Lista de Exercícios de Sistemas Operacionais

1. O que é deadlock, quais as condições para obtê-lo e quais as soluções possíveis?

Deadlock é a situação em que um processo aguarda por um recurso que nunca estará disponível ou um evento que não ocorrerá. Para que ocorra a situação de deadlock, quatro condições são necessárias simultaneamente: exclusão mútua: cada recurso só pode estar alocado a um único processo em um determinado instante; posse e espera: um processo, além dos recursos já alocados, pode estar esperando por outros recursos; não-preempção: um recurso não pode ser liberado de um processo só porque outros desejam o mesmo recurso; espera circular: um processo pode ter de esperar por um recurso alocado a outro processo e vice-versa. Para prevenir a ocorrência de deadlocks, é preciso garantir que uma das quatro condições apresentadas, necessárias para sua existência, nunca se satisfaça. A prevenção de deadlocks evitando-se a ocorrência de qualquer uma das quatro condições é bastante limitada e, por isso, na prática não é utilizada. Uma solução conhecida como Algoritmo do Banqueiro (implementada com a presença das quatro condições) também possui várias limitações. A maior delas é a necessidade de saber com antecedência o número fixo de processos ativos e de recursos disponíveis no sistema. Essa limitação impede que a solução seja implementada na prática, pois é muito difícil prever o número de usuários no sistema e o número de recursos disponíveis. Desta forma, é preciso detector o deadlock e aplicar uma política para a recuperação do sistema.

2. Suponha um sistema computacional com 64Kb de memória principal e que utilize um sistema operacional de 14Kb que implemente alocação contígua de memória. Considere também um programa de 90Kb, formado por um módulo principal de 20Kb e três módulos independentes, cada um com 10Kb, 20Kb e 30Kb.
    a. Como o programa poderia ser executado utilizando-se apenas a técnica de overlay?
    b. Se o módulo de 30Kb tivesse seu tamanho aumentado para 40Kb, seria possível executar o programa? Caso não possa, como o problema poderia ser contornado?

A) Como existe apenas 50Kb para a execução do programa, a memória deve ser dividida em duas áreas: uma para o módulo principal (20Kb) e outra de overlay para a carga dos módulos, em função do tamanho do maior módulo (30 Kb).
B) Não. No caso de não haver como aumentar o espaço de memória real, a única solução seria tentar alterar o programa de forma que o módulo de 40Kb pudesse ser dividido em outros módulos menores independentes.

3. Qual a diferença entre fragmentação interna e externa da memória principal?

Fragmentação interna ocorre em espaços livres e contíguos na memória principal que são pré-alocados por processos, não possibilitando, portanto, o uso por outros processos. Fragmentação externa ocorre em espaços livres e contínuos, porém tão pequenos que não possibilitam a alocação de programas por processos.

4. Considerando as estratégias para escolha da partição dinamicamente, conceitue as estratégias first-fit, best-fit e worst-fit especificando prós e contras de cada uma.

First Fit: Escolhe o primeiro bloco livre de tamanho suficiente para caber o programa. Neste algoritmo a lista de blocos livres é ordenada por endereços. O algoritmo possui baixa complexidade.
Best Fit: Escolhe o melhor bloco livre, ou seja, aquele em que o programa deixa o menor espaço sem utilização. O problema é que a tendência é deixar cada vez mais a memória com pequenos blocos livres não contíguos, aumentando a fragmentação. Além do aumento da fragmentação, esta estratégia é mais custosa.
Worst Fit: Escolhe o pior bloco livre, ou seja, aquele em que o programa deixa maior espaço sem utilização. A ideia é deixar espaços maiores para que outros programas possam utilizá-los.

