Exercícios de Sistemas Operacionais – Prof. Fernando Dotti - Escola Politécnica - PUCRS

Utilizando threads e semáforos, na sua linguagem de programação preferida:

1) Implemente duas soluções para evitar o deadlock no jantar dos filósofos.

2) Usando o modelo produtor/consumidor, de buffer finito, implemente um buffer circular com N elementos, N configurável. Cada elemento do buffer é um booleano inicializado em falso. A operação de inserção coloca true na próxima posição disponível, e avança a posição de inserção para a próxima – partindo do início, ao chegar no final, deve voltar ao inicio circularmente. Se não houver posição livre, deve bloquear até que exista. A operação de retirada tem um controle de posição de retirada. Se não há elementos no buffer, deve bloquear até haver. Se há elemento, retira setando o mesmo para falso e avança a posição de retirada. Incialmente as posições de inserção e retirada são 0. Assim se posicao de inserção == posicao de retirada, o buffer está vazio. Se posição de inserção +1 == posição de retirada, o buffer encheu. As threads consumidoras e produtoras devem ser respectivamente bloqueadas nestes casos. A cada modificação do buffer, imprima o resultado na tela de forma que você possa ver as posições do buffer circularmente se tornarem True e False. As threads devem invocar buffer.insere() e buffer.retira(). Estas operações devem cuidar de toda sincronização utilizando semáforos. O buffer deve suportar várias produtoras e várias consumidoras simultaneamente.

3) Utilizando barreiras reutilizáveis, resolva o seguinte problema.

   Suponha um sistema com uma matriz N x N, com N arbitrário positivo, preenchida com valores randômicos. Esta matriz representa dados a serem processados em fases. Em cada fase, o valor de uma célula passa a ser a média dos valores das vizinhas a Norte, Sul, Leste, Oeste. **Cada célula da matriz é processada por uma thread.** Em cada iteração a thread (nesta sequência) lê os valores de todas vizinhas, calcula a média, espera que todas threads tenham feito isso e então atualiza sua célula. **Esta espera é implementada com uma barreira.** A nova leitura de valores dos vizinhos, para calcular o novo valor da célula, só pode ser feito quando todas threads já atualizaram suas células na fase anterior. O sistema faz um número configurável de iterações. Para resolver este problema, estude barreiras e barreiras reutilizáveis no "Little book of semaphores", além de usar o material de aula.

4) Implemente o problema dos leitores e escritores, Readers and Writers do “Little Book of Semaphores”, com L leitores e E escritores. As threads leitoras ficam em loop lendo, e as escritoras em loop escrevendo. O recurso é um string compartilhado. As operações string recurso.leitura() e recurso.escrita(string) implementam a sincronização conforme o modelo de leitores e escritores. Veja no livro. Como experimento, varie o numero de L de leitoras em relação ao número E de escritoras e avalie como isto afeta a postergação das escritas. Descreva os resultados obtidos e sua análise.