Mini EP 6: Ordenação Dorminhoca

Alfredo Goldman, Elisa Silva e Luciana Marques
MAC 0219-5742 – Programação Concorrente e Paralela 2021

Entrega até 9 de junho de 2021

1. Introdução

A ordenação dorminhoca consiste na ideia de criar um atraso entre a leitura de um valor e sua impressão tal que esse atraso seja proporcional ao seu valor e assim ordenar uma lista de valores.

Uma forma de fazer isso é criando vários processos que dormem para então imprimirem o valor. Porém é claro que tal ordenação não garante saídas corretas sempre. Isso ocorre pelo fato de que quando um processo/thread dorme o SO só garante que ele não rodará antes do tempo definido.

Neste mini EP você irá brincar com um programa que implementa a ordenação dorminhoca a fim de encontrar o menor delay base possível tal que a saída ainda seja correta na maioria das vezes.

2. Tarefas

Baixe do eDisciplinas o arquivo **“**sleepSort.c”. Ele imprime de forma ordenada os números passados como argumento.

Para compilar você precisa definir o delay base que deve ser passado para o preprocessador, o delay base representa quanto tempo o processo vai dormir antes de imprimir o valor 1. O delay base é definido em microsegundos, então para um delay de 1 segundo por unidade o programa deve ser compilado dessa forma:

```
$ gcc -DDELAY=1000000 sleepSort.c -o ss1s
```

E assim para obter um programa que use um delay base de 200ms:

```
$ gcc -DDELAY=200000 sleepSort.c -o ss200ms
```

Exemplo de ordenação dos valores 5, 3, 2, 0 e 1:

```
$ ./ss200ms 5 3 2 0 1
```

Saída esperada:

```
0 1 2 3 5
```

Você deve escrever um programa verificador em qualquer linguagem que teste a corretude dos programas gerados a partir do código “sleepSort.c”.

O programa verificador deve receber o caminho para o executável e um número N. Ele deve verificar N vezes se o executável é capaz de ordenar uma permutação aleatória dos números de 1 à 100. Gere uma permutação aleatória para cada verificação. Se sua linguagem prover métodos de embaralhar os números, pode usar, senão recomendamos implementar o algoritmo de embaralhamento de Fisher-Yates. Ao final dos N testes, o programa verificador deve imprimir “Ok” se o programa verificados produziu a saída correta pelo menos 90% das vezes, senão ele deve imprimir “Not Ok”.

Então você deve buscar o menor delay tal que o programa gerado passe no programa verificador para um N de 200. Uma dica é começar com um número menor de testes (20 por exemplo) e começar com um delay de 100ms (100000 microsegundos).

3. Entrega

Envie em um arquivo .ZIP com o seu número USP como nome o código fonte do seu programa verificador e um arquivo de texto contendo seu nome, menor delay encontrado e sua resposta para a pergunta "Dado dois computadores similares, porém um com o dobro de núcleos e outro com o dobro da frequência da CPU, em qual deles se obteria o menor valor para o delay base?".

Entrega até 9 de junho de 2021.

4. Critério de Avaliação

Os Mini EPs usam um critério de avaliação binária (ou 1 ou 0). Para tirar 1 envie .ZIP conforme especificado no eDisciplinas. Não será avaliada a qualidade do código, mas tenha zelo.

Vale reforçar parágrafo II do artigo 23 do **Código de Ética da USP**:

> Artigo 23 - É vedado aos membros do corpo discente e demais alunos da Universidade:
> [...]
> II. lançar mão de meios e artifícios que possam fraudar a avaliação do desempenho, seu ou de outrem, em atividades acadêmicas, culturais, artísticas, desportivas e sociais, no âmbito da Universidade, e acobertar a eventual utilização desses meios.

Mini EPs plagiados receberão nota 0.

Se tiver dúvidas, envie uma mensagem no fórum do curso ou envie e-mails para elisa@silva.moe, lucianadacostamarques@gmail.com ou gold@ime.usp.br com *[miniEP6]* no assunto do e-mail. Divirta-se!