5. Considere um sistema que possua as seguintes área livres na memória principal, ordenadas crescentemente: 10Kb, 4Kb, 20Kb, 18Kb, 7Kb, 9Kb, 12Kb e 15Kb. Para cada programa abaixo, qual seria a partição alocada utilizando-se as estratégias first-fit, best-fit e worst-fit?

   a) 12Kb
   b) 10Kb
   c) 9Kb

   First-fit: 20Kb, 10Kb e 18Kb
   Best-fit: 12Kb, 10Kb e 9Kb.
   Worst-fit: 20Kb, 18Kb e 15Kb

6. Um sistema utiliza alocação particionada dinâmica como mecanismo de gerência de memória. O sistema operacional aloca uma área de memória total de 50Kb e possui, inicialmente, os programas da tabela a seguir:

5Kb	Programa A
3Kb	Programa B
10Kb	Livre
6Kb	Programa C
26Kb	Livre

   Realize as operações abaixo sequencialmente, mostrando o estado da memória após cada uma delas. Resolva a questão utilizando as estratégias best-fit, worst-fit e first-fit.

a) alocar uma área para o programa D que possui 6 Kb;
b) liberar a área do programa A;
c) alocar uma área para o programa E que possui 4 Kb.

Estratégia Best-fit:

(a)

5Kb	Programa A
3Kb	Programa B
6Kb	Programa D
4Kb	Livre
6Kb	Programa C
26Kb	Livre

(b)

5Kb	Livre
3Kb	Programa B
6Kb	Programa D
4Kb	Livre
6Kb	Programa C
26Kb	Livre

(c)

5Kb	Livre
3Kb	Programa B
6Kb	Programa D
4Kb	Programa E
6Kb	Programa C
26Kb	Livre

Estratégia Worst-fit:

(a)

5Kb	Programa A
3Kb	Programa B
10Kb	Livre
6Kb	Programa C
6Kb	Programa D
20Kb	Livre

(b)

5Kb	Livre
3Kb	Programa B
10Kb	Livre
6Kb	Programa C
6Kb	Programa D
20Kb	Livre

(c)

5Kb	Livre
3Kb	Programa B
10Kb	Livre
6Kb	Programa C
6Kb	Programa D
4Kb	Programa E
16Kb	Livre

Estratégia First-fit:

(a)

5Kb	Programa A
3Kb	Programa B
6Kb	Programa D
4Kb	Livre
6Kb	Programa C
26Kb	Livre

(b)

5Kb	Livre
3Kb	Programa B
6Kb	Programa D
4Kb	Livre
6Kb	Programa C
26Kb	Livre

(c)

4Kb	Programa E
1Kb	Livre
3Kb	Programa B
6Kb	Programa D
4Kb	Livre
6Kb	Programa C
26Kb	Livre

7. O que é swapping e para que é utilizada esta técnica? Qual a importância da realocação dinâmica nesta técnica?

A técnica de swapping foi introduzida para contornar o problema da insuficiência de memória principal. Essa técnica é aplicada à gerência de memória para programas que esperam por memória livre para serem executados. Nesta situação, o sistema escolhe um processo residente, que é transferido da memória principal para a memória secundária (swap out), geralmente disco. Posteriormente, o processo é carregado de volta da memória secundária para a memória principal (swap in) e pode continuar sua execução como se nada tivesse ocorrido.
A realocação dinâmica permite que os programas possam ser retirados da memória principal para a memória secundária e trazidos novamente para a memória principal em qualquer posição.

8. Quais os benefícios oferecidos pela técnica de memória virtual? Como este conceito permite que um programa e seus dados ultrapassem os limites da memória principal?

Os principais benefícios da técnica de memória virtual são possibilitar que programas e dados sejam armazenados independente do tamanho da memória principal. Esta técnica sofisticada de gerência de memória combina as memórias principal e secundária, dando ao usuário a impressão de existir uma memória muito maior do que a MP. O conceito da memória virtual está em desvincular o endereçamento feito pelo programa dos endereços físicos da MP, assim, o programa não fica limitado ao tamanho da memória física disponível. O conjunto de endereços que um processo pode endereçar é chamado de espaço de endereçamento virtual. Analogamente o conjunto de endereços reais é chamado espaço de endereçamento real. O espaço de endereçamento virtual não possui relação direta com o espaço de endereçamento real. Sendo assim, um programa pode fazer referencia a endereços virtuais que estejam fora dos limites da MP.

