

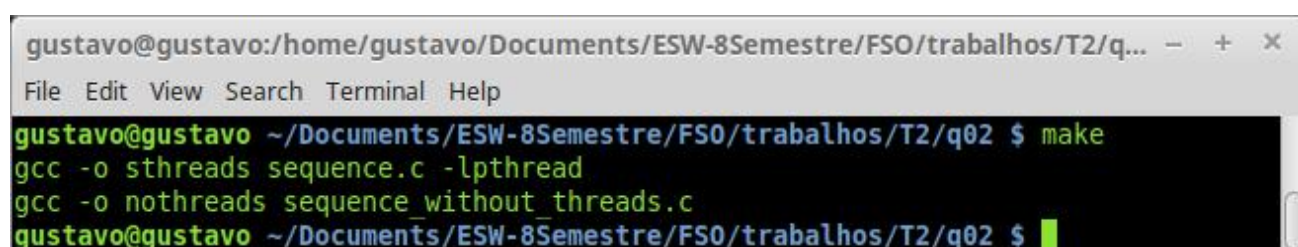
Nome: Gustavo Cavalcante Oliveira
Matrícula: 130009920

Questão 02 do Trabalho 02 de Fundamentos de Sistemas Operacionais.

1 - Instruções

1.1 - Compilação

Para gerar os programas executáveis, basta digitar o comando 'make' no diretório da questão 02. Gerando dois programas executáveis, um chamado sthreads e outro nothreads com e sem utilização de threads respectivamente.



```
gustavo@gustavo:/home/gustavo/Documents/ESW-8Semestre/FSO/trabalhos/T2/q... - + x
File Edit View Search Terminal Help
gustavo@gustavo ~/Documents/ESW-8Semestre/FSO/trabalhos/T2/q02 $ make
gcc -o sthreads sequence.c -lpthread
gcc -o nothreads sequence_without_threads.c
gustavo@gustavo ~/Documents/ESW-8Semestre/FSO/trabalhos/T2/q02 $
```

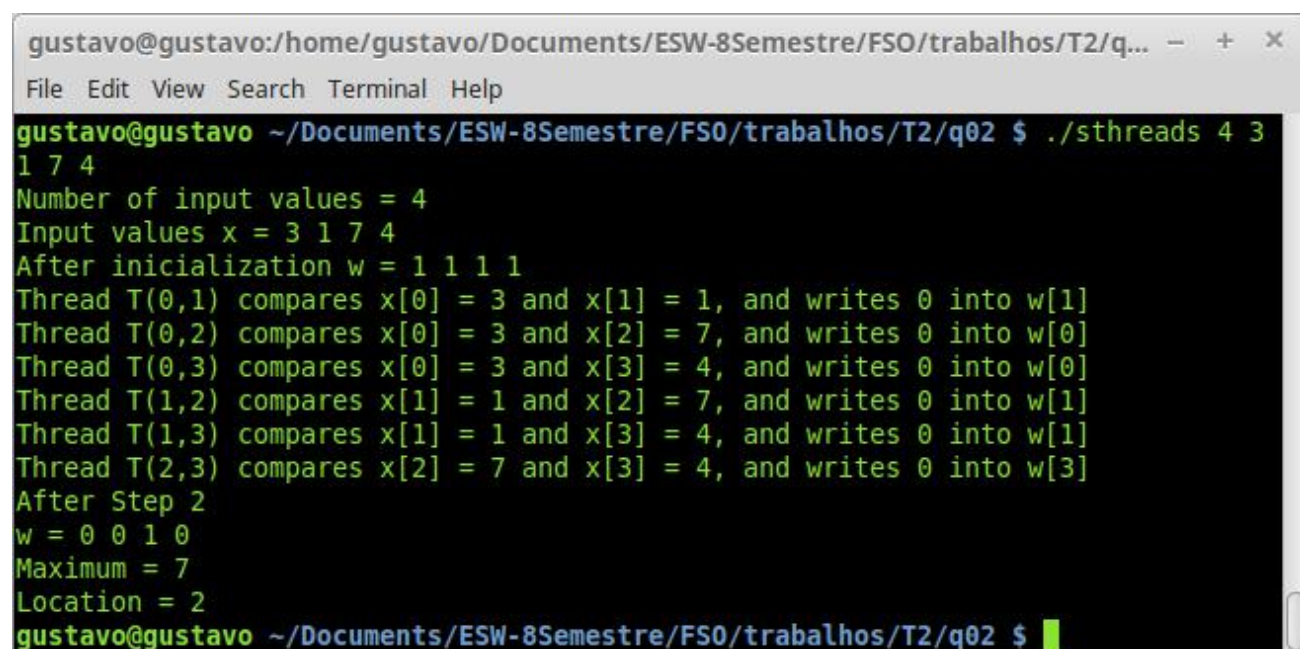
1.2 Executável

As instruções a seguir valem para os dois executáveis.

Para executar um dos programas, basta digitar ./programa N n1 n2 n3 n4 ... nN dentro do diretório da questão 02. Sendo que N é o número correspondente de inteiros que o programa irá tratar e n1, n2, ... , nN são os inteiros de fato.

