DOSSIÊ DE SISTEMA OPERACIONAIS 2002 – 2007

1) Defina a diferença entre escalonamento preemptivo e não preemptivo. Por que o escalonamento preemptivo é geralmente usando em um sistema de computação.

A diferença entre estes dois tipos de escalonamento está no momento em que a decisão de escalonamento é tomada. No escalonamento não preemptivo, uma vez que um processo ganhe a CPU ele não será compulsoriamente suspenso. Ele executará até que libere a CPU voluntariamente (termine de executar) ou até que seja bloqueado (à espera de E/S ou de outro processo). Já no escalonamento preemptivo, um processo que ganha a CPU executa durante um tempo máximo fixado. Se ao término deste tempo o processo não tiver terminado sua execução, ele será interrompido e outro processo será escalonado. A cada interrupção de relógio o controle da CPU é devolvido ao escalonador. O escalonamento ocorrerá também se o processo terminar ou for bloqueado antes do término do período estabelecido.

A principal desvantagem dos algoritmos não preemptivos é que quando um processo ganha a CPU, nenhum outro processo pode realizar suas computações até que o processo atualmente na CPU termine. Se este processo entrar em um laço infinito o sistema irá ficar esperando eternamente um término que nunca ocorrerá e nenhum outro processo poderá ganhar a CPU. Essa desvantagem é o principal motivo pelo qual o escalonamento preemptivo é geralmente usado em sistemas de computação.

2) Explique os mecanismos introduzidos em so's na década de 60 que permitiram que operações de E/S pudessem ser realizadas em paralelo a operações da UCP.

Na década de 60 percebeu-se que a maioria dos *jobs* podiam ser enquadrados em duas categorias: aqueles que passavam a maior parte do tempo realizando computações na UCP (*CPU bound*) e aqueles que passavam a maior parte do tempo realizando operações de E/S (*IO bound*). Para evitar que a UCP e os dispositivos de E/S ficassem ociosos por longos períodos de tempo foi desenvolvido o mecanismo da **multiprogramação**. Este mecanismo consistia em dividir a memória em várias partes, cada uma contendo um *job* diferente. Deste modo enquanto um *job* esperava pela realização de uma operação de E/S (normalmente demorada) outro *job* residente na memória poderia ganhar a UCP para realizar suas computações, minimizando assim o tempo ocioso da UCP e dos dispositivos de E/S.

A introdução de interrupções no UNIVAC 1103 em 1957 e de canais autônomos de E/S, no IBM 7094 em 1959 foram inovações importantes para aumentar a eficiência de processamento: agora Entrada e Saída poderiam ser feitas em paralelo com cálculos pelo processado central, usando a técnica de "armazenamento" (buffering). A introdução de interrupções e de canais autônomos de Entrada/Saída possibilitou o uso efetivo da multiprogramação, isto é, mais de um programa residente simultaneamente na memória e cada um deles tendo a oportunidade de prosseguir durante os intervalos de espera pelo término de E/S, dos outros. A técnica de rolamento pode ser implementada agora de modo automático em um único sistema, eliminando o carregamento manual de fitas: um programa especial SPOOLER (Simultaneous Peripheral Operation On Line) continuamente lê cartões sempre que o programa principal se torna inativo à espera de E/S. Sistemas com rolamento automático foram implantados por volta de 1960

3) Implemente o problema dos leitores e escritores utilizando semáforos. Explique de forma sucinta sua implementação.

```
escritor(){
   while(TRUE){
      info = produzinformacao();
      P(escreveBloco);
         //regiao critica
               escrever(r, info);
      V(escreveBloco);
   }
}
```

```
leitor(){
   while(TRUE){
      //regiao não critica
      info = ler(r);
      P(mutex);
         leitores++;
         if(leitores == 1)
            P(escreveBloco);
      V(mutex);
      P(mutex);
         leitores--;
         if(leitores == 0)
            V(escreveBloco);
      V(mutex);
   }
}
```

recurso r; //recurso compartilhado
semaforo mutex = 1; //usado para proteger a variavel leitores
int leitores = 0; // numero de leitores no buffer
semaforo escreveBloco = 1; //usado para proteger o recurso r

Nesta implementação podemos ter vários leitores acessando o buffer. Sempre que um escritor chegar, ele deve esperar até que o número de leitores seja igual a zero e isso pode gerar uma postergação indefinida no acesso do escritor à região crítica.

