Teste 1 de Sistemas Operacionais

Nome: Daniel Sant' Anna Andrade
Matrícula: 20200036904
Nome: Luan Gadioli Mendonça Berto
Matrícula: 20200034042

1) As linhas de execução de uma thread compõem a execução de um processo. Em muitas aplicações há múltiplas atividades ao mesmo tempo, então, podemos decompor as threads em atividades paralelas. As threads são mais fáceis de serem criadas e destruídas do que os processos, em alguns casos podem ser 100 vezes mais rápidas.

2) **Modo do Usuário**

   Vantagem: O sistema operacional nesse tipo de thread não faz o escalonamento, elas são criadas, executadas, escalonadas e destruídas pelo programador. Isso gera a vantagem de cada processo usar um algoritmo de escalonamento que melhor se adapte a situação. Nesse modo cada processo possui sua própria tabela de threads. Troca de contexto entre os threads de usuário não envolve a passagem de modo usuário para modo supervisor (chamada ao sistema), sendo muito mais rápida.
   Desvantagem: Se um thread fizer uma chamada bloqueante, todo o processo será bloqueado, pois o kernel não terá acesso. Thread de usuário não é interrompido pelo escalonador.

   Modo do Núcleo
   Vantagem: o Núcleo escolhe a thread diretamente, ele é quem vai gerenciar e escalonar. Sendo assim, threads bloqueadas por entrada e saída não bloqueiam o processo. O núcleo permite múltiplas threads em paralelo, possui conhecimento dos threads do processo e Fornece interfaces de gerenciamento de threads.
   Desvantagem: Gerenciar threads em modo kernel é mais caro, devido a alternância entre o modo usuário e o modo kernel. Sobrecarga de gerenciamento: Cada chamada de função de gerenciamento de threads é uma chamada ao sistema

3) Sim, Quando threads são executadas infinitamente, uma solução possível para o problema é obrigar o sistema de tempo de execução a solicitar um sinal de relógio a cada segundo para dar a ele o controle. Porém, Interrupções periódicas de relógio em uma frequência mais alta nem sempre são possíveis, e mesmo que fossem, a sobrecarga total poderia ser substancial. Além disso, um thread talvez precise também de uma interrupção de relógio, interferindo com o uso do relógio pelo sistema de tempo de execução.

4) Porque se transfere para o sistema operacional para liberar espaço para outro processo, então, se utilizar a área da pilha do processo interrompido, esse outro processo se interrompe juntamente.
Existem várias razões para usar uma pilha separada para o kernel. Dois deles são os seguintes. Primeiro, você não deseja que o sistema operacional falhe porque um programa de usuário mal escrito não permite espaço suficiente na pilha. Segundo, se o kernel deixar dados da pilha no espaço de memória de um programa do usuário ao retornar de uma chamada do sistema, um usuário mal-intencionado poderá usar esses dados para encontrar informações sobre outros processos.

5) Através do código fonte é possível verificar a frequência com a qual operações E/S são executadas, já durante a execução deve-se utilizar ferramentas que medem a porcentagem de uso da CPU por um dado programa. Entretanto isto somente é válido em máquinas que permitem apenas um único usuário

6) Se o programa chama ou salta para uma instrução que não esteja na memória, ocorre uma falta de página e o sistema operacional buscará a instrução perdida (e suas vizinhas) do disco. Isso é chamado de uma falta de página. O processo é bloqueado enquanto as instruções necessárias estão sendo localizadas e lidas. Se um thread causa uma falta de página, o núcleo, desconhecendo até a existência dos threads, naturalmente bloqueia o processo inteiro até que o disco de E/S esteja completo, embora outros threads possam ser executados.

7) A fração de tempo desperdiçada da CPU seria de 0,5^5 = 0,03125 ou 1/32 correspondente à chance de todos os cinco processos estarem ociosos.

8) Caso o servidor esteja vinculado à CPU, um servidor multithread apenas torna mais complexo a execução. Com um servidor armazenando dados pequenos, como uma seção de cadastro, é possível armazenar os dados cadastrados na memória do servidor, tornando mais rápida a consulta/utilização desses dados.

9) A utilização da CPU é calculada através da fórmula 1-(p^n), onde p é a fração de tempo pela espera de uma entrada ou saída de um dispositivo e n corresponde ao número de processos. Logo, a chance de que todos os processos estejam à espera de entrada e saída de um dispositivo é 1-(0,4^6) = 1-0,004096 = 0,995904, ou seja, aproximadamente 99,6% de utilização da CPU.

10) Como um computador tem 4096 mb de ram e o sistema ocupa 512 mb, sobra 3584 mb. Como cada processo ocupa 256mb, é possível executar 14 processos simultaneamente.
Utilizando a fórmula de utilização da CPU = 1 - (p^n). p = ?
n = 14

CPU = 99%
Logo:
0,99 = 1 - p^14
p^14 = 1 - 0,99
p = (0,01)^(1/14)
p =~ 0,720
Assim, o tempo máximo de espera de E/S é de 72%