

Laboratório 1

Introdução à programação concorrente em C (com threads)

Programação Concorrente (ICP-361) - 2024.1
Profa. Silvana Rossetto

¹Instituto de Computação/UFRJ

Introdução

O objetivo deste Laboratório é apresentar os conceitos mais básicos de programação concorrente (com threads), criando e avaliando programas concorrentes em C. As seguintes funções serão usadas (em C):

- 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)

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

Considera-se que os programas serão executados em um terminal Linux. Quem estiver usando Windows, provavelmente precisará configurar o ambiente de programação C para incluir a biblioteca pthread.h.

Atividade 1

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

Roteiro:

1. Abra o arquivo **helloC.c**, leia o código do programa e tente entender o que ele faz. **O que será impresso na tela quando esse programa for executado? Acompanhe a explicação da professora.**
2. Compile o programa *helloC.c* acrescentando a opção **-lpthread** e **-Wall** na linha de comando (a opção *lpthread* acrescenta as funções da biblioteca *pthread*) (ex.: `gcc -o helloC helloC.c -lpthread -Wall`).
3. Execute o programa **várias vezes** (ex.: `./helloC`) e observe os resultados impressos na tela.
4. **Os resultados estão de acordo com o esperado? Observe se há mudanças na ordem de execução das threads e tente entender porque isso pode acontecer. Acompanhe a explicação da professora.**

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** e **-Wall** na linha de comando.
3. Execute o programa **várias vezes** e observe os resultados impressos na tela. **Observe a diferença em relação ao programa anterior.**
4. **Por que foi necessário usar `long int` na variável iteradora?** Acompanhe a explicação da professora.

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** e **-Wall** na linha de comando.
3. Execute o programa **várias vezes** e observe os resultados impressos na tela. **Verifique se o programa funcionou como esperado.**
4. **Por que foi necessário criar uma estrutura de dados nova?** Acompanhe a explicação da professora.

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** e **-Wall** na linha de comando.
3. Execute o programa **várias vezes** e observe os resultados impressos na tela.
4. **O que aconteceu de diferente em relação às versões/execuções anteriores?** Acompanhe a explicação da professora.

Atividade 5

Objetivo: Implementar o seu primeiro programa concorrente! Escreva um programa com M threads, para **eleva** ao **quadrado** cada elemento de um vetor de N elementos do tipo **inteiro**. (Para cada elemento a_i do vetor, calcular o novo valor e escrever o resultado na mesma posição do elemento.)

Roteiro:

1. Comece pensando em como dividir a tarefa de elevar ao quadrado todos os elementos de um vetor entre M threads. **Importante: todas as threads deverão executar a mesma função.** Qual(is) argumento(s) deverá(ão) ser passado(s) para a função executada pelas threads?
2. Implemente funções separadas para inicializar e carregar o vetor e verificar se o resultado está correto no final.
3. Na função *main*, chame a função de inicialização do vetor; crie as threads; aguarde o término da execução das threads criadas e **chame a função para verificar se os valores finais do vetor estão corretos.**
4. Execute o programa várias vezes e verifique se ele está funcionando corretamente.

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](#) e as respostas das [questões propostas](#).