4) Defina sincronização condicional e sincronização por exclusão mutua dando exemplos de cada uma. O que são primitivas de exclusão mutua?

A sincronização por exclusão mútua é aquela que garante as regiões críticas de processos concorrentes nunca sejam executadas ao mesmo tempo. A abordagem condicional visa garantir que um processo espere (se for necessário) até que uma dada condição seja verdadeira. No problema dos produtores e consumidores (que possuem um buffer compartilhado), por exemplo, a exclusão mútua é usada para garantir que um produtor e um consumidor não acessem o buffer ao mesmo tempo e a exclusão condicional garante que o consumidor só ira consumir um item do buffer se este não estiver vazio.

As primitivas de exclusão mútua são instruções que devem ser preferencialmente atômicas e que servem para controlar o acesso aos recursos compartilhados. Antes de entrar em sua região crítica, um processo deve chamar uma primitiva de exclusão mútua que verifica se outro processo encontra-se na região crítica, bloqueando o processo que a chamou em caso afirmativo. Uma outra primitiva deve ser chamada ao término do processamento da região crítica para que outros processos presos na primeira primitiva possam continuar.

5) Explique o modo de funcionamento dos seguintes tipos de sistemas operacionais

a) SO de lote

Usado em máquinas de grande porte, a idéia do SO de lote é gravar um grande número de *jobs* em uma fita magnética. O papel do SO é ler um a um todos os programas do lote (gravados na fita) e executá-los gravando suas respectivas saídas em outra fita que, depois do lote ter sido processado, era levada para outra máquina que imprimia seu conteúdo *off-line*.

b) So de tempo repartido

Sistema onde cada usuário conecta-se por meio de um terminal *on-line* e o tempo da CPU é alocado de acordo com as necessidades de cada um. O tempo repartido é uma variante da multiprogramação e permite a vários usuários terem acesso a uma mesma máquina ao mesmo tempo

c) So de tempo real

São sistemas que têm o tempo como parâmetro fundamental, suas ações devem ser executadas dentro de rígidos prazos de tempo. Se um descumprimento ocasional de um prazo for aceitável o sistema é caracterizado como não crítico. Caso contrário, é considerado um sistema de tempo real crítico.

d) So de rede

Um SO de rede, os usuários sabem da existência de outras máquinas, podendo se conectar a elas para gravar arquivos nelas ou ler dados das mesmas. Cada máquina tem seu próprio SO local. Um SO de rede pouco difere de um SO voltado para um único processador mas precisa de uma interface de rede e um software de baixo nível para controlá-la.

e) So distribuído

Um SO distribuído é aquele em que, aos olhos do usuário é idêntico a um SO voltado para um processador, mas que internamente realiza suas computações em vários processadores e grava e lê dados de discos que podem estar em diversas máquinas diferentes. O usuário não precisa saber onde seus programas são executados ou onde seus arquivos estão armazenados pois o SO cuida de tudo isso de maneira eficiente.

6) Ao executar algumas instruções da linguagem de maquina, um processo pode vir a interferir na execução de um outro processo/ descreva os mecanismos utilizados pelo sistema operacional para evitar que isto ocorra.

Com a introdução da multiprogramação surgiu uma política de instruções privilegiadas e chamadas ao sistema, para evitar as interferências e destruição de outro processo devido a códigos maliciosos . Com esta política para executar uma instrução privilegiada o usuário é obrigado a realizar uma chamada ao sistema que irá chavear entre o modo usuário e o modo nucleo através da instrução trap.

7) **O que é preempção? Para que é utilizada? Quais as desvantagens de seu uso?**

A preempção é a interrupção forçada de um processo para que outro ganhe a CPU ao término de um intervalo de tempo (normalmente determinado pelas interrupções de relógio). É utilizada em sistemas de multiprogramação para garantir que todos os processos progridam uniformemente e impedir que um processo malicioso ou mal codificado ganhe a CPU e não a libere para outros processos. A principal desvantagem da preempção é o overhead gerado pela troca de contexto (que deve ser feita toda vez que um processo for interrompido antes de terminar a execução).

8) Explique como funciona a instrução TST e como ela pode ser utilizada pra resolver o problema de sincronização por exclusão mútua entre processos concorrentes.