Exemplo:

./sthreads 4 3 1 7 4



```
gustavo@gustavo:/home/gustavo/Documents/ESW-8Semestre/FSO/trabalhos/T2/q... - + x
File Edit View Search Terminal Help
gustavo@gustavo ~/Documents/ESW-8Semestre/FSO/trabalhos/T2/q02 $ ./sthreads 4 3
1 7 4
Number of input values = 4
Input values x = 3 1 7 4
After initialization w = 1 1 1 1
Thread T(0,1) compares x[0] = 3 and x[1] = 1, and writes 0 into w[1]
Thread T(0,2) compares x[0] = 3 and x[2] = 7, and writes 0 into w[0]
Thread T(0,3) compares x[0] = 3 and x[3] = 4, and writes 0 into w[0]
Thread T(1,2) compares x[1] = 1 and x[2] = 7, and writes 0 into w[1]
Thread T(1,3) compares x[1] = 1 and x[3] = 4, and writes 0 into w[1]
Thread T(2,3) compares x[2] = 7 and x[3] = 4, and writes 0 into w[3]
After Step 2
w = 0 0 1 0
Maximum = 7
Location = 2
gustavo@gustavo ~/Documents/ESW-8Semestre/FSO/trabalhos/T2/q02 $
```

2 - Limitações

2.1 - Número de inteiros:

É necessário que a quantidade de inteiros passadas como argumento estejam de acordo com o valor N, por exemplo, caso N seja 4, espera-se 4 inteiros seguintes. Caso seja passado um número de inteiros inferior a N, o programa irá ser abortado, caso seja passado um número superior, o programa irá ignorar os números excedentes.

Os valores de N devem estar necessariamente entre 0 e 100, caso seja passado um valor que não esteja nesse intervalo, o programa irá abortar.

2.2 - Limites:

2.2.1 - Variáveis

Os inteiros passados no argumento serão armazenados em variáveis do tipo int, caso seja passado números maiores que a capacidade dessa variável, o programa não irá executar de maneira apropriada.

2.2.2 - Memória

O programa não aloca memória dinamicamente de acordo com o valor de N, então é alocado a memória necessária pro caso de teste mais extremo, ou seja, $N = 100$.

3 - Questões:

3.1 - “Por que precisamos de $n(n-1)/2$ em vez de n^2 threads?”:

Pois $n(n-1)/2$ operações já são suficientes para deixar o vetor w com apenas uma posição com valor igual a 1.

3.2 - Questão de análise:

3.2.1 - Implemente o mesmo programa, porém usando abordagem sequencial (convencional), ou seja, não use threads na sua implementação. Verifique se há diferenças de desempenho observadas na busca pelo máximo ao se comparar o comportamento da implementação com threads com a implementação sequencial. Caso haja diferenças, explique-as.

Foi utilizado o programa time para marcar o tempo de execução do programa, com o seguinte caso de teste.

```
time ./sthreads 88 272 111 282 29 89 281 251 176 99 72 75 235 154 80 250 61 133 238
157 134 261 259 234 94 165 108 22 225 46 66 186 204 229 226 13 275 139 119 9 177 112
194 33 298 279 199 36 270 18 200 60 247 266 131 252 127 288 137 17 34 96 84 263 241
167 182 295 106 104 114 47 172 213 81 206 77 210 212 163 174 142 292 170 271 87 236
116 12 > out
```

```
time ./nothreads 88 272 111 282 29 89 281 251 176 99 72 75 235 154 80 250 61 133 238
157 134 261 259 234 94 165 108 22 225 46 66 186 204 229 226 13 275 139 119 9 177 112
194 33 298 279 199 36 270 18 200 60 247 266 131 252 127 288 137 17 34 96 84 263 241
167 182 295 106 104 114 47 172 213 81 206 77 210 212 163 174 142 292 170 271 87 236
116 12 > out
```

Foram feitos 10 testes com o mesmo caso de teste em cada programa, e os resultados estão registrados na tabela abaixo, levando em consideração somente o real time.

	sthreads	nothreads
T1	real0m0.067s	real0m0.003s
T2	real0m0.068s	real0m0.003s
T3	real0m0.071s	real0m0.003s
T4	real0m0.072s	real0m0.003s
T5	real0m0.069s	real0m0.003s
T6	real0m0.066s	real0m0.003s
T7	real0m0.072s	real0m0.003s
T8	real0m0.074s	real0m0.003s
T9	real0m0.070s	real0m0.003s
T10	real0m0.068s	real0m0.003s

Notou-se que a implementação sem utilização de threads é muito mais rápida, o que faz sentido, pois as operações que são executadas pelas threads são muito simples (poucas instruções), logo a chamada de um thread é muito mais custosa que a execução da operação de preencher o vetor w ou fazer a comparação entre os pares.

Adicionou-se o trecho de código abaixo modificação no final da função compare(void *pair) nos dois códigos fontes, sequence.c referente ao programa sthreads e sequence_without_threads.c referente ao programa nothreads.

```
gustavo@gustavo:/home/gustavo/Documents/E... - + x
File Edit View Search Terminal Help

int teste = 100000;
while(teste--);

125,1-8 89%
```

Com essa modificação, objetivemos os seguintes resultados, utilizando o mesmo caso de teste da comparação anterior.

	stthreads	nothreads
T1	real0m0.344s	real0m0.867s
T2	real0m0.367s	real0m0.871s
T3	real0m0.370s	real0m0.869s
T4	real0m0.351s	real0m0.872s
T5	real0m0.353s	real0m0.874s
T6	real0m0.372s	real0m0.868s
T7	real0m0.366s	real0m0.873s
T8	real0m0.360s	real0m0.873s
T9	real0m0.371s	real0m0.874s
T10	real0m0.360s	real0m0.872s

3.2.2 - “O que você sugeriria para otimizar essa implementação?”:

Como mostrado anteriormente na seção 3.2.1, não é nada vantajoso a utilização de threads no problema proposto pelo trabalho, uma possível solução para otimizar a implementação seria em vez de N chamadas de threads, fazer somente 4 chamadas de threads e dividir o intervalo do vetor em 4, cada thread trabalhando com uma parte desse intervalo, desde que o intervalo seja bem maior que 100.