Atividade 2 – Sistemas Operacionais (SO)

Júlia Simone Araújo

1. O que significa time sharing e qual a sua importância em um sistema operacional?

Como o nome já diz, tempo compartilhado, o time sharing tem como premissa permitir o processo de múltiplas tarefas de forma simultânea. Introduzida na década de 60, pelo sistema CTSS, é importante por possuir um tempo (quantum) de retorno enquanto possibilita a gerência de outros processos.

2. Como e com base em que critérios é escolhida a duração de um quantum de processamento?

Baseada pela forma que for definido o melhor jeito de se trabalhar com os processamentos de tarefas, e de acordo com a prioridade estabelecida pela quantidade e velocidade de cada uma a ser processada. Possui como critério: tempo de espera, de execução, de resposta e por fim, eficácia.

3. Considerando o diagrama de estados dos processos apresentado na figura a seguir, complete o diagrama com a transição de estado que está faltando (t6) e apresente o significado de cada um dos estados e transições.

T1: Executando -> Terminada;
T2: Executando -> Suspensa;
T3: Suspensa -> Pronta;
T4: Pronta -> Executando;
T5: Nova -> Pronta;
T6: Executando -> Pronta.

4. Indique se cada uma das transições de estado de tarefas a seguir definidas é possível ou não. Se a transição for possível, dê um exemplo de situação na qual ela ocorre (N: Nova, P: pronta, E: executando, S: suspensa, T: terminada).

E→P: Possível. Exemplo: acontece quando um quantium chega ao fim;
E→S: Possível. Exemplo: caso a atividade em execução solicita algum recurso indisponível, ela libera o processador e fica suspensa até o ser disponível o uso do recurso;
S→E: Impossível;
P→N: Impossível;
S→T: Impossível;
E→T: Possível. Exemplo: quando uma atividade finaliza sua execução ou é cancelada por algum erro;
N→S: Impossível;
P→ S: Impossível.

5. Relacione as afirmações abaixo aos respectivos estados no ciclo de vida das tarefas (N: Nova, P: Pronta, E: Executando, S: Suspensa, T: Terminada):
[N] O código da tarefa está sendo carregado.
[P] A tarefas são ordenadas por prioridades.
[E] A tarefa sai deste estado ao solicitar uma operação de entrada/saída.
[T] Os recursos usados pela tarefa são devolvidos ao sistema.
[P] A tarefa vai a este estado ao terminar seu quantum.
[P] A tarefa só precisa do processador para poder executar.
[P] O acesso a um semáforo em uso pode levar a tarefa a este estado.
[E] A tarefa pode criar novas tarefas.
[E] Há uma tarefa neste estado para cada processador do sistema.
[S] A tarefa aguarda a ocorrência de um evento externo.

6. Explique o que é, para que serve e o que contém um TCB - Task Control Block.

Uma estrutura de dados responsável por trazer as informações de uma tarefa, conhecida como contexto de tarefa. Contém uma função capaz de descrever as principais informações da mesma, como: estado, valores de registro, identificador, etc.

7. O que são threads e para que servem?

É uma estrutura de dados que definem como será o funcionamento do processador, pois elas que recebem e executam instruções. Devido sua capacidade de permitir aplicações rodarem de forma independente, no mesmo ambiente, causa a impressão de paralelismo.

8. Quais as principais vantagens e desvantagens de threads em relação a processos?

Pela divisão de processos proporcionada pelas theards, temos como vantagem o próprio paralelismo em si e até o desenvolvimento em si, por tornar possível a elaboração e criar o programa em módulos. Como desvantagem, temos a complexidade de um trabalho rodar caso tenhamos muitas, pela interação entre elas.

9. Forneça dois exemplos de problemas cuja implementação multi-thread não tem desempenho melhor que a respectiva implementação sequencial.

1. Aplicação que roda execuções matemáticas complexas;
2. Aplicação com grande quantidade de processos, em algum deles um TBC com descrições complexas para serem rodadas em paralelo.

10. Associe as afirmações a seguir aos seguintes modelos de threads: a) many-to-one (N:1); b) one-to-one (1:1); c) many-to-many (N:M):
(a) Tem a implementação mais simples, leve e eficiente. – A (many-to-one)
(b) Multiplexa os threads de usuário em um pool de threads de núcleo. – B (one-to-one)

(c) Pode impor uma carga muito pesada ao núcleo. – B (one-to-one)
(d) Não permite explorar a presença de várias CPUs pelo mesmo processo. – A (many-to-one)
(e) Permite uma maior concorrência sem impor muita carga ao núcleo. – C (many-to-many)
(f) Geralmente implementado por bibliotecas. – A (many-to-one)
(g) É o modelo implementado no Windows NT e seus sucessores. – B (one-to-one)
(h) Se um thread bloquear, todos os demais têm de esperar por ele. – A (many-to-one)
(i) Cada thread no nível do usuário tem sua correspondente dentro do núcleo. – C (many-to-many)
(j) É o modelo com implementação mais complexa. – C (many-to-many)