A instrução TST é uma instrução de máquina que copia o valor de uma variável de impedimento (lock) da memória para um registrador e coloca na variável lock um valor que indica que o recurso compartilhado vigiado pela variável *lock* está em uso. Como no código a seguir:

```
TST Registrador, Lock     ; copia o valor de lock no registrador register e seta Lock em 1
```

Como a instrução TST é atômica temos a garantia de que, uma vez iniciada, sua execução não será interrompida. Isto torna possível implementar a exclusão mútua utilizando a instrução TST com se segue.

Se a variável *lock* está em 1 o recurso compartilhado está em uso, se está em 0 o RC não está em uso. quando de quiser entrar na região crítica é só fazer ma chamada a uma primitiva que chama TST e verifica o valor anteror de *lock* , se for 1 então bloqueia, senão continua:

```
entra_regiao:
        TST Register, Lock
        CMP Register, 0
        JZE ok
        CALL bloqueia_processo
ok:     RET

sai_região:
        MOV Lock, 0
        RET
```

9) Ilustre através do uso de monitores como implementar um alocador de recursos.

```
monitor alocadorderecursos {
      var boolean recursoEmUso = false;
      condition recursoLivre;

      obtemRecurso{
            if ( recursoEmUso )
                  recursoLivre.wait();
            recursoEmUso = true;
      }

      liberaRecurso{
            recursoEmUso = false;
            recursoLivre.signal();
      }
}
```

10) Para que servem os estados supervisor (kernel) e usuário ? explique seu mecanismo de funcionamento.

Servem para proteger algumas instruções privilegiadas da CPU. Estas instruções só podem ser executadas em modo supervisor. Um bit do registrador PSW controla em qual estado a CPU se encontra. O SO é o único software que roda em modo supervisor, todos os outros rodam em modo usuário. Se um aplicativo comum necessita realizar uma instrução privilegiada deve fazê-lo por meio de chamadas ao sistema. As chamadas ao sistema passam o controle para o SO que, primeiramente, executa uma instrução TRAP para mudar o estado da CPU para o modo supervisor e então executar as instruções privilegiadas. Ao término da execução destas instruções o SO retorna para o modo usuário e devolve o controle ao processo que realizou a chamada ao sistema.

11) Cinco processos (A a E) chegam a um sistema de computação com tempos estimados de execução de 10, 6, 2, 4 e 8 unidades de tempo, respectivamente. Suas prioridades são 3, 5, 2, 1 e 4, também respectivamente. Determine o tempo médio de espera e o tempo desde a submissão até a conclusão de cada um dos processos levando-se em consideração as seguinte políticas de escalonamento:

a) First-come, first served

Ordem de execução: 10 – 6 – 2 – 4 – 8
Tempo médio de espera:

(0 + 10 + 16 + 18+ 22)/5
= 13,2
Tempo de submissão até a conclusão:
(10 + 16 + 18 + 22 + 30)/5
= 19,2

b) Shortest job first

Ordem de execução: 2 – 4 – 6 – 8 – 10
Tempo médio de espera:
(0 + 2 + 6 +12 + 20)/5
= 8
Tempo de submissão até a conclusão:
(2 + 6 + 12 + 20 + 30)/5
= 14

c) Escalonamento por prioridade

Ordem de execução: 6 – 8 – 10 – 2 – 4
Tempo médio de espera:
(0 + 6 + 14 + 24 + 26)/5
= 14
Tempo de submissão até a conclusão:
(6 + 14 + 24 + 26 + 30)/5
= 20

12) Considere que exista apenas um processo produtor de mensagens e *n* processos consumidores. Escreva o código do processo produtor e dos processos consumidores utilizando as primitivas de passagem de mensagens **send** e **receive**. Considere na sua implementação que estas operações possuem a seguinte assinatura:

a) Send(destino, msg);

b) Receive(origem, msg);

Explique na sua solução se foram utilizadas primitivas síncronas ou assíncronas, descrevendo o motivo de sua escolha.

```
produtor{
   buffer itens[N];
   num_items = 0;
   mensagem m1, m2;
   while(true){
      if(num_items < N){
        itens[num_tems++] =  produzItem();
      }
      if(num_items > 0){
        receive(&m1);
        constroi_mensagem(&m2, itens[num_itens-–];
        send(m1.sender(), &m2);
      }

   }
}



consumidor{
```

```
    while(true){
      send(prod_id, &m);
      receive(prod_id,&m);
      item = extrai_item(&m);
      consome_item(item);
    }
}
```

As primitivas são síncronas pois o produtor só pode enviar uma mensagem com um item depois de receber uma solicitação de um consumidor e cada consumidor só pode consumir um item depois de recebê-lo do produtor.

