Computação concorrente - Lab 1

Matheus Veras Mondaini - DRE 119030209

1-

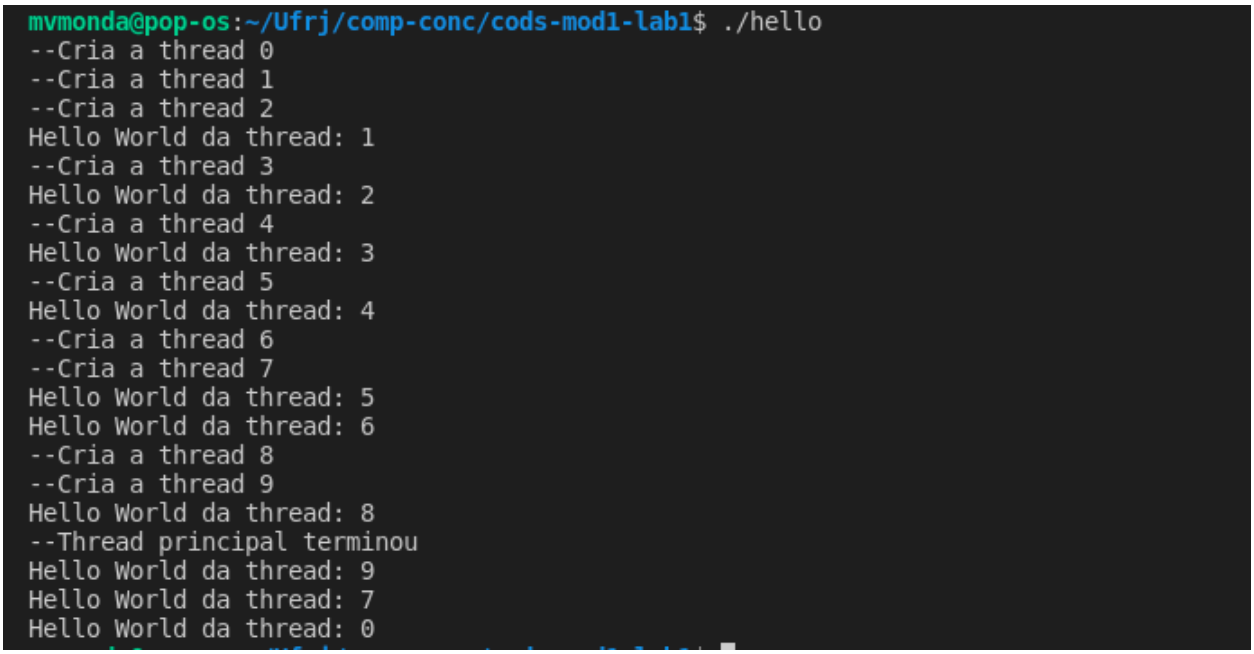
```
mvmonda@pop-os:~/Ufrj/comp-conc/cods-mod1-lab1$ ./hello
--Cria a thread 0
--Cria a thread 1
Hello World
--Cria a thread 2
Hello World
--Cria a thread 3
Hello World
--Cria a thread 4
Hello World
--Cria a thread 5
Hello World
--Cria a thread 6
--Cria a thread 7
Hello World
Hello World
--Cria a thread 8
Hello World
--Cria a thread 9
Hello World
--Thread principal terminou
Hello World
```

Em múltiplas execuções, houve diferenças na ordem de impressão do "Hello World" das threads. Isso ocorre pois as threads são disparadas para execução em paralelo entre e si e em relação ao programa principal. Não há garantia de que a ordem da impressão é a mesma ordem do disparo das threads.

2-

```
mvmonda@pop-os:~/Ufrj/comp-conc/cods-mod1-lab1$ ./hello
--Cria a thread 0
--Cria a thread 1
--Cria a thread 2
Hello World da thread: 0
Hello World da thread: 1
Hello World da thread: 2
--Cria a thread 3
--Cria a thread 4
Hello World da thread: 3
--Cria a thread 5
Hello World da thread: 4
--Cria a thread 6
Hello World da thread: 5
--Cria a thread 7
Hello World da thread: 6
--Cria a thread 8
Hello World da thread: 7
--Cria a thread 9
Hello World da thread: 8
--Thread principal terminou
Hello World da thread: 9
```

```
mvmonda@pop-os:~/Ufrj/comp-conc/cods-mod1-lab1$ ./hello
--Cria a thread 0
--Cria a thread 1
Hello World da thread: 0
--Cria a thread 2
Hello World da thread: 1
--Cria a thread 3
Hello World da thread: 2
--Cria a thread 4
Hello World da thread: 3
--Cria a thread 5
Hello World da thread: 4
Hello World da thread: 5
--Cria a thread 6
--Cria a thread 7
Hello World da thread: 6
--Cria a thread 8
Hello World da thread: 7
--Cria a thread 9
Hello World da thread: 8
--Thread principal terminou
Hello World da thread: 9
```

