrer uma falta a página mais antiga é verificada. Se o bit R dessa página for 0 ela é substituída, caso contrário esse bit é zerado e o ponteiro aponta para a próxima página mais antiga. Esse processo é então repetido até a próxima página com o bit R=0 ser encontrada.

Vamos ver um exemplo:

Imagine a situação mostrada ao lado. Se a página C tiver R=0, então C será removida e o ponteiro apontará para D. Se C tiver R=1, R receberá o valor 0 e o ponteiro apontará para D (neste último caso, a página C não será removida).



QUESTÕES COMENTADAS

1. (2007 - NCE - SEF MG - Tecnologia da Informação)

O conceito que permite que o tamanho total de um programa, ou seja, seu código mais seus dados e a pilha, possa exceder a quantidade total de memória física disponível para ele é:

- A) Memória Virtual;
- B) Multiprocessamento;
- C) Compressão de Dados;
- D) "Best Fit";
- E) Temporização.

Comentários:

O nome já é bem intuitivo: “memória virtual” é aquela que não é a real (física). Podemos ter uma memória virtual maior que a real e ocorre *swapping* entre a memória real e o disco. No fim o disco acaba sendo a extensão da memória real, para “fingir” que a memória é maior do que realmente é! Podemos, por exemplo, executar processos que totalizem 5GB com uma memória RAM de 4GB!

Gabarito: A



2. (2016 - Cetro - Dataprev - Analista de Tecnologia da Informação)

Sobre as técnicas de gerenciamento de memória, assinale a alternativa que apresenta uma característica do gerenciamento de memória virtual por paginação

- A) O armazenamento é realizado por meio de endereçamento sequencial denominado "segmento".
- B) Gera fragmentação externa.
- C) Divide o espaço de endereçamento em blocos de tamanhos variados.
- D) Gera a fragmentação interna.
- E) Gera a fragmentação interna e a fragmentação externa.

Comentários:

(A) Os segmentos são utilizados pela técnica de segmentação, não o de paginação! (B) “Quem” gera a fragmentação externa é a técnica de segmentação; (C) Paginação divide o espaço de endereçamento em páginas do mesmo tamanho (ex.: 4KB); (D) Gera fragmentação interna sim, na última página. Pode ocorrer de não ter a fragmentação, se o processo tiver o tamanho múltiplo do tamanho da página, ex.: página = 4KB e processo = 12KB exatos! (E) Apenas a interna, pois todas páginas possuem o mesmo tamanho e ela pode estar ocupada (processo) ou livre.

Gabarito: D

3. (2016 - CESPE - TRE/PI - Cargo 6)

O componente central de um sistema operacional, que determina o local da memória onde deverá ser colocado o código de um novo processo chamado para ser executado por um processo pai, lido de um arquivo previamente armazenado em um dispositivo de entrada e saída, que, por sua vez, está conectado à rede local, é denominado

- A) gerenciador de sistema de arquivos.
- B) gerenciador de comunicação interprocessos.
- C) gerenciador de memória.
- D) escalonador de processos.
- E) gerenciador de entrada e saída.

Comentários:

Como vimos no 1º parágrafo sobre esse assunto:





O componente central de um sistema operacional que determina o local da memória onde deverá ser colocado o código de um novo processo, lido de um arquivo previamente armazenado em um dispositivo de entrada e saída (HD, por exemplo) é denominado **gerenciador de memória**.

Gabarito: C

4. (2017 - CESPE - TRF - 1ª Região - Analista Judiciário - Informática)

Na técnica denominada escalonamento de processos, o sistema operacional mantém parte do espaço de endereçamento de um processo na memória principal e parte em dispositivo de armazenamento secundário, realizando trocas de trechos de código e de dados entre eles, de acordo com a necessidade.

Comentários:

A questão está falando de memória virtual e *swapping*.

Gabarito: Errado

5. (2018 - CESPE - Polícia Federal - Escrivão de Polícia Federal)

A técnica de *swapping* consiste em transferir temporariamente um processo da memória para o disco do computador e depois carregá-lo novamente em memória.

Comentários:

No detalhe podemos falar que “Há momentos em que não há memória principal suficiente para todos os processos ativos, de maneira que os processos excedentes devem ser mantidos em disco e trazidos para a memória dinamicamente. Para isso pode ser usado uma das seguintes estratégias:”

- **Swapping**: traz cada processo em sua totalidade, executa-o por algum tempo e o coloca novamente no disco;
- **Memória virtual**: os programas podem ser executados mesmo estando apenas parcialmente na memória principal.

