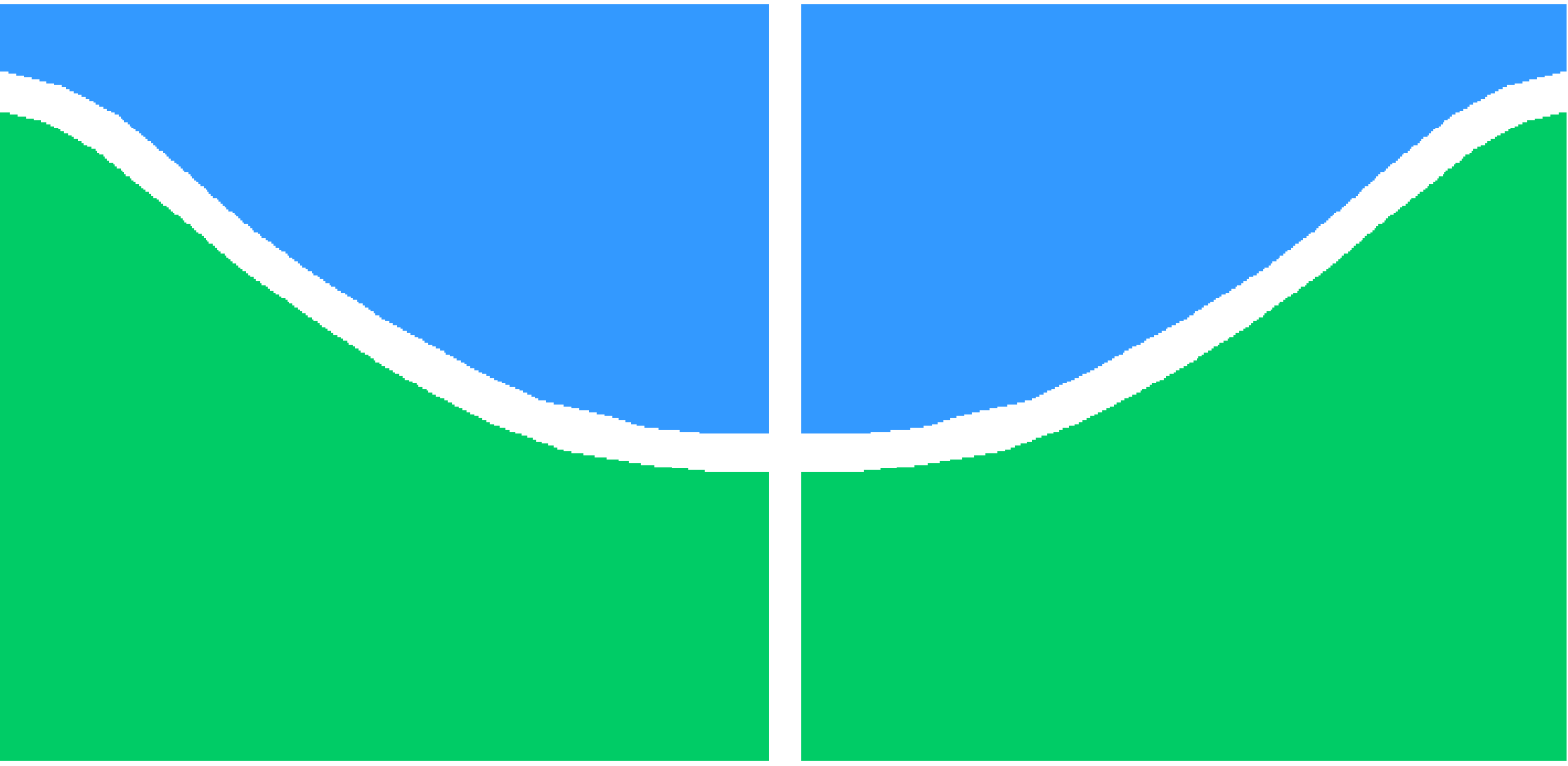

Universidade de Brasília - UnB
Faculdade UnB Gama - FGA
Fundamentos de Sistemas Operacionais

Trabalho 2

Autor: Maria Luciene Felix
Orientador: Prof. Dr. Tiago Alves

Brasília, DF
2016

Maria Luciene Felix

Trabalho 2

Trabalho submetido durante o curso de graduação em Engenharia de Software da Universidade de Brasília como requisito parcial para obtenção de nota da disciplina de Fundamentos de Sistemas Operacionais.

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Prof. Dr. Tiago Alves

Brasília, DF

2016

Sumário

1 Introdução

Este trabalho tem como objetivo consolidar os conhecimentos adquiridos na disciplina de Fundamentos de Sistemas Operacionais ofertada na Universidade de Brasília, Faculdade Gama.

1.1 Ambiente de Desenvolvimento

Para uma melhor realização do trabalho, decidiu-se por unificar, entre a dupla, as ferramentas de desenvolvimento utilizadas, as quais incluem sistema operacional, compilador, depurador e editor de texto. Portanto, o seguinte ambiente de desenvolvimento foi estabelecido:

A configuração do ambiente de desenvolvimento utilizado para a condução desse trabalho é listada a seguir:

- O sistema operacional utilizado foi o *Linux* na distro Elementary OS 0.3.2 Freya LTS;
- O código foi convertido com o compilador *GCC* na versão 4.8.4;
- O código foi depurado, quando necessário, com o debugger *GDB* na versão 7.7.1;
- A edição dos arquivos de código e texto foi realizada com o *Sublime*.

2 Implementação

Esta seção tem como objetivo detalhar as questões implementadas, fornecer informações de compilação e utilização. Adicionalmente apresenta casos de testes válidos e inválidos.

