Laboratório 1
Introdução à programação concorrente em C (com threads)

Programação Concorrente (ICP-361) - 2025-2
Profa. Silvana Rossetto

[1]Instituto de Computação/CCMN/UFRJ

Introdução

O objetivo deste Laboratório é apresentar os conceitos mais básicos de programação concorrente (com threads), criando e observando o comportamento de 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, leia com atenção tudo que está sendo pedido/explicado, 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 e executar um programa concorrente em C, usando a biblioteca `pthread`.

Roteiro:

1. Escreva o programa junto com a professora.
2. Compile o programa *hello.c* na linha de comando fazendo: `gcc -o hello hello.c -Wall`).
3. Execute o programa **várias vezes** (ex.: `./hello`), **mantendo e alterando** a quantidade de threads. Observe o log impresso na tela. **Ele muda de uma execução para outra?**
4. **Os resultados estão de acordo com o esperado?**
   Acompanhe a explanação da professora.

Atividade 2

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

Roteiro:

1. Junto com a professora, altere o programa da atividade anterior para que ele receba e imprima além do ID da thread, o número total de threads do programa.
2. Execute o programa **várias vezes** e observe os resultados impressos na tela. **Verifique se o programa funcionou como esperado.**
3. **Por que foi necessário criar uma estrutura de dados nova?**
   Acompanhe a explanação da professora.

Atividade 3

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

Roteiro:

1. Abra o arquivo **helloJoin.c**, leia o código do programa e tente entender o que ele faz.
2. Compile o programa na linha de comando fazendo: `gcc -o joinhello helloJoin.c -Wall`).
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? Por que?** Acompanhe a explanação da professora.

Atividade 4

Objetivo: Implementar o seu primeiro programa concorrente! Escreva um programa concorrente que execute com **4** *threads* para **incrementar de 1** cada elemento de um vetor de **4*N** elementos do tipo **inteiro**. Para cada elemento a_i do vetor, calcular o novo valor $(a_i + 1)$ e escrever o resultado na mesma posição do elemento.

A tarefa completa deverá ser dividida entre as threads de forma balanceada (as threads devem receber a mesma carga de trabalho). **O valor de N deve ser informado na chamada do programa**.

Roteiro:

1. Comece pensando em como dividir a tarefa completa entre as 4 threads. **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 o vetor de entrada 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. Lembre-se de executar o programa várias vezes e verificar se ele está funcionando corretamente.