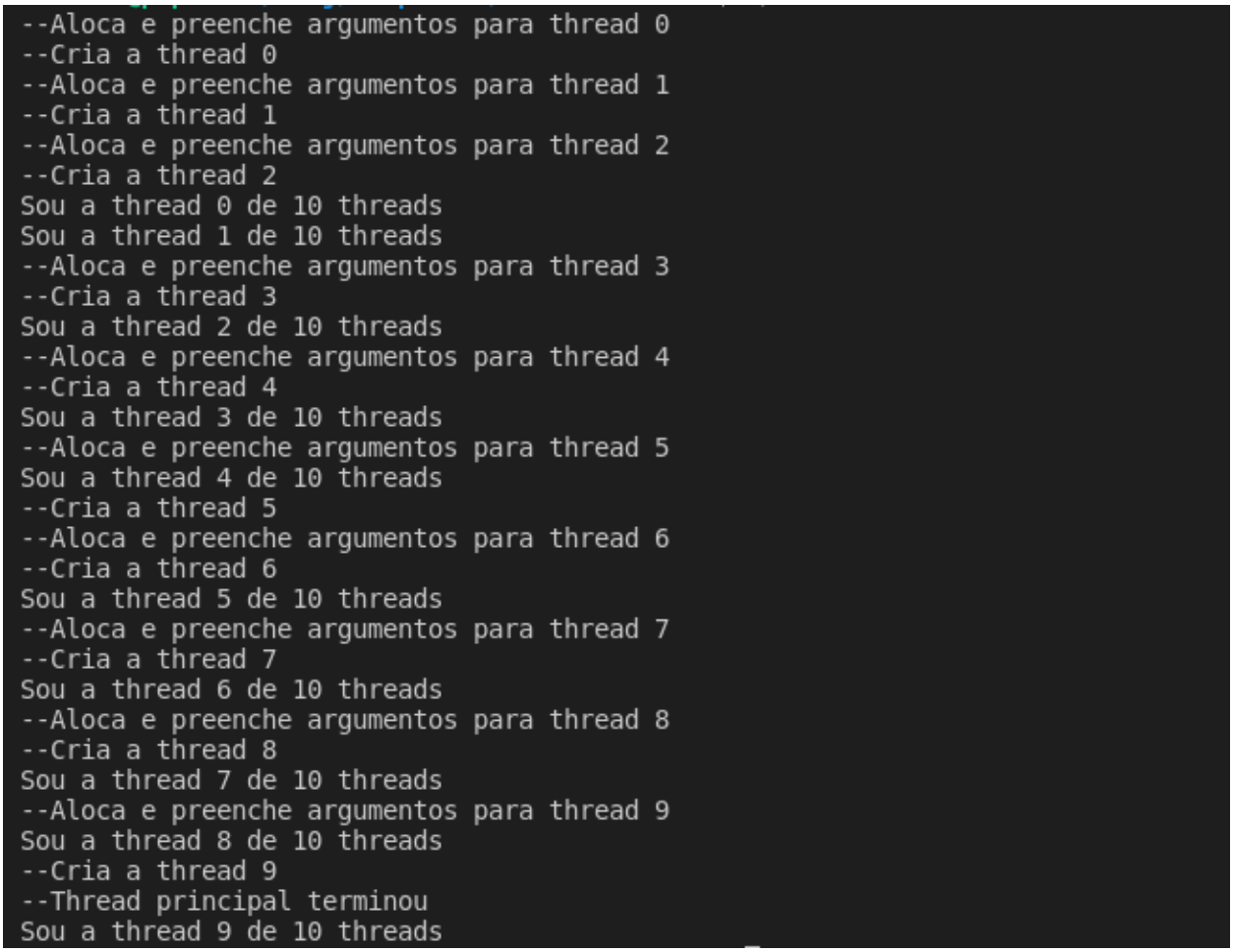


Nesse programa, temos a identificação do número de cada thread e podemos verificar a ordem em que foram executadas. Como pudemos verificar na terceira imagem, não há garantia de que a ordem de disparo seja respeitada na ordem de execução do sistema operacional.

