Universidade Federal de Uberlândia

Faculdade de Computação

Sistemas de Informação

Trabalho de Sistemas Operacionais
Manipulando Threads - Sudoku

Professor: Marcelo Zanchetta do Nascimento

Alunas:

Beatriz Ribeiro Borges - 12021BSI231

Laura Rosado Rodrigues Muniz - 12021BSI216

Sara Rosado Rodrigues Muniz - 12021BSI215

Uberlândia, 25 de janeiro de 2023

Sumário

Descrição do Problema

Este trabalho pretende verificar se com uma dada entrada, lida de um arquivo, é possível resolver o quebra-cabeças “Sudoku”. O jogo usa uma matriz 9×9 elementos em que cada coluna e linha deve conter os dígitos de 1 a9, assim como cada uma das nove submatrizes com 3×3 elementos. A ideia é fazer uma verificação das regras utilizando threads e retornando se a entrada é válida de acordo com as regras ou não.

Solução

Para determinar se a entrada para o Sudoku será válida foram usados 27 threads. 9 para cada submatriz 3x3, 9 para as colunas e 9 para as linhas.

Foi feito um array de inteiros “valida” de tamanho 27 (tamanho da quantidade de threads) que será usado para a verificação se aquela região do thread é válida (passando o inteiro 1) ou não (passando o inteiro 0). Foi inicializado com 0. Também foi definida a Struct contendo linha e coluna, inteiros, que armazena os dados a serem passados para os threads.

Foi definida uma função “ehColunaValida” que verifica se uma coluna específica do quebra-cabeça sudoku é válida. A função primeiro verifica se a linha e a coluna passadas como parâmetros são válidas. Se não, ela sai do thread. Em seguida, a função inicializa um array chamado “validaArray” com todos os elementos definidos como 0. Se o número não é válido ou já apareceu anteriormente, a thread é interrompida. Do contrário, a coluna é válida e o valor correspondente no array “valida” é definido como 1. A mesma lógica foi usada para “ehLinhaValida” que verifica as linhas.

Para verificar as submatrizes foi definida a função “eh3x3Valida”, também executada por um thread. Primeiro, verifica se as linhas e colunas passadas como parâmetros são válidas se não há números repetidos na submatriz. O “validaArray” também é usado aqui. Se a submatriz for válida, a função atualiza o “validaArray” na posição correspondente e encerra a thread.

Na Main busca-se o arquivo para a entrada, os threads são criados e iniciados usando dois loops aninhados para percorrer cada elemento. Para cada um, verifica-se se a linha e coluna são múltiplos de 3 (pois esses são os elementos que começam cada subseção 3x3) e se for, preenche as informações de linha e coluna e cria um thread. Faz-se a verificação das linhas e colunas também criando os threads.

Por fim, usa-se a função “pthread_join” para aguardar a finalização de todos os threads e retorna o resultado se o Sudoku é válido (todas as entradas do array “valido” ser iguais a 1) ou não (se tiver alguma entrada igual a 0).

Pseudocódigo

```
função "ehColunaValida"
validaArray: array [0..8] de inteiro

início função
params <- parametro
linha <- params->linha
col <- params->coluna
se (linha != 0 ou col > 8) então
    escreva ("Linha ou coluna inválida.)
    sair_thread
fim_se

para i de 0 até 8 faça
    num <- sudoku[i][col]
    se (num < 1 ou num > 9 ou validaArray[num - 1] == 1) então
        sair_thread
    senão
        validaArray[num - 1] <- 1
    fim_se
fim_para
valida[18 + col] <- 1
sair_thread
fim função
```

```
função "ehLinhaValida"
validaArray: array [0..8] de inteiro

início função
params <- parametro
linha <- params->linha
col <- params->coluna
se (col != 0 ou linha > 8) então
   escreva ("Linha ou coluna
inválida.)
   sair_thread
fim_se

para i de 0 até 8 faça
   num <- sudoku[linha][i]
   se (num < 1 ou num > 9 ou
validaArray[num - 1] == 1) então
      sair_thread
   senão
      validaArray[num - 1] <- 1
   fim_se
fim_para
valida[9 + linha] <- 1
sair_thread
fim função

início função principal
abrir arquivo
ler arquivo
fechar arquivo

para i de 0 até 8 faça
    para j de 0 até 8 faça
          se (resto de i/3 = 0 e j/3
= 0) então
              dado.linha<-i;

              dado.coluna<- j
               cria thread
          fimse
                se (i=0) então
                  colunaDados.linha<-i;

colunaDados.coluna<- j
                   cria thread
                fimse

                se (j=0) então
                  linhaDados.linha<-i;

                  linhaDados.coluna<- j
                   cria thread
                fim_se
fim_para
aguarda threads
para i de 0 até 8 faça
  se (valida[i]=0)
  escreva("sudoku invalido")
  senão
     escreva("sudoku valido")
fim_algoritmo
```

