SISTEMAS OPERACIONAIS

Lista de Exercícios – Introdução aos Sistemas Operacionais

Professor: Helder Oliveira

Entrega: 25/11/2021

Aluno(a): Vinicius Chaves Botelho - 202004940036

1. **Considere um algoritmo de escalonamento onde os processos que usam o processador por menos tempo são favorecidos. Porque este algoritmo favorece programas I/O bound e posterga programas CPU bound?**
   Lembre que:
   - **CPU Bound é quando o tempo de processamento depende mais do processador do que das entradas e saídas.**
   - **I/O bound é utilizado para designar os sistemas que fazem uso intensivo de entrada/saída.**

   Programas I/O bound que normalmente são processos de primeiro plano que interagem com usuários são priorizados, também por gastarem mais tempo no processador, logo são favorecidos pelo algoritmo.

2. **Em todos os computadores atuais, pelo menos parte dos tratadores de interrupções é escrita em linguagem de montagem. Por quê?**
   Porque ações não podem ser expressas em linguagens de alto nível pois elas não conseguem ter acesso ao hardware da CPU. Ademais, linguagens de montagem costumam ser as mesmas para todas as interrupções

3. **Quando uma interrupção ou uma chamada de sistema transfere controle para o sistema operacional, geralmente uma área da pilha do núcleo separada da pilha do processo interrompido é usada. Por quê?**
   Caso o kernel venha deixar os dados de sua pilha num espaço comum ao programa de usuário, um hacker será capaz de captar estes dados para achar informações sobre outros processos no retorno de uma chamada de sistema

4. **Presuma que você esteja tentando baixar um arquivo grande de 2 GB da internet. O arquivo está disponível a partir de um conjunto de servidores espelho, cada um deles capaz de fornecer um subconjunto dos bytes do arquivo; presuma que uma determinada solicitação especifique os bytes de início e fim do arquivo. Explique como você poderia usar os threads para melhorar o tempo de download.**
   O processo do cliente pode criar threads separados, cada thread pode buscar uma parte diferente do arquivo de um dos servidores espelho. Isso pode ajudar a reduzir o tempo de inatividade.

5. **Se um processo multithread bifurca (utilizando fork), um problema ocorre se o filho recebe cópias de todos os threads do pai. Suponha que um dos threads originais estivesse esperando por entradas do teclado. Agora dois threads estão esperando por entradas do teclado, um em cada processo. Esse problema ocorre alguma vez em processos de thread único?**
   Caso um processo de thread único estiver bloqueado no teclado, ele não poderá bifurcar- se, logo o problema não ocorreria
6. **Na Figura do slide 13 da aula 07, o conjunto de registradores é listado como um item por thread em vez de por processo. Por quê? Afinal de contas, a máquina tem apenas um conjunto de registradores**.
   Todas as CPUs têm alguns registradores internos para armazenamento de variáveis e resultados temporários, o registrador utilizado por um thread é independente, ou seja, daquele único thread.
7. **É possível que um thread seja antecipado por uma interrupção de relógio? Se a resposta for afirmativa, em quais circunstâncias?**
   Acredito
8. **Qual é a maior vantagem de se implementar threads no espaço de usuário? Qual é a maior desvantagem?**
   A maior vantagem é a eficiência. Não são necessárias traps no kernel para alternar threads. A maior desvantagem é que, se um thread é bloqueado, todo o processo é bloqueado.
9. **No problema do jantar dos filósofos, deixe o protocolo a seguir ser usado: um filósofo de número par sempre pega o seu garfo esquerdo antes de pegar o direito; um filósofo de número ímpar sempre pega o garfo direito antes de pegar o esquerdo. Esse protocolo vai garantir uma operação sem impasse? Lembre: Impasse refere-se a uma situação em que dois ou mais processos ficam impedidos de continuar suas execuções - ou seja, ficam bloqueados, esperando uns pelos outros.**
   Haverá pelo menos um garfo livre e no mínimo um filósofo que pode pegar os dois garfos simultaneamente. Desse modo, não haverá o impasse.

10. **Considere que cinco tarefas em lote, A até E, chegam a um centro de computadores quase ao mesmo tempo. Elas têm tempos de execução estimados de 10, 6, 2, 4 e 8 minutos. Suas prioridades (externamente determinadas) são 3, 5, 2, 1 e 4, respectivamente, sendo 5 a mais alta. Para cada um dos algoritmos de escalonamento a seguir, determine o tempo de retorno médio do processo. Ignore a sobrecarga de chaveamento de processo.**
    a. **Circular.**
    b. **Escalonamento por prioridade.**
    c. **Primeiro a chegar, primeiro a ser servido (siga a ordem 10, 6, 2, 4, 8).**
    d. **Tarefa mais curta primeiro.**

Primeiramente, durante os primeiros 10 minutos, cada trabalho recebe 1/5 da CPU. Durante os próximos 8 minutos, cada trabalho recebe 1/4 da CPU, após o que o término termina. Então, cada um dos três trabalhos restantes obtém 1/3 da CPU por 6 minutos, até que B termine, e assim por diante. Os tempos de finalização dos cinco trabalhos são 10, 18, 24, 28 e 30, por uma média de 22 minutos. Para o agendamento prioritário, B é executado primeiro. Após 6 minutos, está terminado. Os outros trabalhos terminam aos 14, 24, 26 e 30, por uma média de 18,8 minutos. Se os trabalhos são executados na ordem de A a E, terminam em 10, 16, 18, 22 e 30, por uma média de 19,2 minutos. Por fim, o primeiro trabalho mais curto produz tempos de acabamento de 2, 6, 12, 20 e 30, por uma média de 14 minutos.