Note que a estratégia *swapping* transfere o processo por completo, enquanto a memória virtual permite partes.

Gabarito: Certo

6. (2018 - CESGRANRIO - LIQUIGÁS - Profissional Júnior - Analista de Sistemas)

A gerência de memória efetuada pelos sistemas operacionais inclui a definição de uma política para a substituição de páginas de memória, que trata da escolha de uma página da memória principal que deve ser substituída quando uma nova página precisa ser carregada.





Dentre as políticas básicas mais utilizadas, há uma que opta por substituir a página que está há mais tempo sem ser referenciada, sendo conhecida como

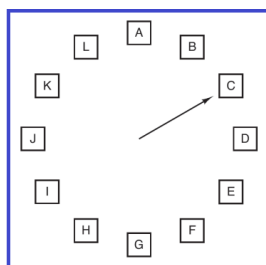
- A) FIFO.
- B) LIFO.
- C) Optimal.
- D) LRU.
- E) Clock Policy.

Comentários:

Vamos prestar atenção nesta parte: “substituir a página que está há mais tempo sem ser referenciada”. Ou seja, a que recentemente está sendo pouco usada. Vamos analisar cada sigla:

(A) FIFO = First In, First Out (uma fila); (B) LIFO = Last In, First Out (uma pilha, nem vimos nesta aula); (C) O algoritmo ótimo é aquele que “prevê” o futuro, não é implementável; (D) Least Recently Used = Menos Utilizado Recentemente (Opa! Só pela sigla já matamos a xarada!); (E) O relógio é aquele aprimoramento do algoritmo FIFO que considera também o bit R. Melhor relembrar o exemplo:

Imagine a situação mostrada ao lado. Se a página C tiver $R=0$, então C será removida e o ponteiro apontará para D. Se C tiver $R=1$, R receberá o valor 0 e o ponteiro apontará para D (neste último caso, a página C não será removida).



Gabarito: D

7. (2018 - COMPERVE - UFRN - Engenheiro - Engenharia da Computação)

Considere as afirmativas abaixo referentes à ocorrência de uma falta de página no gerenciamento de memória de um computador.

I É responsabilidade do sistema operacional detectar uma falta de página.

II O sistema operacional é acionado através de uma interrupção de hardware.

III O processo que sofre a falta de página passa para o estado de espera até que a página seja carregada na memória cache.

IV A falta de página é corrigida com a carga da página em falta, da memória virtual para a memória física.

Estão corretas as afirmações





A) II e III.

B) I e III.

C) II e IV.

D) I e IV.

Comentários:

(I) É responsabilidade do hardware (gera uma interrupção ao S.O.); (II) Isso mesmo, conforme acabamos de ver; (III) Nada disso, leia a IV; (IV) Agora sim! Faltou página? Tem que buscar no disco e carregar na memória!

Gabarito: C

8. (2018 - FGV - MPE-AL - Analista do Ministério Público - Desenvolvimento de Sistemas)

A implementação de linguagens de programação em geral depara-se com a questão do gerenciamento de memória. Nesse contexto, assinale a opção que melhor descreve o que é conhecido como “memory leak”.

A) A liberação de trechos de memória ainda em uso por um ou mais objetos.

B) A impossibilidade de liberar a memória ocupada por objetos que se tornaram inalcançáveis.

C) A incapacidade de recuperar trechos de memória virtualizados.

D) A invasão de trechos de memória por objetos não autorizados.

E) O compartilhamento indesejado de trechos de memória por dois ou mais objetos.

Comentários:

Memory leak (vazamento de memória) que pode ocorrer em duas situações:

1. Blocos de memória estão alocados e disponíveis para serem utilizados pelo programa, mas não são acessíveis porque a aplicação não tem nenhum ponteiro apontando para essa área de memória. Ou seja, tais blocos de memória não podem ser utilizados nem pelo programa nem por outros processos (ficam alocados e sem acesso!);
2. Blocos de memória possuem dados que poderiam ser liberados por estarem inacessíveis e que, por algum “esquecimento”, ainda são referenciados no código. Ou seja, mesmo sem estarem sendo usados, não podem ser liberados.

Gabarito: B



9. (2018 - FAURGS - TJ-RS - Analista de Suporte)

Em relação à paginação, assinale a alternativa correta.