9. Qual a principal diferença entre os sistemas que implementam paginação e segmentação?

A principal diferença entre os dois sistemas está relacionada a forma como o espaço de endereçamento virtual está dividido logicamente. Na paginação, o espaço de endereçamento está dividido em blocos com o mesmo número de endereços virtuais (páginas), enquanto que na segmentação o tamanho dos blocos pode variar (segmentos).

10. O que são tabelas de páginas e tabelas de segmentos? Explique para que servem os bits de validade, referencia e modificação encontrados nestas tabelas.

São tabelas de mapeamento, utilizadas no mecanismo de memória virtual, que possibilitam que endereços virtuais sejam traduzidos em endereços reais. O bit de validade é usado para indicar se a página ou o segmento em questão encontra-se na memória principal. O bit de modificação, indica 1 quando a página foi escrita na memória principal e 0, caso contrário. O bit de referência indica se a página foi referenciada na memória principal.

11. O que é um page fault, quando ocorre e quem controla a sua ocorrência? Como uma elevada taxa de page fault pode comprometer o sistema operacional?

O page fault ocorre todas as vezes que um processo faz referência a um endereço virtual pertencente a uma página virtual que não se encontra mapeada em uma página real, ou seja, não está, no momento, na memória principal. A ocorrência de um page fault é verificada através do bit de validade presente na entrada da tabela de páginas referente à página virtual. Uma elevada taxa de page fault pode comprometer o desempenho do sistema devido ao excessivo overhead de operações de E/S gerados pela paginação.

12. Compare as políticas de busca de páginas apresentadas.

Na 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.

Na paginação antecipada, o sistema carrega para a memória principal, além das páginas referenciadas, outras páginas que podem ou não ser necessárias ao processo ao longo do seu 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.

13. Quais as vantagens e desvantagens da alocação de páginas variável comparada à alocação fixa?

A política de alocação de páginas determina quantos quadros (frames) um processo pode manter na memória principal.
Na alocação fixa cada processo tem um número máximo de frames que pode ser usado durante a execução. É uma política simples de ser implementada mas não é sempre uma boa opção pois os processos possuem necessidades diferentes na alocação de memória. Se o número de frames alocado para um processo for pouco, uma página do próprio processo deve ser descartada para que outra seja carregada. O limite de frames deve ser definido no momento da criação do processo com base no tipo de aplicação que será executada. Se o limite for pequeno → o processo terá alto índice de page faults, por outro lado, se o limite for alto → poucos processos compartilharão a memória, podendo aumentar o swapping de processos

Na alocação variável o número máximo de frames alocado para um processo durante a sua execução pode variar. Um processo com alta taxa de paginação pode ter o limite máximo de frames aumentado para diminuir o alto índice de page faults. Já, um processo com baixa taxa de paginação pode ter seus frames alocados para outros processos. A alocação variável é mais flexível mas exige que o sistema operacional monitore constantemente o comportamento dos processos gerando maior overhead.

14. Um sistema com gerência de memória virtual por paginação possui tamanho de página com 512 posições, espaço de endereçamento virtual com 512 páginas e memória real com 10 frames.

    a) Como é o formato do endereço virtual deste sistema?
    O endereço virtual possui 9 bits para endereçar a tabela de páginas e 9 bits para o deslocamento dentro da página.

15. Um sistema operacional implementa gerência de memória virtual por paginação. Considere endereços virtuais com 16 bits, referenciados por um mesmo processo durante sua execução e sua tabela de páginas abaixo com no máximo 256 entradas, sendo que estão representadas apenas as páginas presentes na memória real. Indique para cada endereço virtual a seguir a página virtual em que o endereço se encontra e o respectivo deslocamento.

    a) $(2047)_{10}$
    Página virtual 7, deslocamento 255.
    b) $(1098)_{10}$
    Página virtual 4, deslocamento 74.
    c) $(451)_{10}$
    Página virtual 1, deslocamento 195.

