Primeira Lista de Exercícios - SOO

Rafael Gonçalves de Oliveira Viana

4º semestre de 2017

1. Para um programador de uma linguagem de alto nível, uma chamanda de sistema assemelha-se com qualquer outra rotina de biblioteca. Existe algum motivo pelo qual o programador deveria evitar chamadas de sistema?

   Resposta: Quando a execução de uma chamada de sistema é solicitada, o sistema operacional salva todo o contexto do processo (para continuar mais tarde de onde parou), verifica as permissões envolvidas no pedido e autoriza (se for o caso) o processador a executar o serviço solicitado. Quando o processador termina a execução da chamada de sistema, o sistema operacional retorna o controle para o processo, colocando-o novamente na fila de processos prontos para a execução. Sendo assim bom otimo motivo para evitar as chamadas de sistema.

2. Como o Sistema Operacional pode impedir que um processo de usuário comunique-se diretamente com um dispositivo, como um disco rígido? Por que o Sistema Operacional deveria intervir nessa comunicação?

   Resposta Existem certas instruções que não podem ser colocadas diretamente à disposição das aplicações(Exemplo E/S), pois a sua utilização indevida ocasionaria sérios problemas à integridade do sistema(Exemplos usar blocos de memoria utilizada por algum processo do kernel). As instruções que têm o poder de comprometer o sistema são conhecidas como instruções privilegiadas (modo kernel), enquanto as instruções não-privilegiadas são as que não oferecem perigo ao sistema(Que ficam em modo Usuario),o registrador da UCP, que indica o modo de acesso corrente.

3. O programa abaixo computa $A = B*C+C*B$, sendo A, B e C matrizes quadradas de mesma ordem. Do ponto de vista computacional, seria interessante realizar as duas multiplicações simultaneamente se hover mais de uma CPU disponível. Para isso um programador escreveu:

```
void calcula(Matriz *A, Matriz *B, Matriz *C) {
  int filho;
  Matriz T1, T2;
  filho = fork();
  if (filho == 0) {
    /* Filho executa uma multiplicacao */
    multiplica(B,C,&T1);
  } else {
    /* Pai multiplica, espera o filho e soma */
    multiplica(C,B,&T2);
    waitpid(filho, NULL, 0);
    soma(&T1,&T2,A);
  }
}
```

Assuma que as matrizes estão corretamente preenchidas e alocadas e que as funções `multiplica` e `soma` funcionam corretamente armazenando o resultado no último argumento. Esta técnica produz um resultado correto mais rapidamente?

Resposta: fadfa

4. Um processo pode estar nos estados *Executando*, *Pronto* ou *Bloqueado*. O que significam? Dê três exemplos de como um processo pode ir do estado Executando para Bloqueado. Um processo Bloqueado pode sofrer preempção?

Resposta Um processo bloquea porque obviamente não pode prosseguir, em geral porque está esperando por uma entrada ainda não fornecida ou a CPU entrego seu vez de execução para outro processo.

[1] Quando se faz uma chamada de sistema o processo bloqueia, para aguardar a resposta do Hardware, até que a chamada volte e ele siga continuidade com sua atividade.

[2] Quando o escalonador decide que ele, e o processo, ja compartiharam a CPU de uma forma justa, e é hora dele deixar outro processo ocupar a CPU.

[3] O ultimo caso é quando não há temporariamente CPU disponivel para executar o processo.

5. O código do exercício 3 poderia ser convertido para uso de *threads* usando `pthread_create` e `pthred_join` para criar e esperar a *thread* nova. Quais seriam as principais diferenças entre as duas técnicas?

   Resposta A diferença de usar uma Thread a memória compartilhada (acessam a mesma possição da memória) que as possue entre si, isso é essencial para certas aplicações nas quais multiplos processos com seus espaços de endereçamento separados não funcionarão, uma segunda diferença seria a velocidade em criar e destruir, processos são cem vezes mais lento do Threads.

6. Comente duas grandes desvantagens da implementação de *threads* pelo processo em relação à implementação pelo Sistema Operacional, em relação a liberdade de execução de cada *thread*.

   Resposta Uma diferença importante entre threads de usuários e os threads de núcleo é o desempenho. O chaveamento de thead com threads de usúario usa poucas instruções de máquina. Para os threads de núcleo, o chaveamento requer um chaveamento completo do contexto, com alteração do mapa de memória e invalidação da chave, oque significa uma demora várias ordens de magnitude.Por outro lado os threads de núcleo, um thread bloqueado pela E/S não suspende o processo inteiro, como ocorre nos threads de usúario.