2.1 Questão 1

Para compilar essa questão foi utilizado um arquivo Makefile, no qual contém as regras de compilação. Para gerar o executável digite *make* no terminal e em seguida rode o programa com o comando *./alarm*.

O programa implementado comporta-se da seguinte maneira após sua execução:

Espera 5 segundos e então imprimi a mensagem **Alarm clock**.

Não foi identificada nenhuma inconsistência nessa questão.

2.2 Questão 2

Para a realização da questão 2 foram gerados dois programas. Ambos realizam o cálculo do valor máximo de uma sequência de inteiros. O primeiro chamado *main* utiliza-se de *threads* enquanto o segundo chamado *analise* foi implementado da maneira tradicional.

2.2.1 Primerio programa - main

Para gerar o executável do primeiro programa, *main*, rode no terminal o comando: make main. Ele ira produzir o arquivo *max*.

No terminal, execute o programa *./max* e informe: o tamanho da lista e a sequência de inteiros. Ficando assim:

- ./max tamanhoLista inteiro1 inteiro2 ... tamanhoLista-1

2.2.1.1 Caso de Teste: Valores válidos

Supondo que o usuário informe a seguinte entrada:

- ./max 4 3 2 7 9

o sistema irá imprimir:

- O tamanho da lista
- Os valores da lista
- O vetor W após a inicialização
- O vetor W após o passo 2
- O valor máximo
- E a posição do valor máximo

Resultado:

Number of input values = 4

Input values x = 3 2 7 9

After initialization w = 1 1 1 1

After Step 2

w = 0 0 0 1

Maximum = 9

Location = 3

2.2.1.2 Caso de Teste: Valores inválidos

Caso o usuário informe um número maior que 100 para o tamanho da lista, ou ainda, informe uma sequência de números maior que o tamanho da lista, o programa informa uma mensagem de erro e encerra o programa.

2.2.2 Segundo programa - analise

Gere o executável para o segundo programa com o comando: make analise. Ele irá produzir o arquivo compilado *alanise*.

Ele é executado da mesma forma que o primeiro programa, ficando assim:

- ./analise 4 3 2 7 9

O resultado produzido será:

Number of input values = 4 Input values x = 2 3 7 9 Max value: 9

2.2.3 Questão de Análise

QUESTÃO: Por que precisamos n(n-1)/2 em vez de n*n *threads*?

Gerando **n(n-1)/2** *threads* permite criar um número suficiente de *threads* capaz de realizar a comparação de dois inteiros de toda a lista apenas uma vez.

Comparação dos dois programas

Para realizar a comparação dos dois programas foi utilizada a função *time* do próprio sistema operacional. Ela fornece o tempo de execução do programa informado.

Para utilizá-la informe *time* no momento da execução do programa, da seguinte forma:

- time ./max 4 3 2 7 9
- time ./analise 4 3 2 7 9

O programa realizado com *threads* demorou 21 segundos, enquanto o programa tradicional levou apenas 19 segundos.

Ao implementarmos programas com *threads* temos a errônea impressão que consequentemente ele será mais rápido. Contudo, foram criadas várias *threads* para realizar uma operação simples. Tais *threads* consomem mais recursos do sistema, e por isso levam um tempo maior para executar a mesma operação.

2.2.4 Inconsistências identificadas

Na etapa 2 do primeiro programa, *main*, mais especificamente no método: **comparing_threads** foi gerado um número **n(n-1)/2** de *threads*, conforme requisitado na questão. No entanto, para realizar a comparação entre os números da lista, foi necessário utilizar duas estruturas *for*. Dessa forma, para uma lista de tamanho 4, será preciso 6 *threads*, porém, da maneira que foi implementado, cada uma realiza 6 comparações, produzindo um total de 36 comparações.

2.3 Questão 3

Para implementar a questão 3 foram gerados os programas q03a e q03b, o programa q03c não foi implementado. O programa q03a realiza o cálculo do produto entre duas matrizes utilizando apenas uma *thread*, enquanto o programa q03b realiza o mesmo cálculo com um maior número de *threads*.

Para gerar o arquivo executável da questão q03a, digite no terminal: make qa. Que irá gerar o executável *q03a*. Já para a questão q03b, rode o comando: make qb, gerando o arquivo *q03b*.

Para rodar os programas gerados informe o nome do executável mais o arquivo de entrada *int.txt*. Ficando da seguinte maneira:

- ./q03a < in.txt
- ./q03b < in.txt

Ambos irão produzir o mesmo resultado.

2.3.1 Caso de Teste: Valores válidos

Foi fornecido um arquivo de entrada chamado *in.txt* contendo entradas válidas. Nele é informado valores referentes a duas matrizes, sendo a primeria matriz 3x2 e a segunda 2x3.

Os programas irão calcular o produto de duas matrizes e imprimir os valores referentes a matriz resultado e as matrizes informadas. Ficando da seguinte maneira:

Dados da matriz A:

1 4

2 5

3 6

Dados da matriz B:

7 9 11

8 10 12

Matriz resultante:

39 49 59

54 68 82

69 87 105

2.3.2 Limitações

Ambos os programas gerados não fazem verificação quanto a entrada de dados, assim sendo, podem gerar resultados inesperados. Sua maior limitação está no fato de que eles suportam apenas a multiplicação de matrizes (3x2)*(2x3).

2.4 Referencial Teórico

O livro *Advanced linux programming* (MITCHELL; OLDHAM; SAMUEL, 2001) foi utilizado para auxiliar na aquisição de conhecimento, especialmente em relação a construção do *Makefile* e o sistema de *debugger*.

Referências

MITCHELL, M.; OLDHAM, J.; SAMUEL, A. *Advanced linux programming*. [S.l.]: New Riders, 2001. Citado na página 9.