- A) É uma forma de alocação não contígua de memória para o espaço de endereçamento de um processo.
- B) Apresenta como vantagem gerar fragmentação externa apenas na última página do processo.
- C) Divide o espaço de endereçamento do processo em quatro páginas: código, dados, heap (monte) e pilha.
- D) Elimina a fragmentação interna e reduz a fragmentação externa.
- E) É um método de gerência de memória usado apenas em sistemas que empregam memória real.

Comentários:

Paginação: técnica utilizada pela maioria dos sistemas com memória virtual. Em qualquer computador existe um determinado conjunto de endereços de memória que programas podem referenciar. Trata-se de endereços virtuais que formam o espaço virtual de endereçamento do processo. Em computadores sem memória virtual, o endereço virtual é colocado diretamente no barramento de memória uma palavra da memória física com mesmo endereço é lida ou escrita.

A paginação normalmente gera **fragmentação interna na última página**, pois raramente um processo tem tamanho múltiplo da página! Um processo pode ter inúmeras páginas (depende o tamanho do processo e da página!). É utilizado em sistemas que utilizam **memória virtual**.

Gabarito: A

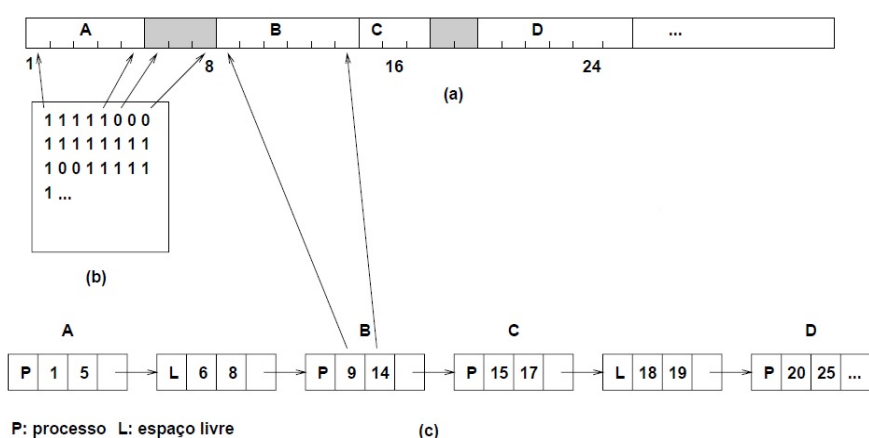
10.(2018 - CESPE - EBSEH - Técnico em Informática)

Uma das técnicas mais complexas para o gerenciamento do uso de memória é o mapa de bites, que consiste em manter uma lista encadeada de segmentos de memória alocados e disponíveis.

Comentários:

Como vemos na figura ao lado, o gerenciamento baseado em bits é relativamente simples, basta controlar as unidades de alocação que estão ocupadas ou livres (bits 1 ou 0).

Pode ser utilizada uma lista encadeada ou não, depende de como é realizada a implementação.





Gabarito: Errado

11.(2018 - FUNDATEC - AL-RS - Analista Legislativo - Analista de Tecnologia da Informação e Comunicação)

Para a gerência de memória de um sistema operacional, existem algoritmos de substituição de página. Um deles, de baixa sobrecarga, possui o seguinte modo de operação: (1) a primeira página a entrar é a primeira a sair; (2) pode ser implementado através de uma lista de todas as páginas correntemente na memória, sendo que a página mais antiga ocupa o início dessa lista e a mais recente ocupa o fim; e (3) quando falta uma página, a mais antiga é retirada e a nova é colocada no fim da lista. Trata-se do algoritmo:

- A) FIFO (First In, First Out).
- B) LRU (Least Recently Used).
- C) NRU (Not Recently Used).
- D) Ótimo.
- E) Segunda chance.

Comentários:

Esse algoritmo é o mais simples de entender. É uma fila que estamos acostumados no dia a dia. O primeiro a entrar é o primeiro a sair, sem prioridades ou coisas do tipo. Pode ser implementado com vetor, lista (claro que lista é melhor). Estamos falando do FIFO e o examinador ajudou colocando o significado da sigla entre parênteses. 😊

Gabarito: A

12.(2018 - COMPERVE - UFRN - Engenheiro - Engenharia da Computação)

Suponha uma memória composta por 5 partições fixas, sendo elas de 500MB (reservado e totalmente ocupado pelo sistema operacional), 200MB, 100MB, 74MB e 300MB, exatamente nesta ordem. O usuário lançou 4 processos A, B, C e D de tamanhos 99MB, 70MB, 250MB e 190MB, respectivamente. Logo em seguida, ele lança o processo E de 87 MB. Sabendo que a alocação da partição visa minimizar a fragmentação interna e que o sistema operacional utiliza memória virtual, o valor que corresponde à área da fragmentação interna da memória após a inserção do processo E é de

- A) 87 MB.
- B) 70 MB.
- C) 77 MB.
- D) 91 MB.

