Trabalho 1 – Verificador de Sudoku Multithread

INE5412 – Sistemas Operacionais I – UFSC
Prof. Márcio Castro

1 Introdução

O sudoku é um quebra-cabeça baseado na colocação lógica de números criado por *Howard Garns*, um projetista e arquiteto de 74 anos aposentado. O objetivo do jogo é a colocação de números de 1 a 9 em cada uma das células vazias numa grade de 9x9, constituída por 3x3 subgrades chamadas **regiões**. Considere que as linhas, colunas e regiões da grade são numeradas de 1 à 9. As regiões são numeradas da seguinte forma: região 1 (linhas e colunas de 1 à 3), região 2 (linhas de 1 à 3 e colunas de 4 à 6), região 3 (linhas de 1 à 3 e colunas de 7 à 9), região 4 (linhas de 4 à 6 e colunas de 1 à 3), região 5 (linhas de 4 à 6 e colunas de 4 à 6), região 6 (linhas de 4 à 6 e colunas de 7 à 9), etc. A Figura 1 apresenta um exemplo de um quebra-cabeça sudoku a ser resolvido.

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

Figura 1: Exemplo de um quebra-cabeça sudoku.

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

Figura 2: Exemplo de quebra-cabeça sudoku resolvido.

A regra para a colocação dos números nas células vazias é a seguinte. Em cada **coluna**, **linha** e **região** da grade, os números de 1 à 9 só podem aparecer uma única vez. Em outras palavras, não é permitido a repetição de um número em uma mesma linha, coluna ou região da grade. A Figura 2 apresenta um exemplo de uma solução para o quebra-cabeças da Figura 1. Note que, nesse caso, não há repetição de um mesmo numero em uma mesma linha, coluna ou região da grade. Logo, essa solução está **correta**.

2 Definição do Trabalho

Será disponibilizado no Moodle um programa em C que lê um quebra-cabeça sudoku de um arquivo e o armazena em uma matriz de tamanho 9x9. A partir desse código, o primeiro trabalho da disciplina de Sistemas Operacionais I consistirá em desenvolver um programa concorrente utilizando POSIX threads (Pthreads) que verifica se uma dada solução do quebra-cabeça sudoku está correta ou não.

O programa concorrente deverá receber um segundo parâmetro pela linha de comando referente ao número de *threads* a serem utilizadas. Após serem criadas, as *threads* deverão dividir o trabalho de verificação da solução do quebra-cabeça sudoku fornecido. A forma de divisão do trabalho de verificação da solução do quebra-cabeça sodoku entre as *threads* deverá ser definida pelo grupo. Porém, deseja-se evitar ao máximo que *threads* permaneçam ociosas sem realizar nenhum trabalho. A solução deverá funcionar para qualquer número de *threads*.

Sempre que um a *thread* encontrar um erro na solução ela deverá imprimir na tela **"Thread XX: erro na YY ZZ"**, onde: XX representa o número da *thread*; YY indica se foi em uma linha, coluna ou região; e ZZ representa o número da linha, coluna ou região onde o erro ocorreu. As *threads* criadas deverão receber uma **numeração sequencial** de 1 até o número de *threads*. Além disso, **ao final da execução, deverá ser impresso número total de erros encontrados.**

As Figuras 3 e 4 mostram um exemplo de saída quando uma solução estiver correta e incorreta, respectivamente. Na Figura 3, nenhuma *thread* encontrou um erro na solução fornecida e, portanto, a frase "A solução está correta!" foi impressa na tela. Por outro lado, a Figura 4 apresenta um caso onde a solução do quebra-cabeças sudoku fornecida está incorreta. Nesse exemplo, foram utilizadas somente duas *threads*, as quais identificaram erros na solução e imprimiram os resultados dos erros encontrados por elas na tela. A sua solução concorrente deverá seguir **rigorosamente** esse formato de saída. Logicamente, a ordem de apresentação dos resultados assim como a distribuição das tarefas entre

```
$ ./sudoku input_grid_correto.txt 2
Quebra-cabecas fornecido:
5 3 4 6 7 8 9 1 2
6 7 2 1 9 5 3 4 8
1 9 8 3 4 2 5 6 7
8 5 9 7 6 1 4 2 3
4 2 6 8 5 3 7 9 1
7 1 3 9 2 4 8 5 6
9 6 1 5 3 7 2 8 4
2 8 7 4 1 9 6 3 5
3 4 5 2 8 6 1 7 9
Erros encontrados: 0.
```

Figura 3: Um exemplo de saída quando a solução estiver correta.

```
$ ./sudoku input_grid_errado.txt 2
Quebra-cabecas fornecido:
3 5 4 6 7 8 9 1 2
6 7 2 1 9 5 3 4 8
1 9 8 3 4 2 5 6 7
8 5 9 7 6 1 4 2 7
4 2 6 8 5 3 7 9 1
7 1 3 9 2 4 8 5 6
9 6 1 5 3 7 2 8 4
2 8 7 4 1 9 6 3 5
3 4 5 2 8 6 1 7 9
Thread 1: erro na coluna 1.
Thread 2: erro na coluna 2.
Thread 1: erro na coluna 9.
Thread 2: erro na linha 4.
Thread 1: erro na regiao 6.
Erros encontrados: 5.
```

Figura 4: Um exemplo de saída com duas *threads* quando a solução estiver incorreta.

as *threads* poderá mudar em função da forma de divisão do trabalho adotada e também de uma execução à outra por se tratar de um programa concorrente.

3 Grupos, Avaliação e Entrega

O trabalho deverá ser realizado **obrigatoriamente** em grupos de **3 alunos**. Os alunos serão responsáveis por formar os grupos com auxilio da ferramenta "**Escolha de Grupos - Trabalho 1 (T1)**" disponível no Moodle.

Pelo menos um dos integrantes de cada grupo deverá submeter um arquivo ZIP contendo: (i) o código fonte em C da solução do trabalho; e (ii) um relatório em PDF (mínimo 2 páginas) explicando e justificando a solução adotada. A data/hora limite para o envio dos trabalhos é **24/04/2017 às 23h55min**. **Não será permitida a entrega de trabalhos fora desse prazo: trabalhos não entregues no prazo receberão nota zero.**

O professor irá avaliar não somente a corretude mas também o desempenho e a clareza da solução. Além disso, os relatórios que explicam a solução adotada também serão avaliados. **A implementação e relatório terão pesos 70% e 30% da nota do trabalho, respectivamente**.