13) **Explique o que é um sistema de operacional, quais as funções que ele desempenha em um sistema de computação e seus principais componentes.**

Um sistema operacional é um conjunto de programas que tornam um hardware possível de ser utilizado. Ele deve fornecer ao usuário uma interface com o hardware mais simples que o hardware sozinho tornando a máquina mais flexível e fácil de programar. Além disso o sistema operacional deve gerenciar os recursos da máquina como memória dispositivos de E/S, dados, processos, CPUs, possibilitando um uso controlado e eficiente de tais recursos.

14) O que é escalonamento preemptivo? Utilizando o diagrama de estados de processos, indique em que situações o escalonamento preemptivo ocorre.

O escalonamento preemptivo é o escalonamento que interrompe o processo em execução na CPU ao final de um período de tempo para que outros processos possam ganhar a CPU. Em outras palavras o escalonamento preemptivo é aquele que ocorre não só quando o processo na CPU termina ou é bloqueado mas também ao término de um determinado período de tempo (indicado por uma interrupção de relógio).

15) O que é troca de contexto e para que serve? Como corre a troca de contexto quais as vantagens e desvantagens de sua utilização?

A troca de contexto é a alternância de processos na CPU. Na troca de contexto o conteúdo dos registradores, mapas de memória e outras informações relevantes do processo atualmente na CPU são salvos (em disco ou na própria memória) e as informações referentes ao próximo processo a ganhar a CPU são carregadas. A principal vantagem da troca de contexto é que processos concorrentes progridam uniformemente em sistemas monoprocessados. A desvantagem é que ela gera um *overhead*, já que algum tempo é perdido na troca de contexto.

16) O que é sincronização por exclusão mutua e sincronização condicional?

Explique porque a primitiva de entrada na região critica abaixo não funciona.

Idéia básica: usar uma variável *x* como chave para o acesso a RC

a) X=0 indica que a RC esta em uso

b) X=1 indica que a RC esta livre.

```
Laço LDA ; carrega AC c/ valor de x
BZ laço; se zero volta ao teste
CLR x ; zera x, entra na RC
```

Não funciona porque esta rotina pode ser interrompida pelo escalonador de processos. Um exemplo: Um processo A quer entrar na RC e chama a rotina acima. O processo A copia o conteúdo de X para uma variável local e verifica que a RC está livre (X=1) e é interrompido pelo escalonador antes da instrução CLR x. Um processo B é então escalonado e tenta também entrar na região crítica. Ao verificar x o processo B descobrirá que a região crítica está livre e irá zerar x e entrar na região crítica. Se depois de B zerar x A for escalonado ele irá para a próxima instrução que executaria, no caso CLR x, zeraria x e entraria na região crítica. Desta forma teríamos dois processos simultaneamente na RC.

17) Explique o que é a instrução TST e como ela pode ser utilizada para a implementação de exclusão mutua.

18) Explique como implementar um monitor para controlar o acesso a um recurso compartilhado.

Um Monitor deve ser um recurso fornecido pela linguagem de programação, para que, desse modo, o compilador possa tratar de modo especial os métodos do monitor. Um monitor deve constar de variáveis internas, chamadas variáveis permanentes, que representem o estado do recurso compartilhado e de métodos que executem operações sobre o mesmo (como , por exemplo, leitura e escrita). Portanto, para se alterar o estado de um recurso o processo deve chamar os procedimentos do monitor. A sincronização por exclusão mútua é garantida pelo fato de que, em um dado instante apenas um processo poderá estar executando métodos do monitor, daí a necessidade de que o compilador reconheça os monitores como estruturas especiais. Já a sincronização condicional deve ser feita explicitamente através do uso de variáveis condicionais, que podem liberar ou bloquear processos através das operações *wait* e *signal.*

19) Para impedir a ocorrência de bloqueios perpétuos, o implementador de um SO decidiu impor uma ordenação Linear aos recursos do sistema. Após alocar um recurso, os processo somente poderão solicitar recurso que possuam numeração superior ao recurso já alocado. Esta solução impede a ocorrência de bloqueios perpétuos? Se sim, explique porque a solução funciona, caso contrario mostre um exemplo no qual ela falha.