16. Descreva os algoritmos de substituição de páginas OPT, FIFO, LRU, NRU e segunda chance, apresentando vantagens e desvantagens.

OPT: Possui a melhor taxa de falha de página dentre todos os algoritmos. Substitui a página que ficará mais tempo sem ser referenciada no futuro. Difícil saber de antemão a ordem em que as páginas serão referenciadas durante a execução. Como produz um resultado ótimo, é usado para estudos comparativos para se testar a eficiência de outros algoritmos.

FIFO: Quando uma página precisar ser substituída, a página mais antiga será escolhida para sair da memória. Fácil de entender e programar. É razoável pensar que uma página que já foi carregada para a MP há mais tempo, já tenha sido referenciada o suficiente, porém, isto nem sempre é verdade e seu desempenho pode não ser bom. Pode ser que a primeira página carregada seja a mais utilizada no sistema.

LRU: Substitui a página menos recentemente usada, ou seja aquela cuja última referência foi feita há mais tempo. Se for considerado o princípio da localidade, é provável que uma página que não tenha sido referenciada recentemente não seja usada novamente em um futuro próximo. É uma boa estratégia, porém possui maior sobrecarga pois deve manter atualizado o momento do último acesso a cada página (usa um timestamp).

NRU: Usa um bit de referência para indicar se a página foi usada recentemente ou não. Inicialmente, quando uma página é trazida para a memória, o seu bit de referência é zero. Quando uma página que já está na memória e é

referenciada o seu bit muda para 1. O algoritmo substitui a página que não foi usada recentemente. Pode ser mais eficiente ao se usar bits de referência adicionais.

Segunda Chance: Cria uma modificação no algoritmo FIFO, com o objetivo de não substituir uma página intensamente usada. O bit de referência da página mais antiga é inspecionado: Se o bit = 0, então a substituição prossegue, se o bit = 1, então a página recebe uma segunda chance, seu bit de referência é zerado e sua hora de chegada é reiniciada (vai pro fim da fila). Se uma página é usada com frequência, e seu bit está sempre 1, então ela não será substituída

17. Considere um sistema de memória virtual que implemente paginação, onde o limite de frames por processo é igual a três. Descreva para os itens abaixo, onde é apresentada uma sequência de referências à páginas pelo processo, o número total de page faults para as estratégias de realocação de páginas FIFO e LRU. Indique qual a mais eficaz para cada item.

    i. 1 / 2 / 3 / 1 / 4 / 2 / 5 / 3 / 4 / 3

FIFO

1	*2*	*3*	*1*	*4*	*2*	*5*	*3*	*4*	*3*
1	*1*	*1*		*4*		*4*			
	2	*2*		*2*		*5*			
		3		*3*		*3*			

LRU

1	*2*	*3*	*1*	*4*	*2*	*5*	*3*	*4*	*3*
1	*1*	*1*		*1*	*1*	*5*	*5*	*5*	
	2	*2*		*4*	*4*	*4*	*3*	*3*	
		3		*3*	*2*	*2*	*2*	*4*	

FIFO = Total PF = 5 (melhor política)
LRU = Total PF = 8

    ii. 1 / 2 / 3 / 1 / 4 / 1 / 3 / 2 / 3 / 3

FIFO

1	*2*	*3*	*1*	*4*	*1*	*3*	*2*	*3*	*3*
1	*1*	*1*		*4*	*4*		*4*	*3*	
	2	*2*		*2*	*1*		*1*	*1*	
		3		*3*	*3*		*2*	*2*	

LRU

1	*2*	*3*	*1*	*4*	*1*	*3*	*2*	*3*	*3*
1	*1*	*1*		*1*			*1*		
	2	*2*		*4*			*2*		
		3		*3*			*3*		