Entradas e Saídas

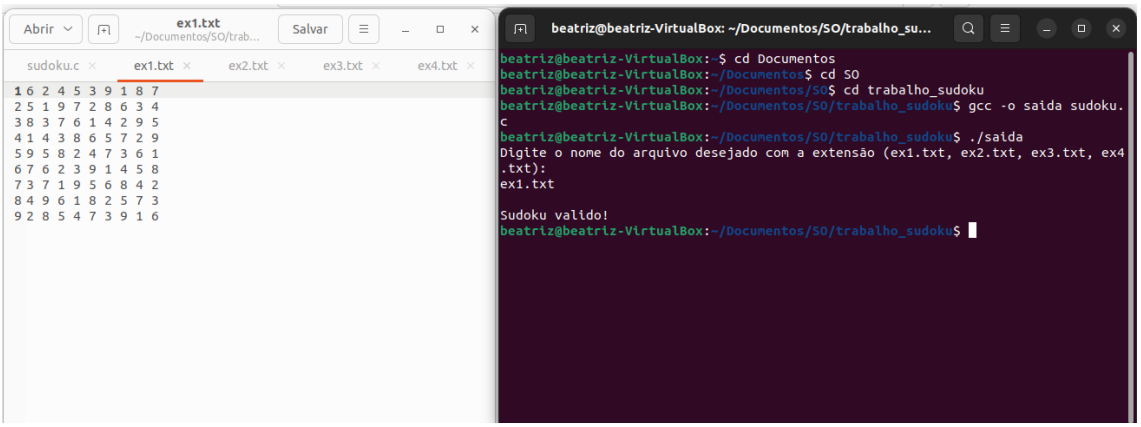

Figura 1: exemplo 1 com saída válida

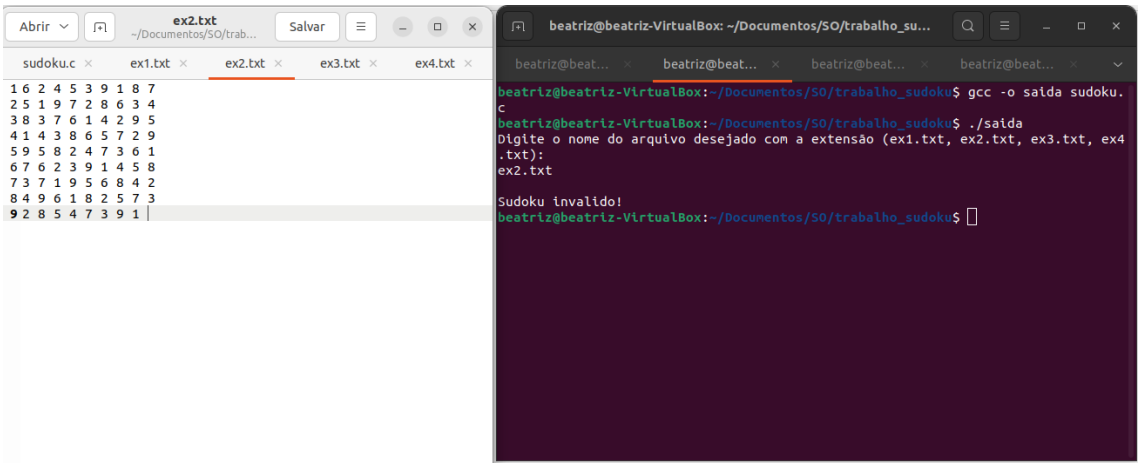

Figura 2: exemplo 2 com saída inválida