Sim. Um deadlock ocorreria se um processo *A* que tivesse alocado um recurso *i* e um processo *B* tivesse alocado um recurso *j* e *A* requisitasse *j* ou *B* requisitasse *i.* Entretanto, essa situação não pode ocorrer, pois se $i < j$ *B* não pode requisitar *i* e se $j < i$ *A* não pode requisitar *j*. Dessa forma o método da ordenação linear dos recursos impede a ocorrência de uma corrente circular, uma das condições

para que haja um deadlock. Este método não é utilizado na prática porque pode ser impossível encontrar uma ordem para os recursos que satisfaça todas as aplicações. Além do mais, ao adicionar um novo recurso ao sistema, provavelmente todas as aplicações e talvez até o próprio sistema teria de ser reescrito.

20) Quais são as principais funções do sistema de gerenciamento de memória?

Saber a cada instante o estado de utilização da memória.
Determinar a técnica e a política de alocação.
Determinar a técnica e a política de recuperação da memória.

21) Explique como funciona a tradução de um endereço virtual em real nas seguintes políticas:

a) Alocação paginada

b) Alocação segmentada com paginação.

22) Explique como é implementado o algoritmo de substituição de paginas LRU em sistemas com paginação.

23) Explique a política de gerenciamento de memória que utiliza paginação com segmentação. Descreva as tabelas utilizadas para implementar e como se relacionam.

24) Explique como a política de substituição de pagina LRU pode ser implementada.

25) Explique como pode ser implementada a relocação de programas na memória considerando a política de gerenciamento alocação dinâmica particionada.

Antes de tudo, é necessário que o compilador produza código relocável (código que possa ser carregado a partir de qualquer posição física na memória). A relocação, no contexto da política de alocação particionada é realizada sempre que um endereço é fornecido ao sistema de memória. Para implementá-la basta interceptar esse endereço e somar a ele o valor do ponto de carga do programa. Este esquema pode ser implementado utilizando-se um registrador base. Toda vez que um processo ganhar a CPU, o registrador base será carregado com o endereço físico do ponto de carga (início da partição da memória onde o processo está alocado). Todo endereço gerado pelo processo (endereço lógico) é transformado em endereço físico adicionando-se a ele o valor do registrador base. Para impedir que o processo tente

acessar uma região da memória fora daquela a ele alocada pode-se utilizar um registrador limite com a último endereço físico da partição de memória alocada ao processo.

26) Explique a política de gerenciamento de memória que utiliza paginação. Não deixe de descrever as tabelas utilizadas para implementação da política e como elas se relacionam. Como pode ser implementando o uso de memória virtual utilizando essa política.

A política de paginação da memória divide o espaço de endereçamento lógico em partes chamadas páginas e o espaço de endereçamento físico em partes chamadas frames (molduras de páginas), ambas de mesmo tamanho. São então utilizadas dois tipos de tabelas:

- Uma **tabela de páginas** para cada processo, contendo uma entrada para cada página do processo (número da página, frame onde foi carregada).
- Uma **tabela de frames**, única no sistema, contendo uma entrada para cada frame existente na memória física e seu estado de utilização (disponível ou não e, eventualmente, quem o está utilizando).

Assim sendo, toda vez que um processo referenciar uma posição da memória, o fará através de um endereço lógico, formado pelo número da página e o offset do endereço. O mecanismo responsável pelo gerenciamento de memória irá procurar pelo frame no qual a página referenciada está carregada, e traduzirá o endereço lógico para o físico, que será formado pelo número do frame e o mesmo offset presente no endereço lógico.

O uso de memória virtual pode ser implementado adicionando-se na tabela de páginas uma informação sobre o estado da página (presente ou não na memória). Como o objetivo da memória virtual é fazer com que programas maiores que a memória disponível possam ser executados, teríamos mais páginas do que frames. Aquelas páginas que não estivessem carregadas na memória (não estão em nenhum frame) ficariam armazenadas em disco. Sempre que uma página que não estiver na memória for referenciada, uma interrupção por falta de página será gerada e uma página da memória será escolhida para ser carregada no disco. A página referenciada ocupará então o frame que foi "liberado".

27) Em que circunstância uma interrupção por falta de pagina é gerada? Como o evento que gera esta interrupção é detectado? Que ações são realizadas pelo SO face a ocorrência de uma interrupção desse tipo?