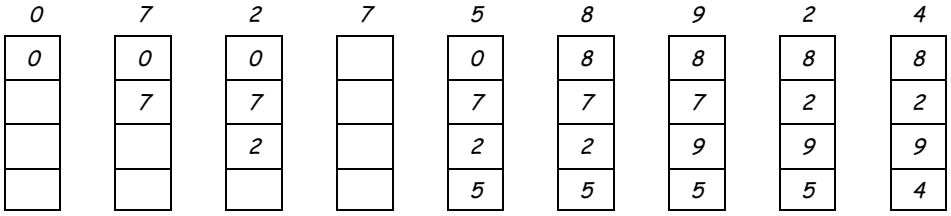
FIFO = Total PF = 7
LRU = Total PF = 5 (melhor política)

18. Em um computador, o endereço virtual é de 16 bits e as páginas têm tamanho de 2Kb endereços. O WSL (Working Set List) de um processo qualquer é de quatro páginas. Inicialmente,

nenhuma página está na memória principal. Um programa faz referência a endereços virtuais situados nas páginas 0, 7, 2, 7, 5, 8, 9, 2 e 4, nesta ordem.

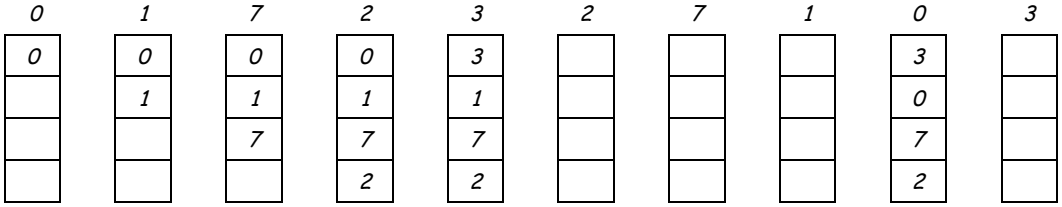
i. Quantos bits do endereço virtual destinam-se ao número da página? E ao deslocamento?
5 bits para o número da página e 11 bits para o deslocamento (Tamanho da página 2kb = 2048 = 2^{11})

ii. Ilustre o comportamento da política de substituição LRU, mostrando, a cada referência, quais páginas estão em memória, os page faults causados e as páginas escolhidas para saírem da memória.

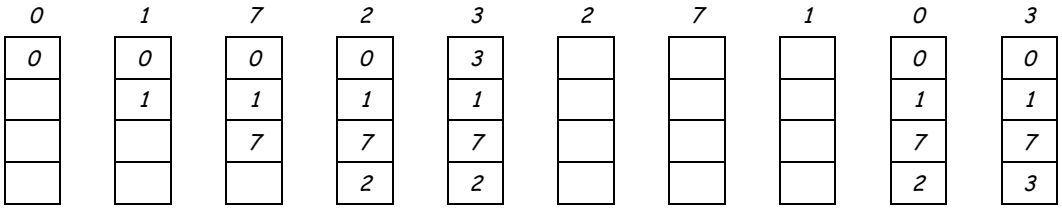


19. Considere um processo com limite de páginas reais igual a quatro e um sistema que implemente a política de substituição FIFO. Quantos page faults ocorrerão considerando que as páginas virtuais são referenciadas na seguinte ordem: 0 1 7 2 3 2 7 1 0 3. Repita o problema utilizando a política LRU.

FIFO gera 6 page faults



LRU gera 7 page faults



20. Explique porque páginas pequenas podem aumentar a taxa de paginação.

Existe uma relação entre o tamanho da página e o número de operações de E/S que o sistema deverá executar para carregar as páginas da memória secundária para a memória principal. Quanto menor o tamanho da página, maior o número de operações de E/S, aumentando a taxa de paginação. Por outro lado, páginas pequenas oferecem menor fragmentação interna.

21. Existe fragmentação em sistemas que implementam gerência de memória virtual? Se existe, que tipo de fragmentação é encontrado em sistemas paginados? Que tipo de fragmentação é encontrado em sistemas com segmentação?

O problema da fragmentação existe tanto na gerência de memória virtual por paginação quanto na por segmentação. A fragmentação interna ocorre na memória virtual por paginação na última página, caso não seja totalmente ocupada. A fragmentação externa ocorre na memória virtual por segmentação em função dos espaços livres deixados entre segmentos alocados na memória principal.