No código, é possível ver como o argumento é passado para cada thread, utilizando um ponteiro genérico como argumento

3-

```
mvmonda@pop-os:~/Ufrj/comp-conc/cods-mod1-lab1$ ./hello
--Aloca e preenche argumentos para thread 0
--Cria a thread 0
--Aloca e preenche argumentos para thread 1
--Cria a thread 1
Sou a thread 0 de 10 threads
--Aloca e preenche argumentos para thread 2
--Cria a thread 2
Sou a thread 1 de 10 threads
--Aloca e preenche argumentos para thread 3
--Cria a thread 3
Sou a thread 2 de 10 threads
--Aloca e preenche argumentos para thread 4
--Cria a thread 4
Sou a thread 3 de 10 threads
--Aloca e preenche argumentos para thread 5
--Cria a thread 5
Sou a thread 4 de 10 threads
--Aloca e preenche argumentos para thread 6
--Cria a thread 6
Sou a thread 5 de 10 threads
--Aloca e preenche argumentos para thread 7
--Cria a thread 7
Sou a thread 6 de 10 threads
--Aloca e preenche argumentos para thread 8
--Cria a thread 8
Sou a thread 7 de 10 threads
--Aloca e preenche argumentos para thread 9
--Cria a thread 9
Sou a thread 8 de 10 threads
--Thread principal terminou
Sou a thread 9 de 10 threads
```



Sim, funcionou como esperado e demonstrou que podemos utilizar qualquer ponteiro como argumento da thread, incluindo structs customizadas.

4-

```
mvmonda@pop-os:~/Ufrj/comp-conc/cods-mod1-lab1$ ./hello
--Aloca e preenche argumentos para thread 0
--Cria a thread 0
--Aloca e preenche argumentos para thread 1
--Cria a thread 1
Sou a thread 0 de 10 threads
--Aloca e preenche argumentos para thread 2
--Cria a thread 2
Sou a thread 1 de 10 threads
--Aloca e preenche argumentos para thread 3
Sou a thread 2 de 10 threads
--Cria a thread 3
--Aloca e preenche argumentos para thread 4
--Cria a thread 4
Sou a thread 3 de 10 threads
--Aloca e preenche argumentos para thread 5
--Cria a thread 5
Sou a thread 4 de 10 threads
--Aloca e preenche argumentos para thread 6
--Cria a thread 6
Sou a thread 5 de 10 threads
--Aloca e preenche argumentos para thread 7
--Cria a thread 7
Sou a thread 6 de 10 threads
--Aloca e preenche argumentos para thread 8
--Cria a thread 8
Sou a thread 7 de 10 threads
--Aloca e preenche argumentos para thread 9
--Cria a thread 9
Sou a thread 8 de 10 threads
Sou a thread 9 de 10 threads
--Thread principal terminou
```

```
mvmonda@pop-os:~/Ufrj/comp-conc/cods-mod1-lab1$ ./hello
--Aloca e preenche argumentos para thread 0
--Cria a thread 0
--Aloca e preenche argumentos para thread 1
--Cria a thread 1
Sou a thread 0 de 10 threads
--Aloca e preenche argumentos para thread 2
--Cria a thread 2
Sou a thread 1 de 10 threads
--Aloca e preenche argumentos para thread 3
--Cria a thread 3
Sou a thread 2 de 10 threads
--Aloca e preenche argumentos para thread 4
--Cria a thread 4
Sou a thread 3 de 10 threads
--Aloca e preenche argumentos para thread 5
--Cria a thread 5
Sou a thread 4 de 10 threads
--Aloca e preenche argumentos para thread 6
--Cria a thread 6
Sou a thread 5 de 10 threads
--Aloca e preenche argumentos para thread 7
--Cria a thread 7
Sou a thread 6 de 10 threads
--Aloca e preenche argumentos para thread 8
--Cria a thread 8
Sou a thread 7 de 10 threads
--Aloca e preenche argumentos para thread 9
--Cria a thread 9
Sou a thread 8 de 10 threads
Sou a thread 9 de 10 threads
--Thread principal terminou
mvmonda@pop-os:~/Ufrj/comp-conc/cods-mod1-lab1$
```

A diferença para os programas anteriores é que o término da thread principal("Thread principal terminou") ocorre sempre por último na execução do programa, pois há um chamada para thread_join que aguarda o final da execução do array de threads criadas.

5-