28) Descreva como pode ser implementado o suporte a múltiplos relógios virtuais para que uma chamada ao sistema do tipo ALARM possa ser disponibilizada pelo SO.

O suporte a múltiplos relógios é implementado usando uma lista encadeada contendo todas as solicitações para o relógio, classificada por tempo. Cada entrada na lista informa quantos tics do relógio a partir da entrada anterior o sistema deve esperar para sinalizar o processo. A cada tic a variável próximo sinal é decrementada, e quando a mesma chegar a zero, o sinal correspondente a primeira entrada na lista é gerado e a mesma é removida. Então próximo sinal é atualizada com a próxima entrada na cabeça da lista.

29) Suponha que um bloco de disco em um sistema Unix consiga armazenar 2048 endereços de disco de 32bits. Qual o tamanho máximo de arquivo que este sistema suporta se utilizarmos apenas blocos de índice primários? E se utilizarmos locos de índices secundários?

30) Explique a técnica de scan N passos utlizada para otimizar o tempo de Seek.

O braço do disco caminha para cima e para baixo como no scan, só que com uma diferença, quando novas requizições de E/S chegam e o braço está caminhando em uma direção, estas serão armazenadas e posteriormente realizadas. Isto elimina a postergação indefinida, melhorando assim o tempo de seek, comparado com outras técnicas de escalonamento em disco como a SSTF(shortest seek time first),

31) Dispositivos de entrada e saída são divididos em dois grupos. Que grupos são esses, como eles funcionam? De exemplos.

São os dispositivos de blocos e os a caracter. Os primeiros armazenam a informação em blocos de tamanho fixo, cada um com seu endereço próprio. Com uma propriedade de poder escrever em cada bloco de forma independente, exemplo os discos. Os dispositivos a caracter, enviam ou recebem uma cadeia de caracteres sem nenhuma estrutura de bloco. Não são endereçáveis e não possuem seek, exemplo mouse, teminais, impressora...

32) Explique como funciona a técnica de polling?

Dois bits constantes de um gerenciador de status são utilizados para coordenar a comunicação entre a controladora e o host. O bit busy é setado pela controladora quando está operando o dispositivo e é zerado por ela quando estiver pronta para aceitar um novo comando de E/S. O host sinaliza que enviou um comando de E/S setando o bit command-ready

33) Explique como é organizado o software de entrada e saída do SO. Escrevendo suas camadas e a finalidade de cada uma.

Software do utilizador – Solicita E/S; Software independente do dispositivo – nomeia, protege, bloqueia, armazena ou aloca ; Driver do dispositivo – seta os registradores e verifica seus status, passando o controle para o hardware; Rotinas de Interrupção - "acorda" o dispositivo quando a operação de E/S for concluída pelo hardware; Hardware – efetua a operação de E/S.

34) Explique a técnica de acesso a memória (DMA) e como funciona.

A técnica de acesso direto a memória é utilizada para não ocorrer interferência da UCP toda vez que dados forem transferidos entre a memória e um dispositivo, a não ser no início e no final da transferência. Funcionando da seguinte forma: Quando a UCP deseja ler ou gravar algum bloco de dados em um dispositivo de E/S, a UCP envia um comando para a DMA, contendo: o tipo de tarefa de E/S a ser efetuada, o endereço do dispositivo desejado, o endereço inicial da memória onde o bloco de dados será lido ou gravado e a quantidade de bytes a serem lidos ou gravados(tamanho do bloco). Durante toda essa transferência a UCP fica liberada para para fazer outras tarefas, o controlador DMA ao terminar a operação de transferência envia um sinal(interrupção) à UCP, avisando que a operação foi concluida e os dados já estão disponíveis. A área utilizada pelo controlador DMA é chamada de buffer.

35) Explique como funciona as técnicas de armazenamento dos blocos de arquivos chamados de alocação continua e ligação de blocos. Faça uma analise comparativa entre elas.

36) Considere a seguinte string de referencia a paginas: 1,2,3,4,2,1,5,6,2,1,2,3,7,6,2,1,2,3,6. quantas interrupções por falta de pagina ocorrem qando são utilizados os algoritmos de substituição de pagina FIFO e LRU, assumindo-se que existem dois, quatro e seis blocos de memória disponíveis? Lembre-se que inicialmente todos os blocos estão vazios. Portanto, as paginas custarão inicialmente uma interrupção por falta de pagina para serem trazidas à memória.