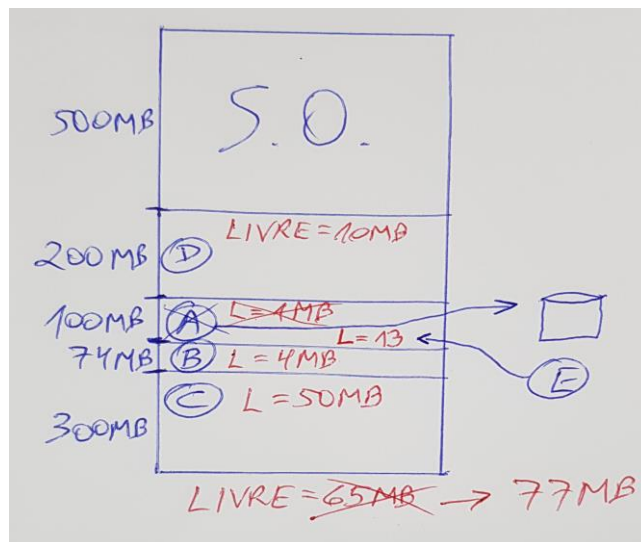




Comentários:

A questão não está muito fácil de ler. Ela quer dizer o seguinte: há 5 partições fixas, sendo que a primeira é destinada ao S.O. (500 MB). As outras 4 possuem os tamanhos 200 MB, 100 MB, 74 MB e 300 MB, nessa ordem. Atente-se à parte que diz “Sabendo que a alocação da partição visa minimizar a fragmentação interna e que o sistema operacional utiliza memória virtual”.

Então temos que “encaixar” cada processo na partição que sobre menos espaço “desperdiçado”. Acompanhando no desenho ao lado, colocamos os processos na ordem D, A, B, C. Quando surge um processo E (87MB), surge a pergunta: em que partição colocar?



Como não há mais partição livre e há memória virtual, temos que eleger um para tirar da memória (colocar no disco). A questão não fala em algoritmo para isso, mas sabemos que queremos evitar a fragmentação interna. Então devemos escolher a partição em que melhor se “encaixa” o processo E: a partição de tamanho 100 MB. Depois de tirar o processo A da memória e colocar o processo E em “seu lugar”, é só fazer o cálculo do espaço livre. 😊

Gabarito: C

13.(2019 - IF-PA - IF-PA - Técnico de Tecnologia da Informação)

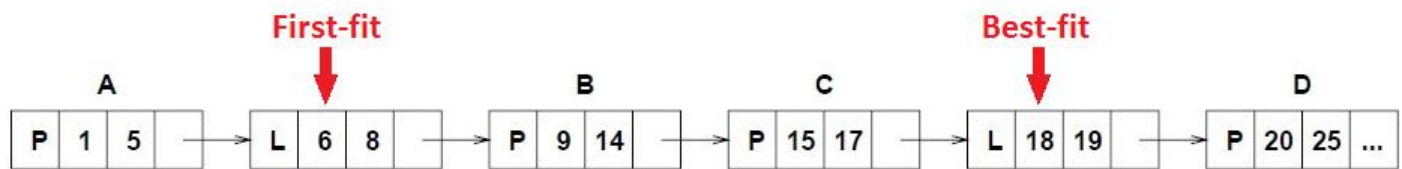
Em relação à estratégia de alocação de partição best-fit, marque a alternativa ERRADA:

- A) a melhor partição é escolhida.
- B) procura deixar o menor espaço de livre em uma partição.
- C) a pior partição de tamanho suficiente é escolhida.
- D) a lista de partições livres é ordenada por tamanho.
- E) aumenta o problema de fragmentação nas partições.

