Módulo 1 - Laboratório 1
Introdução ao uso de threads em C

Computação Concorrente (MAB-117) - 2020.1 REMOTO
Prof. Silvana Rossetto

[1]DCC/IM/UFRJ

Introdução

O objetivo deste Laboratório é aprender como criar programas concorrentes em C, usando as funções principais da biblioteca *Pthread*. As seguintes funções serão usadas:

- int **pthread_create** (pthread_t *thread, const pthread_attr_t *attr, void *(*start_routine) (void *), void *arg); (retorna 0 no caso de sucesso)
- void **pthread_exit** (void *retval);
- int **pthread_join** (pthread_t thread, void **retval); (retorna 0 no caso de sucesso)

Acompanhe a explanação da professora na vídeo-aula deste laboratório. Se tiver dúvidas, entre em contato por email.

Para cada atividade, siga o roteiro proposto e responda às questões colocadas.

Considera-se que os programas serão executados em um terminal Linux.

Atividade 1

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

Roteiro:

1. Abra o arquivo **hello.c**, leia o código do programa e tente entender o que ele faz.
2. Compile o programa *hello.c* acrescentando a opção **-lpthread** na linha de comando (a opção `lpthread` acrescenta as funções da biblioteca *pthread*) (ex.: `gcc -o hello hello.c -lpthread`).
3. Execute o programa **várias vezes** (ex.: `./hello`) e observe os resultados impressos na tela. **Há mudanças na ordem de execução das threads? Por que isso ocorre?**

Atividade 2

Objetivo: Mostrar como passar um argumento para uma thread.

Roteiro:

1. Abra o arquivo **hello_arg.c**, leia o código do programa e tente entender o que ele faz.
2. Compile o programa acrescentando a opção **-lpthread** na linha de comando.
3. Execute o programa **várias vezes** e observe os resultados impressos na tela. **Qual foi a diferença em relação ao programa anterior? Por que foi necessário alocar espaço de memória para o argumento de cada thread?**

Atividade 3

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

Roteiro:

1. Abra o arquivo **hello_args.c**, leia o código do programa e tente entender o que ele faz.
2. Compile o programa acrescentando a opção **-lpthread** na linha de comando.
3. Execute o programa **várias vezes** e observe os resultados impressos na tela. **O programa funcionou como esperado?**

Atividade 4

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

Roteiro:

1. Abra o arquivo **hello_join.c**, leia o código do programa e tente entender o que ele faz.
2. Compile o programa acrescentando a opção **-lpthread** na linha de comando.
3. Execute o programa **várias vezes** e observe os resultados impressos na tela. **O que aconteceu de diferente em relação às versões/execuções anteriores?**

Atividade 5

Objetivo: Implementar um programa concorrente, com **duas** threads (além da thread principal), para incrementar de 1 cada elemento de um vetor de N ($10 < N < 100$) elementos.

Roteiro:

1. Comece pensando em como dividir a tarefa de incrementar todos os elementos do vetor entre duas threads.
2. Qual(is) argumento(s) deverá(ão) ser passado(s) para a função executada pelas threads?
3. Na função/thread *main* faça a inicialização do vetor e imprima seus valores iniciais.
4. Implemente a tarefa de cada thread.
5. Na função/thread *main* imprima os valores finais do vetor.
6. Teste seu programa.

Entrega do laboratório: Disponibilize o código implementado na **Atividade 5** em um ambiente de acesso remoto (GitHub ou GitLab). Use o formulário de entrega desse laboratório para enviar o link do repositório do código implementado.