37) Por que motivo um processo não consegue acessar uma região de memória que não lhe pertence quando a alocação pagina é utilizada? Como o sistema opreacional poderia permitir ao processo o acesso a uma região de memória? Porque o sistema operacional concederia este direito e quando não concederia?

38) Por que a segmentação e a paginação são as vezes comninadas em um único esquema de gerenciamento de memória?

39) Explique as ações tomadas pelo sistema operacional quando o processo realiza chamadas para a abertura de um arquivo(operações ***Open()*** e ***Close*()***)

É adicionado a uma tabela de arquivos dados iniciais acerca do arquivo,como o posicionamento de um ponteiro no inicio do arquivo(endereçamento de leitura e escrita), é incrementado um contador indicando quantos processo estão usando o arquivo e é indicado a localização do arquivo em disco. Para abrir o arquivo é passado seu nome, tipo, localização na memória, tamanho, proteção, tempo(data de criação e outros) e identificação do usuário. Quando um arquivo é fechado(primitiva close()), é liberado todo espaço alocado para esse arquivo, limpando seus dados da tabela de arquivos abertos.

40) Descreva como ocorre uma operação de E/S quando a técnica de polling é utilizada. Em que circunstância o sistema operacional a utiliza ?!

Primeiramente o host seta o bit command-ready indicando que uma operação de E/S foi enviada, a controladora seta o bit busy, avisando que está se operando com o dispositivo, por fim quando a mesma estiver terminado com o dispositivo o bit command-ready e o bit busy são zerados indicando que o dispositivo está livre.

41) Explique como funciona a tradução de endereço virtual em real nas seguinte políticas de gerenciamento de memória

a) Alocação particionada dinâmica relocável

b) Alocação segmentada com paginação

c) Alocação por paginação

42) Quais as vantagens de alocação segmentada com relação a alocação paginada? Explique o principal problema de alocação segmentada e porque ele ocorre.

43) Explique como pode ser implementado o controle de varias chamadas a função ALARM, de forma a permitir que cada processo seja executado no horário indicado.

44) Em um serviço de diretório que organiza os arquivos de forma hierárquica, quando um diretório é removido, todos os arquivos e diretórios subseqüentes são também removidos. Explique porque esta operação pode causar problemas em serviços de diretórios que organizam os arquivos em grafos. Como podemos contornar o problema ?

Resp:

Serviços de diretórios organizados em grafos podem ter um arquivo referenciado por mais de um pai, então não podemos apagar todos os filhos de um arquivo a partir do mesmo sem antes verificar se algum arquivo filho possue mais de um pai e se o mesmo já foi apagado para poder o filho ser apagado. Uma solução proposta é adicionar um contador para cada arquivo, sendo que a cada ponteiro criado incrementa o contador e a cada ponteiro eliminado na hierarquia decrementa o contador, sendo que quando o contador chega a zero, podemos apagar o arquivo.

45) Explique as princiapais funcionalidades dos seguinte componente s de software constantes do SO que controlam a E/S em um sistema de computação.
   a) Software independente de dispositivo
   b) Drives de dispositivos
   c) Manipuladores de interrupção.

46) Descreva as técnicas para mapeamento dos blocos utilizados por arquivos em um volume de armazenamento, conhecida com blocos de índices e mapa de arquivos.

47) O que são domínios de proteção e como eles podem ser representado em um sistema?

48) Explique a técnica de SCAN N PASSOS utilizada para otimizar o tempo de seek em operações de E/S em disco.

49) Apresente uma classificação para dispositivos de E/S ressaltando as diferenças entre os tipos identificados e citando exemplos de dispositivos para cada um deles.

50) Descreva como ocorre uma operação de E/S considerando as camadas que compõe o software de E/S de um sistema operacional.

51) Explique o que são dicionários hierárquicos e dicionários baseados em grafos. Descreva como ocorre o gerenciamento de exclusão de arquivos e dicionários de cada um deles.

52) Argumente a favor ou contra a utilização de blocos de grande tamanho para volumes de armazenamento.

53) Um modelo de proteção pode ser visto de forma abstrata como uma matriz de aceso. Explique duas formas de implementar esta matriz e realize uma comparação entre estas abordagens.