Comentários:

Best-fit: procura pela **lista inteira** e “pega” o espaço livre de tamanho mais próximo de N. É lento e cria na memória espaços livres pequenos que dificilmente serão alocados. Porém, para N grande, esse algoritmo aumenta as chances de se encontrar na lista um espaço livre de tamanho adequado, visto que minimiza o uso espaços livres grandes para atender requisições pequenas. Imagine que um bloco de tamanho 2 seja solicitado, o algoritmo *First-fit* alocaria o espaço livre 6-7, e o *Best-fit* o espaço livre 18-19:





Vamos ver qual está ERRADA:

(A) A melhor partição é escolhida, considerando que “melhor” quer dizer a que mais se “encaixa”, sobrando menos espaço de desperdício (que no fim causa fragmentações); (B) Exatamente o que vimos na explicação da A; (C) Essa está errada, é o oposto! A “melhor” é escolhida, conforme vimos na explicação da A; (D) No desenho que vemos acima temos uma única lista de processos e “partes” livres, mas é possível a implementação com listas de partições livres ordenadas por tamanho (inclusive melhora o desempenho); (E) Como o *best-fit* procura a partição em que o processo melhor se “enxaija”, sobram espaços pequenos (o que aumenta a fragmentação nas partições).

Gabarito: C

14.(2019 - IF-PA - IF-PA - Técnico de Tecnologia da Informação)

Em relação à gerencia de memória, marque a alternativa CORRETA:

- A) a técnica de overlay divide o programa em módulos para ser carregado em memória.
- B) a alocação continua simples é utilizada em sistemas multiprogramáveis.
- C) a alocação particionada estática é utilizada em sistemas monoprogramados.
- D) a técnica de memória virtual combina as memórias cachê e memória principal.
- E) a técnica de swapping é utilizada para resolver o problema de velocidade na memória secundária.

Comentários:

Um fato notório na evolução da informática é que o tamanho dos programas vem superando a quantidade de memória disponível para carregá-los. Uma solução que geralmente era adotada no passado era a divisão do programa em partes (*overlays*). O *overlay* 0 começa a ser executado primeiro. Quando finaliza, o próximo *overlay* é executado, e assim sucessivamente. Os *overlays* eram mantidos em disco e permutados para a memória pelo sistema operacional.

Gabarito: A

15.(2019 - IF-PA - IF-PA - Técnico de Tecnologia da Informação)

Em relação à estratégia de alocação de partição first-fit, marque a alternativa ERRADA:

- A) a primeira partição livre de tamanho suficiente é escolhida.
- B) realiza vários cálculos consumindo um alto poder de processamento.





- C) as listas de áreas livres estão ordenadas crescente e por endereços.
- D) quando comparada com as estratégias best-fit e worst-fit é mais rápida.
- E) consome menos recurso do sistema.

Comentários:

First-fit poderia ser traduzido para algo como “primeiro a se encaixar”. E é isso que ele faz! A primeira partição livre de tamanho suficiente é escolhida e a busca para nesse momento. Não tem cálculos com alto poder de processamento, só compara o tamanho livre com o tamanho do processo. É mais rápida que *best-fit* e *worst-fit*, pois essas duas precisam percorrer toda a lista! Por isso tudo, consome menos recursos! Podem ser implementadas listas com as áreas livres para tornar mais rápida a busca (mas não é obrigatório), ordenadas pelo tamanho livre e endereço. A errada é a alternativa B porque **não existem cálculos complexos! Apenas escolhe a primeira partição livre!**

Gabarito: B





LISTA DE QUESTÕES

1. (2007 - NCE - SEF MG - Tecnologia da Informação)

O conceito que permite que o tamanho total de um programa, ou seja, seu código mais seus dados e a pilha, possa exceder a quantidade total de memória física disponível para ele é:

- A) Memória Virtual;
- B) Multiprocessamento;
- C) Compressão de Dados;
- D) "Best Fit";
- E) Temporização.

2. (2016 - Cetro - Dataprev - Analista de Tecnologia da Informação)

Sobre as técnicas de gerenciamento de memória, assinale a alternativa que apresenta uma característica do gerenciamento de memória virtual por paginação

- A) O armazenamento é realizado por meio de endereçamento sequencial denominado "segmento".
- B) Gera fragmentação externa.
- C) Divide o espaço de endereçamento em blocos de tamanhos variados.
- D) Gera a fragmentação interna.
- E) Gera a fragmentação interna e a fragmentação externa.

