Universidade de Brasília Faculdade UnB Gama

Projeto 01

Prof. Mateus Mendelson
mendelson@unb.br

Problema 1: Validador de Sudoku

Um *Sudoku* utiliza um grid 9 x 9 no qual cada linha e cada coluna, bem como cada um dos nove subgrids 3 x 3, deve conter todos os dígitos 1 ... 9. A figura abaixo apresenta um exemplo de um Sudoku válido. Este projeto consiste em desenvolver um programa multithread que determina se a solução de um Sudoku é válida.

6	2	4	5	3	9	1	8	7
5	1	9	7	2	8	6	3	4
8	3	7	6	1	4	2	9	5
1	4	3	8	6	5	7	2	9
9	5	8	2	4	7	3	6	1
7	6	2	3	9	1	4	5	8
3	7	1	9	5	6	8	4	2
4	9	6	1	8	2	5	7	3
2	8	5	4	7	3	9	1	6

Há diferentes formas de fazê-lo. Uma estratégia sugerida consiste em criar threads que checam os seguintes critérios:

- Uma thread que checa se cada coluna contém os dígitos de 1 até 9
- Uma thread que checa se cada linha contém os dígitos de 1 até 9
- Nove threads que checam se cada subgrid de 3 x 3 contém os dígitos de 1 até 9

Essa abordagem resultaria em onze threads distintas para validar o Sudoku. Entretanto, você é livre para criar ainda mais threads for esse projeto. Por exemplo, em vez de criar uma thread que checa todas as nove colunas, poderiam ser criadas nove threads distintas e cada uma checar uma coluna.

O grid deve ser lido a partir de um arquivo texto de entrada, contendo os elementos do Sudoku. O resultado deve ser impresso na tela.

Retornando resultados para a thread pai

Cada thread realizará a validação de determinada região do grid. Quando uma thread finaliza sua análise, é necessário retornar seu resultado para a thread pai. Uma boa maneira de lidar com isso é criar um array de inteiros visível por todas as threads. O i-ésimo índice no array corresponde à i-ésima thread. Se uma thread escreve em sua respectiva posição o valor 1, há indicação de que a região do Sudoku sobre sua responsabilidade é válida. O valor 0 indica que não é válida. Quando todas as threads terminarem suas tarefas, a thread pai realizará a checagem de cada entrada do array para determinar se o grid apresentado é válido.

Problema 2

O departamento de ciência da computação de uma universidade tem um assistente de ensino (AE) que ajuda os estudantes de graduação em suas tarefas de programação durante as horas normais de expediente. O escritório do AE é pequeno e só tem espaço para uma mesa com uma cadeira e um computador. Existem três cadeiras no corredor fora do escritório em que os estudantes podem sentar e esperar se, no momento, o AE estiver ajudando outro estudante. Quando não há estudantes que precisem de ajuda durante as horas de expediente, o AE senta-se à mesa e tira uma soneca. Se um estudante chega durante as horas de expediente e encontra o AE dormindo, deve acordá-lo para pedir ajuda. Se um estudante chega e encontra o AE ajudando outro estudante, deve sentar-se em uma das cadeiras do corredor e esperar. Se não houver cadeiras disponíveis, o estudante voltará mais tarde.

Usando threads, locks mutex e semáforos do POSIX, implemente uma

solução que coordene as atividades do AE e os estudantes. Detalhes desse exercício são fornecidos abaixo.

Os Estudantes e o AE

Usando o Pthreads, comece criando n estudantes (n aleatório, podendo variar de 3 até 40). Cada estudante será executado como um thread separado. O AE também será executado como um thread separado. Os threads de estudantes se alternarão entre a programação por um período de tempo aleatório e a busca de ajuda do AE. Se o AE estiver disponível, eles obterão ajuda e, após receber ajuda 3 vezes, encerrarão sua execução. Caso contrário, sentarão em uma cadeira no corredor (a quantidade c de cadeiras deve ser igual a metade da quantidade n de estudantes) ou, se não houver cadeiras disponíveis, voltarão a programar e procurarão ajuda em um momento posterior. Se um estudante chegar e notar que o AE está dormindo, deve notificar o AE usando um semáforo. Quando o AE terminar de ajudar um estudante, deve verificar se há estudantes esperando por ajuda no corredor. Se houver, o AE deve ajudar esses estudantes, um de cada vez. Se não houver estudantes presentes, o AE pode voltar a dormir. Talvez a melhor opção para a simulação da programação dos estudantes — assim como do AE fornecendo ajuda a um estudante — seja fazer os threads apropriados adormecerem por um período de tempo aleatório.

Sempre que o estado de algum thread mudar, uma mensagem na tela deve ser exibida indicando seu novo estado. Por exemplo, quando o AE estiver atendendo o estudante 3, deve-se imprimir na tela "AE ajudando o estudante 3". Quando um estudante estiver sentado na fila, deve-se imprimir "Estudante 3 sentado na cadeira" e assim por diante.

As Seções 4.4.1 e 5.9.4 do livro em português apresentam as funções que devem ser utilizadas na resolução desse problema.