FSO - Trabalho 2, Questão 2

> Alunos: Dylan Guedes e Artur Bersan
>
> Matrículas: 12/0115727 e 14/0016813

Sistema Operacional Utilizado

> Ubuntu Linux - Mx86_64 Kernel 3.13.0-32-generic
>
> Linux Fedora 24 - x86_64 Kernel 4.6.3-300

Ambiente de Desenvolvimento

> Computadores pessoais. Utilizamos o Linux, Makefile, git, valgrind e vim.

Instruções

Compilando

Para compilar, basta rodar o comando:

```
$ make
```

Utilização

Para utilizar o programa, o usuário deverá inserir o nome do executável gerado, o tamanho do array e em seguida, uma sequencia de inteiros

como parâmetros da linha de comando. Por exemplo:

```
$ ./a.out 4 3 1 7 4
```

O programa irá apresentar a quantidade de inputs os valores dos inputs, o array 'w' antes e depois da inicialização, as comparações entre os elementos, o maior valor e a posição dele.

Caso de Teste Válido

Um caso de teste válido para o programa seria, executar o binário com os seguintes argumentos:

```
$ ./a.out 6 1 20 3 40 5 60
```

Limitações conhecidas

Caso o usuário digitar um número muito grande, irá causar overflow.

Questões

> "Por que precisamos n(n-1)/2 em vez de n^2 threads?"

Para responder a essa pergunta, primeiro temos que perceber que o algortimo prega uma combinação em que o valor de 'P' = 2, seguindo a formula:

```
n!/p!*(n-p)!
```

Algebricamente falando:

- Passo 1:

```
n!/2!*(n-2)!
```

- Passo 2:

```
(n*(n-1)*(n-2)!)/(n-2)!*2!
```

- Passo 3:

Cancelando n-2! com n-2! tem-se como resultado:

```
n(n-1)/2
```

Resultados obtidos

Para executar esse passo, foi necessário compilar o programa 'sequencial.c' e rodar tanto o programa em questão quanto o main.c. Para compilar o sequencial.c:

```
gcc -o prog sequencial.c -std=c99
```

As instruções de uso do programa 'sequencial.c', são as mesmas do 'main.c'

Com o tamanho de array = 4:

- Com thread:

```
real  0m0.006s
user  0m0.002s
sys   0m0.006s
```

- Sem thread:

```
real  0m0.003s
user  0m0.001s
sys   0m0.002s
```

Com o tamanho de array = 10:

Com Thread:

```
real  0m0.010s
user  0m0.004s
sys   0m0.011s
```

Sem thread:

```
real  0m0.003s
user  0m0.001s
sys   0m0.002s
```

Com o tamanho de array = 20:

Com Thread:

```
real  0m0.024s
```

```
user  0m0.009s
sys   0m0.025s
```

Sem thread:

```
real  0m0.003s
user  0m0.001s
sys   0m0.002s
```

Análise dos Resultados

A não utilização de threads teve um desempenho melhor para o array de todos os tamanhos analisados. Isto se deve ao fato de que as operações realizadas no problema são relativamente simples e o overhead de criar threads, esperar, dar join em cada uma, dentre outras coisas, faz com que um tempo relativamente grande seja gasto gerenciando esse recurso.