CMP1126 - Sistemas Computacionais Aplicados II
Max Gontijo de Oliveira
2º Trabalho - Simulação de Deadlock

1 Resumo

Desenvolver um programa que cria *threads* que concorrem por recursos compartilhados. O programa deverá garantir exclusão mútua aos recursos, de modo que isso leve ao *deadlock*.

2 Detalhes

O programa deverá simular a existência de 6 recursos identificáveis. Esses recursos podem ser, por exemplo, simplesmente uma posição em um vetor de strings, onde cada posição desse vetor é um dos recursos.

O programa deverá garantir a exclusão mútua dos recursos criando um semáforo para cada recurso (também poderia ser um vetor de semáforos).

O programa deverá criar 4 *threads* que irão executar as mesmas linhas de código. Nessa execução, as *threads* irão alocar recursos com exclusividade, usá-los, e liberá-los. Cada *thread* irá alocar um número k de recursos ao mesmo tempo para que, então possa utilizá-los. Sendo 4 *threads*, uma *thread* irá alocar apenas um recurso por vez ($k = 1$); outra *thread* irá alocar dois recursos de uma vez ($k = 2$); outra *thread* irá alocar três recursos de uma vez ($k = 3$); e outra *thread* irá alocar quatro recursos de uma vez ($k = 4$)

Essas linhas devem ser executadas em *loop* infinito seguindo os seguintes passos:

1. Imprimir uma mensagem informando que k recursos serão sorteados.
2. Obter randomicamente uma lista de k recursos, onde k representa a quantidade de recursos que uma *thread* irá alocar.
3. Imprimir uma mensagem informando que os k recursos irão ser alocados.
4. Alocar (bloquear) os recursos da lista gerada randomicamente um a um, **na ordem em que foram sorteados**.
   - Antes do bloqueio de cada um dos recursos, imprimir uma mensagem informando que o recurso em questão será bloqueado pela *thread* em questão.
   - Utilizar o semáforo individual de cada recurso sorteado para realizar os bloqueios.
   - Após o bloqueio de cada um dos recursos, imprimir uma mensagem informando que o recurso em questão foi bloqueado pela *thread* em questão.
5. Simular o uso dos recursos
   - Nessa simulação, basta a impressão na tela de uma simples mensagem indicando que o recurso em questão está sendo utilizado pela *thread* em questão.
6. A *thread* em questão deveria dormir por um tempo (uns 200 a 500 milissegundos) apenas para segurar o recurso por um tempinho.
7. Liberar (desbloquear) os recursos alocados.
   - A *thread* deverá simplesmente liberar os semáforos.
   - Após a liberação de cada um dos recursos, imprimir uma mensagem informando que o recurso em questão foi liberado pela *thread* em questão.
8. A *thread* em questão deveria dormir por um tempo (uns 200 a 500 milissegundos) apenas para esperar um pouco antes de começar a alocar recursos novamente.

Executando os passos conforme descrito acima, o programa deveria entrar em *deadlock* em algum momento.

Quando isso acontecer, deverá ser possível, através das últimas mensagens impressas, determinar a dependência circular existente. Em outras palavras, deverá ser possível identificar que recursos cada uma das *threads* está aguardando para prosseguir.

Por fim, o aluno deverá resolver o problema do *deadlock* por meio de alocação ordenada dos recursos.

3 Artefatos de entrega

Deverá ser entregue apenas o código fonte do programa escrito na linguagem que o aluno escolher. Este deverá ser enviado para o e-mail do professor (`max.gontijo.oliveira@gmail.com`).