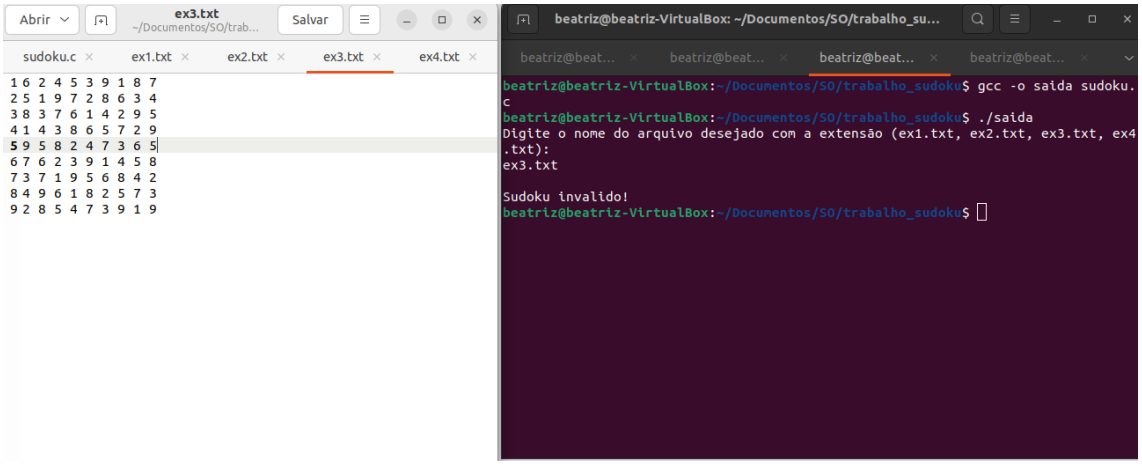

Figura 3: exemplo 3 com saída inválida

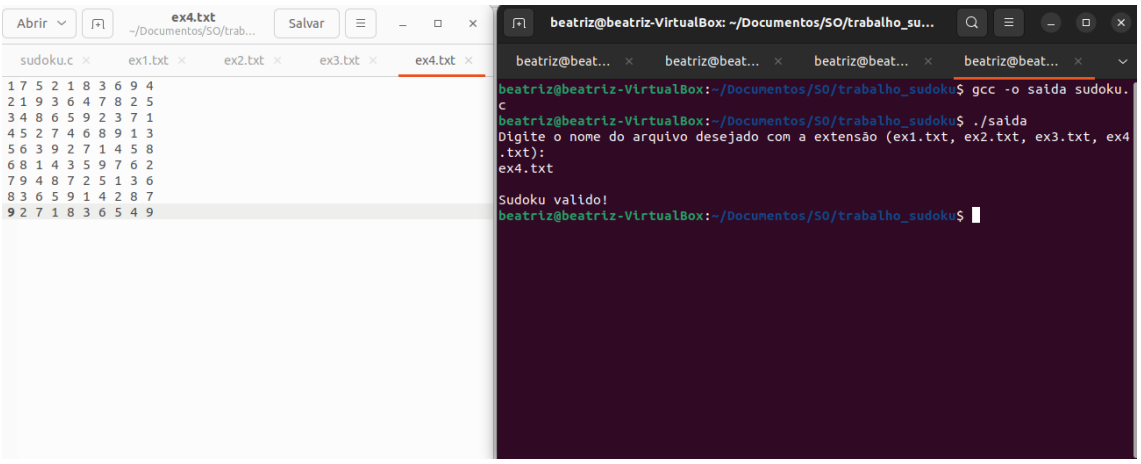

Figura 4: exemplo 4 com saída válida