3. (2016 - CESPE - TRE/PI - Cargo 6)

O componente central de um sistema operacional, que determina o local da memória onde deverá ser colocado o código de um novo processo chamado para ser executado por um processo pai, lido de um arquivo previamente armazenado em um dispositivo de entrada e saída, que, por sua vez, está conectado à rede local, é denominado

- A) gerenciador de sistema de arquivos.
- B) gerenciador de comunicação interprocessos.
- C) gerenciador de memória.
- D) escalonador de processos.
- E) gerenciador de entrada e saída.



4. (2017 - CESPE - TRF - 1ª Região - Analista Judiciário - Informática)

Na técnica denominada escalonamento de processos, o sistema operacional mantém parte do espaço de endereçamento de um processo na memória principal e parte em dispositivo de armazenamento secundário, realizando trocas de trechos de código e de dados entre eles, de acordo com a necessidade.

5. (2018 - CESPE - Polícia Federal - Escrivão de Polícia Federal)

A técnica de swapping consiste em transferir temporariamente um processo da memória para o disco do computador e depois carregá-lo novamente em memória.

6. (2018 - CESGRANRIO - LIQUIGÁS - Profissional Júnior - Analista de Sistemas)

A gerência de memória efetuada pelos sistemas operacionais inclui a definição de uma política para a substituição de páginas de memória, que trata da escolha de uma página da memória principal que deve ser substituída quando uma nova página precisa ser carregada.

Dentre as políticas básicas mais utilizadas, há uma que opta por substituir a página que está há mais tempo sem ser referenciada, sendo conhecida como

- A) FIFO.
- B) LIFO.
- C) Optimal.
- D) LRU.
- E) Clock Policy.

7. (2018 - COMPERVE - UFRN - Engenheiro - Engenharia da Computação)

Considere as afirmativas abaixo referentes à ocorrência de uma falta de página no gerenciamento de memória de um computador.

- I É responsabilidade do sistema operacional detectar uma falta de página.
- II O sistema operacional é acionado através de uma interrupção de hardware.
- III O processo que sofre a falta de página passa para o estado de espera até que a página seja carregada na memória cache.
- IV A falta de página é corrigida com a carga da página em falta, da memória virtual para a memória física.





Estão corretas as afirmações

- A) II e III.
- B) I e III.
- C) II e IV.
- D) I e IV.

8. (2018 - FGV - MPE-AL - Analista do Ministério Público - Desenvolvimento de Sistemas)

A implementação de linguagens de programação em geral depara-se com a questão do gerenciamento de memória. Nesse contexto, assinale a opção que melhor descreve o que é conhecido como “memory leak”.

- A) A liberação de trechos de memória ainda em uso por um ou mais objetos.
- B) A impossibilidade de liberar a memória ocupada por objetos que se tornaram inalcançáveis.
- C) A incapacidade de recuperar trechos de memória virtualizados.
- D) A invasão de trechos de memória por objetos não autorizados.
- E) O compartilhamento indesejado de trechos de memória por dois ou mais objetos.

9. (2018 - FAURGS - TJ-RS - Analista de Suporte)

Em relação à paginação, assinale a alternativa correta.

- A) É uma forma de alocação não contígua de memória para o espaço de endereçamento de um processo.
- B) Apresenta como vantagem gerar fragmentação externa apenas na última página do processo.
- C) Divide o espaço de endereçamento do processo em quatro páginas: código, dados, heap (monte) e pilha.
- D) Elimina a fragmentação interna e reduz a fragmentação externa.
- E) É um método de gerência de memória usado apenas em sistemas que empregam memória real.

10. (2008 2018 - CESPE - EBSERH - Técnico em Informática)

Uma das técnicas mais complexas para o gerenciamento do uso de memória é o mapa de bites, que consiste em manter uma lista encadeada de segmentos de memória alocados e disponíveis.



11.(2018 - FUNDATEC - AL-RS - Analista Legislativo - Analista de Tecnologia da Informação e Comunicação)

Para a gerência de memória de um sistema operacional, existem algoritmos de substituição de página. Um deles, de baixa sobrecarga, possui o seguinte modo de operação: (1) a primeira página a entrar é a primeira a sair; (2) pode ser implementado através de uma lista de todas as páginas correntemente na memória, sendo que a página mais antiga ocupa o início dessa lista e a mais recente ocupa o fim; e (3) quando falta uma página, a mais antiga é retirada e a nova é colocada no fim da lista. Trata-se do algoritmo:

- A) FIFO (First In, First Out).
- B) LRU (Least Recently Used).
- C) NRU (Not Recently Used).
- D) Ótimo.
- E) Segunda chance.