O que é o thrashing em sistemas que implementam memória virtual?

Thrashing é consequência da excessiva paginação/segmentação em sistemas que implementam memória virtual, levando o sistema a dedicar mais tempo com operações relacionadas à gerência da memória do que no processamento das aplicações dos usuários.

22. Como arquivos podem ser estruturados?

Existem 3 maneiras: sequência de bytes, sequência de registros e árvores. A forma mais simples de organização de arquivos é através de uma sequência não-estruturada de bytes, na qual o sistema de arquivos não impõe nenhuma estrutura lógica para os dados. Qualquer significado e organização deve ser imposto pelos programas dos usuários. Oferece máxima flexibilidade.
Na sequência de registros, o arquivo é visto como uma sequência de registros de tamanho fixo. A ideia central é que a operação de leitura retorna um registro e a operação de escrita sobrepõe ou anexa um registro. Antigamente, na época do cartão perfurado de 80 colunas, muitos sistemas de arquivos trabalhavam com sequência de registros de 80 caracteres, correspondente às imagens dos cartões. Sistemas de propósito geral não usam mais este tipo de modelo.
Na estrutura em árvore um arquivo é constituído por uma árvore de registros, que podem possuir tamanhos diferentes. Cada registro contém um campo chave para que a busca seja mais rápida. Pouco usada.

23. Quais as diferentes formas de implementação de uma estrutura de diretórios?

Estrutura de diretório de nível único, com dois níveis e em árvore.

24. O que é alocação contígua de blocos e quais benefícios a desfragmentação pode proporcionar quando esta técnica é utilizada?

A alocação contígua consiste em armazenar um arquivo em blocos sequencialmente dispostos no disco. É uma estratégia de simples implementação e de bom desempenho. Porém, com o tempo, quando os arquivos vão sendo removidos são deixadas lacunas de blocos livres. A desfragmentação pode solucionar o problema da fragmentação reorganizando todos os arquivos no disco de maneira que só exista um único segmento de blocos livres.

25. Em uma aplicação concorrente que controla saldo bancário em contas correntes, dois processos compartilham uma região de memória onde estão armazenados os saldos dos clientes A e B. Os processos executam, concorrentemente os seguintes passos:

Processo 1 (Cliente A)

```
/* saque em A */
1a) x := saldo_A;
1b) x := x - 200;
1c) saldo_A := x;

/* deposito em B */
1d) x := saldo_B;
1e) x := x + 100;
1f) saldo_B := x;
```

Processo 2 (Cliente B)

```
/*saque em A */
2a) y := saldo_A;
2b) y := y - 100;
2c) saldo_A := y;

/* deposito em B */
2d) y := saldo_B;
2e) y := y + 200;
2f) saldo_B := y;
```

Supondo que os valores dos saldos de A e B sejam, respectivamente, 500 e 900, antes dos processos executarem, pede-se:

a) Quais os valores corretos para os saldos dos clientes A e B após a execução dos processos?

Cliente A = 200 e Cliente B = 1.200

b) Quais os valores finais dos saldos dos clientes se a sequência de execução das operações for: 1a, 2a, 1b, 2b, 1c, 2c, 1d, 2d, 1e, 2e, 1f, 2f?

Cliente A = 400 e Cliente B = 1.100

c) Utilizando semáforos, proponha uma solução que garanta a integridade dos saldos e permita o maior compartilhamento possível dos recursos entre os processos, não esquecendo a especificação da inicialização dos semáforos.

Processo 1 (Cliente A)

```
/* saque em A */
DOWN(S1);
x := saldo_A;
x := x - 200;
saldo_A := x;
UP(S1);

/* deposito em B */
DOWN(S2);
x := saldo_B;
x := x + 100;
saldo_B := x;
UP(S2);
```

Processo 2 (Cliente B)

```
/*saque em A */
DOWN(S1);
y := saldo_A;
y := y - 100;
saldo_A := y;
UP(S1);

/* deposito em B */
DOWN(S2);
y := saldo_B;
y := y + 200;
saldo_B := y;
UP(S2);
```