7. Abaixo está uma implementação candidata a resolver o problema da exclusão mútua. Quais condições da exclusão mútua são satisfeitas e quais não são por este código?

```
int trava=0;

void escreve(...) {
```

```
  ...
  while (trava != 0) ; /* espera a trava */
    /* Comandos da região crítica */
  trava = 1;           /* terminou a região crítica */
}

void le(...) {
  ...
  while (trava != 1) ; /* espera a trava */
    /* Comandos da região crítica */
  trava = 0;           /* terminou a região crítica */
}
```

8. Um semáforo pode ser implementado usando espera ocupada (por exemplo, pelo algoritmo de Peterson), supondo que a região crítica está disponível na imensa maioria dos casos e a espera acontece tão pouco que a perda de CPU é desprezível. Por que esta simplificação não deve ser usada se o S.O. não for preemptivo?

   Resposta: Em essência, o que essas soluções fazem é quando quer entrar em sua região crítica, um processo verifica se sua entrada é permitida. Se não for, ele ficará em uma laço esperando até que seja permitida a entrada. Esse método não só gasta tempo de CPU, mas também pode ter efeitos inesperados.

9. Um escalonador em lote executou um processo A por 3 hora, um processo B por mais 1 hora e, por fim, um processo C por 2 horas. Qual o tempo médio de retorno dos processos? Como você poderia melhorá-lo?

   Resposta: A primeira tarefa terminara no tempo a, o segundo termina no tempo a+b e asim por diante. O tempo médio de retorno é (3a + 1b +2c)/3 = 2 horas média.

10. Um escalonador de sistema iterativo por chaveamento circular é usado em um servidor, cujas CPUs estão completamente utilizadas enquanto o uso dos discos está abaixo de 10%. Um usuário que está fazendo uma cópia de segurança de seus arquivos está reclamando da demora. O que

pode estar causando o baixo uso do disco, embora exista trabalho a ser feito? Justifique.

(a) O quantum do escalonador está pequeno demais;

(b) O quantum do escalonador está grande demais;

(c) Há pouca memória para cache dos arquivos;

(d) A CPU é muito lenta para a demanda;

(e) Os discos são muito lentos.

Resposta:

> Este Escalonador esta dando privilegio de execução as processos limitados por CPU primeiro, e logo após para os limitados por E/S essa estrátegia tende deixar o disco ocioso, em seguida quando as tarefas limitadas por E/S executarem disputarão o disco e a CPU se encontará ociosa. A melhor maneira e manter o sistema todo executando de uma vez, formulando-se cuidadosamente essa mistura de processos."Tanenbaum"

A solução para essa questão esta relacionada ao escalonador de processso estar dando a vez para o menor tempo de utilizacao de CPU, aceitando os menores serviços primeiro e como a CPU esta sobrecarregada, vai demora para dar a vez para queles, que querem gastar um grande tempo de CPU, deixando por ultimo aqueles que desejam realizar chamada de sistema(E/I).

Adotar um quantum muito curto causa muitos chaveamentos de processo e reduz a eficiência da CPU, mas um quantum muito longo pode gerar uma resposta pobre ás requisições interativas curtas. Um quantum em torno de 20 ms a 50 ms é bastante razoável.(Pag.93)

11. Um gerenciador de impressão evita impasses no uso exclusivo da impressora, processando as solicitações posteriormente. Em qual técnica de tratamento de impasses esta estratégia se encaixa? Por que ela não é facilmente utilizável em outros casos?

Resposta: No quais dois ou mais processos estão lendo ou escrevendo algum dado comparatilhado e cujo resultado final depende de quem executa precisamente e quando, são chamadas de codições de corrida (race conditions), e complicado utilizar em outro exemplo porque é porque os resultados da maioria dos testes não apresentam problemas, mas uma hora em um momento raro, algo estranho e inexplicável acontece(Pag.71).

12. Qual a direfença entre o grafo de recursos e o grafo de recursos e intenções? Em quais situações cada um é usado?

    Resposta: Um grafo de recurso mostra os recusos utilizados pelos processos em relações de arestas,ja o grafo de recurso e intenções, adiciona um recurso e mais um tipo especial de aresta significando que um processo pretende usar um recuso. Um estado seguro não contém ciclos, mesmo com as arestas adicionadas.