54) Na solução abaixo para o problema dos produtores e consumidores existe algum problema caso seja invertida a ordem das operações P(full) e P(mutex) no processo consumidor? Caso não exista, argumente pela corretude da solução. Caso exista, diga qual o problema e mostre um exemplo onde ele ocorre.

```
Producer(){
                    While (TRUE)
                       produceItem(item);
```

```
            P(empty);
              P(mutex)
                   enterItem(item);
              V(mutex);
            V(full);
      }
   }
   Consumer(){
      While(TRUE){
         P(full);
            P(mutex);
              removeItem(item);
            V(mutex);
         V(empty);
      }
   }
```

Isto pode levar a um *deadlock*. Se um consumidor tentar acessar a RC com o *buffer* vazio ele fará um P no mutex e irá zerar o mutex. Logo em seguida ele fará um P no full e irá bloquear (full = 0) com mutex em 0. Dessa forma qualquer produtor que tentar escrever um item no buffer terá que fazer um P no mutex e bloqueará também. Assim sendo o *buffer* continuará vazio e o primeiro consumidor e estava bloqueado continuará assim para sempre e nunca alcançará a instrução V(mutex) causando uma situação de bloqueio perpétuo.

2007.1

55. Um pesquisador desenvolveu uma nova abordagem para impedir a ocorrência de deadlock que funciona da seguinte forma: considere um conjunto de recursos A..E pelos quais os processos concorrem. Para prevenir a ocorrência de deadlock, adiciona-se um sexto recurso F. Sempre que um processo desejar alocar qualquer um dos recursos A..E, ele deve primeiramente alocar o recurso F. Esta abordagem funciona? Justifique.

(Imcompleta)
Essa abordagem funciona. Evita a corrente circular

56. Explique como um segmento pode ser compartilhado por mais de um espaço de endereçamento quando alocação segmentada com paginação está sendo utilizada.

57. Dois processos, p1 e p2, foram projetados de forma que p2 imprime um conjunto de bytes produzidos por p1. Escreva os procedimentos executados por p1 e p2 para ilustrar como eles podem ser sincronizados através do uso de semáforos.

(Essa questão não chegou a ser aplicada porque o assunto não pertencia ao conteúdo da avaliação. Portanto tem uma grande chance de cair em prova)

58. Processos concorrentes e cooperantes necessitam trocar informações e se sincronizar. Explique os mecanismos de comunicação e sincronização usualmente disponíveis e em que situações eles são tipicamente empregados.

59. O que é uma porta de comunicação e para que serve? Como o processo a cria e quais recursos o SO aloca para seu gerenciamento? Como uma porta é identificada utilizando-se TCP/IP ?

É uma via de comunicação entre processos que estão na mesma máquina ou máquinas remotas. Serve para enviar e/ou receber mensagens.
O processo requisita uma porta o SO que fornece um número de porta alocada, e o sistema operacional gerencia através de monitores o envio, recepção e quem utiliza a porta.
TCP/IP: identificação número endereço IP

60. Descreva as tarefas realizadas pelo sistema operacional ao receber uma chamada à função open(/media/disk/foto.jpg). Ao descrever essas tarefas, comente quais as estruturas básicas do sistema são lidas do volume de armazenamento.

61. Descreva as vantagens e desvantagens de se definir blocos de tamanho pequeno em relação ao tamanho médio dos arquivos mantidos em um volume de armazenamento.

Vantagens: com o tamanho dos blocos pequenos, os arquivos armazenados tendem a não desperdiçar espaço de armazenamento ao contrário da escolha do tamanho do bloco grande que tende a desperdiçar espaço de armazenamento já que um arquivo de 1Kb, em bloco de 6Mb, consumiria os 6Mb para ser armazenado.
Desvantagens: está intimamente ligada ao desempenho. Como tamanhos blocos pequenos a recuperação será lenta, já que ele deve ler muitos blocos.

62. Em sistemas multiprogramados de tempo repartido diversos usuários compartilham o sistema simultaneamente. Descreva os mecanismos que evitam que um processo interfira na execução de outro.

(Imcompleta) Modo kernel e modo usuário.

61. Diversos processos acessam o saldo de uma dada conta corrente. Alguns processos realizam operações de crédito o outros de débito na mesma. Mostre como os monitores podem der utilizados para gerenciar o acesso compartilhado ao saldo da conta corrente.