Atividade 1

Objetivo: Mostrar como criar um programa concorrente em C usando a biblioteca *pthread*.

Há mudanças na ordem de execução das threads? Por que isso ocorre?

Sim. Pois a ordem de execução depende do escalonamento das threads realizado pelo sistema operacional, que pode mudar a cada vez que o programa é rodado.

Atividade 2

Objetivo: Mostrar como passar um argumento para uma thread.

Qual foi a diferença em relação ao programa anterior? Por que foi necessário alocar espaço de memória para o argumento de cada thread?

As diferenças são:

- a função *PrintHello* passa a utilizar o valor de arg;
- a função principal agora passa um parâmetro para a thread;
- a função principal agora reserva um espaço para os identificadores das threads.

A alocação de espaço para o argumento das threads se deu pois é necessário guardar uma cópia do identificador da thread, já que a cada iteração ele é atualizado.

Atividade 3

Objetivo: Mostrar como passar mais de um argumento para uma thread.

O programa funcionou como esperado?

Sim. Cada thread imprimiu seu identificador e o número total de threads. Por ser um programa concorrente, não podemos garantir a ordem com que cada thread imprime e como não há um *pthread_join*, a thread principal pode imprimir antes das thread filhas terminarem.

Atividade 4

Objetivo: Mostrar como fazer a thread principal (*main*) aguardar as outras threads terminarem.

O que aconteceu de diferente em relação às versões/execuções anteriores?

Dessa vez, devido a função *pthread_join*, a função principal aguardou a finalização das outras threads para só então imprimir sua mensagem e finalizar.