Avaliação 6 – Semáforos, problemas clássicos e impasses

Fonte: Prof. Msc. Jacson Rodrigues; Maziero, 2019. Silberschatz, A. Operating System Concepts - 8th Ed.

Responda as questões **justificando** sua solução.

1. Considere a seguinte política de alocação de recursos: solicitações e devoluções de recursos são permitidas a qualquer momento. Se uma solicitação de recursos não pode ser satisfeita porque os recursos não estão disponíveis, então verificam-se os processos que estão bloqueados, aguardando recursos. Se eles têm o recurso solicitado, então esse recurso é tirado dele e dado para o processo solicitante.
   a) Pode ocorrer um impasse? Se você responder “sim”, dê um exemplo. Se você responder “não”, especifique qual condição necessária não ocorre.

2. É possível ter um deadlock envolvendo apenas um único processo? Explique sua resposta.

3. Suponha que um sistema esteja em um estado inseguro. Mostre que é possível que os processos completem sua execução sem entrar em um estado de impasse.

4. Considere um sistema de computador que executa 5.000 processos por mês sem esquema de prevenção de impasse ou de impedimento de impasse. Impasses ocorrem cerca de duas vezes por mês, e o operador deve encerrar e executar novamente cerca de 10 processos por impasse. Cada processo vale cerca de $ 2 (em tempo de CPU), e o os processos encerrados tendem a ser concluídos pela metade quando são abortados. Um programador de sistemas estimou que um algoritmo de prevenção de impasses (como o algoritmo do banqueiro) pode ser instalado no sistema com um aumento no tempo médio de execução por trabalho de cerca de 10 por cento. Uma vez que a máquina atualmente tem 30 por cento de tempo ocioso, todos os 5.000 processos por mês ainda poderiam ser executados, embora o tempo de execução aumentasse em cerca de 20 por cento em média.
   a) Quais são os argumentos para instalar a prevenção de impasses ?
   b) Quais são os argumentos contra a instalação?

5. O código abaixo é uma tentativa de implementar uma solução para Barreiras. Como, *count* é protegido por um *mutex*, essa variável conta a quantidade de todas as *threads* que passam pela barreira. As primeiras n-1 threads esperam até todas as outras (n no total) cheguem até *barrier*, que está inicialmente ocupada. Quando a n-ésima thread chega, ela destrava a barreira.

```
n = the number of threads                 mutex.wait( )
count = 0                                         count = count + 1
mutex = Semaphore (1)                     mutex.signal ( )
barrier = Semaphore (0)
                                          if count  == n
                                                  barrier.signal ( )

                                          barrier.wait ( )

                                          <critical point>
```

6. Considere uma barbearia onde existe uma sala de espera com N cadeiras e uma sala de serviço com uma cadeira para corte de cabelo. Se não há clientes esperando o barbeiro deve ir dormir. Se um cliente entrar na barbearia e todas as cadeiras de espera estejam ocupadas o cliente deve deixar o local. Se o barbeiro está ocupado (cortando o cabelo de outro cliente), porém há pelo menos uma cadeira de espera livre, então o cliente ocupa uma dessas cadeiras. Se o barbeiro está dormindo, o cliente deve acordá-lo. Escreva um programa (pseudocódigo) para coordenar o barbeiro e os clientes. Utilize as seguintes informações:

   i. A thread do cliente deve invocar uma função *ObterCorteCabelo( )*;
   ii. Se um cliente chega e a barbearia está cheia, a função *Volta( )* é chamada;
   iii. O barbeiro deve chamar a função *CortarCabelo( )*;
   iv. Quando *CortarCabelo( )* é invocada deve haver um cliente chamando *ObterCorteCabelo( )* de forma concorrente.

   Usar as seguintes variáveis:

   - numero_clientes = 0
   - mutex = Semaphore(1)

- cliente = Semaphore(0)
- barbeiro = Semaphore(0)

A variável *numero_clientes* conta o número de clientes em espera e é protegida pela variável *mutex*. O barbeiro espera por um cliente até que algum entre na barbearia. Então o cliente acorda o barbeiro o toma um assento. Caso seja necessário utilizar outras variáveis justifique sua escolha e explique a finalidade de cada uma.