13. Os atributos de um arquivo podem ser armazenos em seu nó-índice ou na entrada do diretório ao qual pertencem. Quais desvantagens existem na última opção?

    Resposta: Armazenar os atributos em sua entrada acaba prejudicando o desempenho, o motivo e simples, a tabela para conter toda lista encadeada de todos os blocos de disco, é proporcional em tamanho ao próprio disco. Se o disco tiver n blocos,a a tabela precisará de n entradas, Conforme os discos crescem, essa tablea cresce linearmente. Por outro lado o esquema de i-node requer um arranjo na memória cuja tamanho é proporcional ao número máximo de arquivos que podem estar abertos ao mesmo tempo.

14. Cite e justifique inconvenientes importantes em entradas de diretório de tamanho fixo e de tamanho variável nos Sistemas de Arquivos modernos.

    Resposta: Ter diretório de tamanho variavel, faz com que o disco fique fragmentado, ou crie latencia em buscas, porém se os tamanhos

forém fixo seria mas facil. Imagine as consequências desse tipo de projeto. O usuário inicializa um editor ou um processador de texto para escrever um documento. A primeira coisa que o programa pergutna é quantos bytes o documento final conterá. A questão deve ser respondida ou o programa não prosseguirá. Se o número fornecido se mostrar pequeno de mais o programa terá termino prematuramente, pois todas lacunas estão preencidas é não há lugar para o restante do arquivo, assin e tentado outro valor para alocar, essa técnica erá muito utilizada em grandes máquinas.

15. Quantos blocos ocupam (a) uma lista de blocos livres e (b) um mapa de blocos livres para o seguinte Sistema de Arquivos: blocos de 1KB em uma partição de disco de 512MB. (Para simplificar os cálculos, considere o padrão internacional, onde 1KB = 1000B e 1MB = 1000000B.)

    Resposta: (512000000B / 1000 B) = 512000 blocos. Essa partição vai ocupar 512000 blocos livres no mapa de blocos livres.

16. Como o *log* (ou *journal*) ajuda a manter a consistência do Sistema de Arquivos?

    Resposta: A premissa básica é a de manter um registro sobre o que o sistema de arquivos irá fazer antes que ele efetivamente o faça, de modo que, se o sistema falhar antes da execução do trabalho planejado, é possível, após a reinicialização do sistema, recorrer ao log para descobrir oque estava acontecendo no momento da parada e retomar o trabalho, esse tipo de sistema de arquivos denominado sistemas de arquivos journaling(log).

17. Considere o seguinte problema: em uma floresta, há um penhasco separando duas regiões elevadas próximas e existe um único cipó ligando as regiões sobre o imenso vale. Há tempos, os macacos da floresta utilizam o cipó para deslocar-se de um lado para outro. Como é de sua natureza, os macacos passam a vida atravessando o vale pelo cipó para fazer algo do outro lado e depois retornam para outra atividade. O maior problema na vida dos macacos é que, se dois deles encontram-se no cipó, indo em direções opostas, eles ficam bloqueados e impedem a passagem de quaisquer outros macacos, gerando muita confusão. Contudo, vários

macacos podem usar o cipó simultaneamente, quando deslocam-se no mesmo sentido. Implemente uma solução para o o uso do cipó que evita o bloqueio dos macacos, restringindo apenas o acesso dos macacos em cada ponta do cipó. Note que a passagem pelo cipó não é uma atividade rápida, já que cair no penhasco seria bastante dolorido. Logo, cada macaco pode tomar um tempo não conhecido a princípio para atravessá-lo.

Resposta: Esses macacos são doidao pega mao unica :D

18. Considere que um S.O. detecte impasses usando o grafo de recursos. Uma vez detectado um impasse, a solução adotada é matar o processo bloqueado no impasse que detém mais recursos (em caso de empate, escolhe-se qualquer um). O S.O. detecta impasses assim que são formados e cada processo só pode requisitar um recurso por vez. É possível garantir que, imediatamente após a morte do processo escolhido, o impasse foi desfeito (pelo menos um dos processos que estavam envolvidos pode voltar a executar)? Caso seja possível, justifique, caso contrário, elabore um exemplo de grafo de recursos exibindo a situação imeditamente anterior ao impasse, o impasse formado e o resultado após a morte do processo escolhido.

Resposta: ainda sera feito.!