Atividade 6 – Sistemas Operacionais (SO)

Júlia Simone Araújo

1.

a) No cenário descrito um impasse não pode ocorrer, pois a preempção é permitida. Isso que dizer que qualquer processo ao solicitar um recurso em qualquer momento, mesmo que um processo bloqueado tenha reservado o recurso, ele é transferido para o novo processo. O novo processo não precisará esperar a execução do processo bloqueado para ter acesso ao recurso desejado.

2.

Não, pois um impasse envolve dois ou mais processos que ficam bloqueados, esperando um pelos outros. Pela sua característica de recursos não-preempetíveis, o sistema roda de forma cíclica, “manter e esperar”, levando a um dos processos envolvidos não poder conter um recurso, ainda que esteja esperando por outro que esteja mantendo.

3.

Sabendo que um estado inseguro não leva necessariamente a um impasse, isso quer dizer que um sistema pode, ainda que em estado inseguro, permitir que todos os processos sejam concluídos sem a ocorrência de um impasse.
Considerando o cenário de um sistema com 12 recursos alocados entre os processos P0, P1 e P2, seguindo a política:

- P0 = 10 (máximo); 5 (atual); 5 (necessário);
- P1 = 4 (máximo); 2 (atual); 2 (necessário);
- P2 = 9 (máximo); 3 (atual); 6 (necessário).

Há, atualmente, dois recursos disponíveis e o sistema está em um estado inseguro pois o processo P1 poderia completar, liberando um total de quatro recursos. Porém não podemos garantir que P0 e P2 podem completar, como também é possível que um processo libere recursos antes da requisição de outro. Nesse cenário temos a possibilidade de P0 concluir, liberando nove recursos, bem como P2 conseguir concluir em seguida.

4.

a) Economizaria o trabalho do programador de precisar reiniciar o sistema e rodar novamente, do zero, processos que foram processados pela metade devido a estarem envolvidos no conflito. Além do tempo que ele economizaria de espera para que elas sejam concluídas de verdade.

b) O tempo de execução seria mais lento em cada processo que esteja sendo executado e, claro, o custo de instalação que poderia não trazer retorno instantemente.

5.

Ao analisar o código identifica-se um problema no count==n, isso porque se o n-ésimo encadeamento for interrompido neste ponto e o n-ésimo encadeamento vem através do mutex, ambos descobrirão count == n e sinalizarão para destravar a barreira.

6.

```
numero_clientes = 0 // numero de clientes
mutex = semaphore(1)
cliente = semaphore(0)
barbeiro = semaphore(0)
cadeiras = n //numero de cadeiras

cliente() {
        down(mutex); //entra na região crítica

        if(numero_clientes == n){
                Volta();
                up(mutex) //sai da região crítica
        }

        else{
                numero_clientes++; //aumenta o numero de clientes
                up(cliente); //avisa que há um novo cliente para cortar o cabelo
                up(mutex); //sai da região crítica
                down(barbeiro); //espera o barbeiro cortar o cabelo
                ObterCorteCabelo(); //pede para cortar
        }
}


barbeiro( ){
        while(1){
        down(cliente); //dorme sem cliente
        down(mutex); //entra na região crítica
        numero_clientes--; //diminui numero de clientes nas cadeiras
        up(mutex); //barbeiro pronto pra cortar cabelo
        up(barbeiro); //sai do estado inseguro
        CortarCabelo( ); //corta o cabelo
}
```