

Relatório de Laboratório 1 - ICP361

A professora disponibilizou um código em C pronto para explicar as principais funções de controle de threads.

<https://s3-us-west-2.amazonaws.com/secure.notion-static.com/9c6c67a5-1a02-4236-9b2d-872f9b134fee/helloC.c>

E com isso, ela nos deu um roteiro para o lab1.

<https://s3-us-west-2.amazonaws.com/secure.notion-static.com/227b0bc4-e260-408a-8845-3887bee6a9b6/lab1.pdf>

Com

```
pthread_t thread[10]
```

podemos criar 10 threads, mas essa função em si não as cria ainda, só anuncia quantas threads serão.

A função que propriamente cria as threads é

```
pthread_create(a,b,c,d)
```

onde no argumento “a” passamos a thread por referência, “b” é um atributo mas que por enquanto vamos usar apenas NULL, “c” é uma rotina que vai ser executada na criação da thread, como se fosse a main da thread, e “d” é um ponteiro como por exemplo o id da thread.

Se o argumento “c” de `pthread_create` é como se fosse uma “main” da thread, então `pthread_exit()` é o seu return. (Comentários onde estou apenas associando as coisas a algo que já conheço, pode ser algo muito além disso.)

E para nos garantir de que as threads não vão ficar sempre abertas, conseguimos escolher que elas parem com

```
pthread_join(threads[thread_id], NULL)
```

Então com tudo isso de função nas nossas mãos, nos foi pedido o seguinte:

Implemente o seu primeiro programa concorrente! Escreva um programa com duas threads, para multiplicar por 2 cada elemento de um vetor de 10000 elementos. (Para cada elemento i do vetor, calcular o novo valor e escrever o resultado na mesma posição do elemento.)

A primeira forma que eu pensei de fazer isso é dividindo nosso array em 2, do item 0 ao 4999 e do item 5000 ao 9999. E assim cada thread vai multiplicar a mesma quantidade de elementos em paralelo. Mas aí que a gente pensa “hmm aqui se trata de 10000, mas e se fosse 10001, já seria problemático, e eu teria que modificar o programa. Mas nem é um problema tão grande, eu vou usar a parte inteira da divisão que vai matar esse problema.

A thread 0 vai cuidar os itens até a metade do array, se for 10 itens no array, ele vai de 0 a 4, se for 11 ele vai de 0 a 4 ainda, porque `array_size/2` é 5, para 11 e para 10 em C, veja como ficou :

```
for(int i = 0; i < array_size/2; i++){  
    array[i] = 2 * array[i];  
}
```

E para a thread 1, eu continuo da metade pra frente:

```
for(int i = array_size/2; i < array_size; i++) {  
    array[i] = 2 * array[i];  
}
```

Bem simples, e tudo funciona.