12.(2018 - COMPERVE - UFRN - Engenheiro - Engenharia da Computação)

Suponha uma memória composta por 5 partições fixas, sendo elas de 500MB (reservado e totalmente ocupado pelo sistema operacional), 200MB, 100MB, 74MB e 300MB, exatamente nesta ordem. O usuário lançou 4 processos A, B, C e D de tamanhos 99MB, 70MB, 250MB e 190MB, respectivamente. Logo em seguida, ele lança o processo E de 87 MB. Sabendo que a alocação da partição visa minimizar a fragmentação interna e que o sistema operacional utiliza memória virtual, o valor que corresponde à área da fragmentação interna da memória após a inserção do processo E é de

- A) 87 MB.
- B) 70 MB.
- C) 77 MB.
- D) 91 MB.

13.(2019 - IF-PA - IF-PA - Técnico de Tecnologia da Informação)

Em relação à estratégia de alocação de partição best-fit, marque a alternativa ERRADA:

- A) a melhor partição é escolhida.
- B) procura deixar o menor espaço de livre em uma partição.





- C) a pior partição de tamanho suficiente é escolhida.
- D) a lista de partições livres é ordenada por tamanho.
- E) aumenta o problema de fragmentação nas partições.

14.(2019 - IF-PA - IF-PA - Técnico de Tecnologia da Informação)

Em relação à gerencia de memória, marque a alternativa CORRETA:

- A) a técnica de overlay divide o programa em módulos para ser carregado em memória.
- B) a alocação continua simples é utilizada em sistemas multiprogramáveis.
- C) a alocação particionada estática é utilizada em sistemas monoprogramados.
- D) a técnica de memória virtual combina as memórias cachê e memória principal.
- E) a técnica de swapping é utilizada para resolver o problema de velocidade na memória secundária.

15.(2019 - IF-PA - IF-PA - Técnico de Tecnologia da Informação)

Em relação à estratégia de alocação de partição first-fit, marque a alternativa ERRADA:

- A) a primeira partição livre de tamanho suficiente é escolhida.
- B) realiza vários cálculos consumindo um alto poder de processamento.
- C) as listas de áreas livres estão ordenadas crescente e por endereços.
- D) quando comparada com as estratégias best-fit e worst-fit é mais rápida.
- E) consome menos recurso do sistema.

GABARITO

- | | | |
|-----------|------------|-------|
| 1. A | 8. B | 15. B |
| 2. D | 9. A | |
| 3. C | 10. Errado | |
| 4. Errado | 11. A | |
| 5. Certo | 12. C | |
| 6. D | 13. C | |
| 7. C | 14. A | |



ESSA LEI TODO MUNDO CONHECE: PIRATARIA É CRIME.

Mas é sempre bom revisar o porquê e como você pode ser prejudicado com essa prática.



1 Professor investe seu tempo para elaborar os cursos e o site os coloca à venda.



2 Pirata divulga ilicitamente (grupos de rateio), utilizando-se do anonimato, nomes falsos ou laranjas (geralmente o pirata se anuncia como formador de "grupos solidários" de rateio que não visam lucro).



3 Pirata cria alunos fake praticando falsidade ideológica, comprando cursos do site em nome de pessoas aleatórias (usando nome, CPF, endereço e telefone de terceiros sem autorização).



4 Pirata compra, muitas vezes, clonando cartões de crédito (por vezes o sistema anti-fraude não consegue identificar o golpe a tempo).



5 Pirata fere os Termos de Uso, adultera as aulas e retira a identificação dos arquivos PDF (justamente porque a atividade é ilegal e ele não quer que seus fakes sejam identificados).



6 Pirata revende as aulas protegidas por direitos autorais, praticando concorrência desleal e em flagrante desrespeito à Lei de Direitos Autorais (Lei 9.610/98).



7 Concurseiro(a) desinformado participa de rateio, achando que nada disso está acontecendo e esperando se tornar servidor público para exigir o cumprimento das leis.



8 O professor que elaborou o curso não ganha nada, o site não recebe nada, e a pessoa que praticou todos os ilícitos anteriores (pirata) fica com o lucro.



Deixando de lado esse mar de sujeira, aproveitamos para agradecer a todos que adquirem os cursos honestamente e permitem que o site continue existindo.