# Laboratório 5
# Sincronização por condição usando variáveis de condição

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

[1]IC/CCMN/UFRJ

## Introdução

O objetivo deste Laboratório é introduzir o mecanismo de **sincronização por condição** usando **variáveis de condição** da biblioteca Pthread.

## Atividade 1

**Objetivo:** Experimentar o padrão de **sincronização coletiva** (barreira).

**Roteiro:**

1. Abra o arquivo `barreira.c`. Ele apresenta uma implementação do padrão barreira e um exemplo simples de uso. Acompanhe a explanação da professora.
2. Execute o programa e **verifique seus resultados**. **As threads estão executando de forma sincronizada, concluindo uma interação para então iniciar outra?**
3. Descomente a linha 43 (que chama a 'barreira'). Execute novamente o programa e **avalie os resultados**.

## Atividade 2

**Objetivo:** Mostrar um exemplo inicial de uso de **variáveis de condição** para controlar a ordem de execução das threads de um programa.

**Roteiro:**

1. Abra o arquivo `hellobye.c` e **identifique qual é o requisito lógico/condicional da aplicação** (qual é a ordem de impressão requerida para as expressões HELLO e BYEBYE). Acompanhe a explanação da professora.
2. Execute a aplicação várias vezes e verifique se o requisito lógico é sempre cumprido.
3. Inverta a ordem de criação das threads A e B e execute o programa novamente. Verifique os resultados.

## Atividade 3

**Objetivo:** Incrementar o exemplo anterior colocando agora **duas** threads A e fazendo a thread B esperar ambas executarem "hello".

**Roteiro:**

1. Abra o arquivo `hello2bye.c` e **identifique qual é o requisito lógico/condicional da aplicação** (qual é a ordem de impressão requerida para as expressões HELLO e BYEBYE). Acompanhe a explanação da professora.
2. Execute a aplicação várias vezes e verifique se o requisito lógico é sempre cumprido.
3. Inverta a ordem de criação das threads A e B e execute o programa novamente.

## Atividade 4

**Objetivo:** Incrementar o exemplo anterior colocando agora **duas** threads B (ambas devem esperar as duas threads A executarem "hello").

**Roteiro:**

1. é com você agora!!
2. Execute a aplicação várias vezes e verifique se o requisito lógico é sempre cumprido.
3. Inverta a ordem de criação das threads A e B e execute o programa novamente.

## Atividade 5

**Objetivo:** Retomar exercício da aula anterior e implementar um requisito condicional de execução das threads.

**Roteiro:**

1. Revisite o programa `soma-lock-atom.c` avaliado no Lab4.
2. Altere a implementação da thread `extra` de modo a garantir que ela imprima todos os valores de `soma` que são múltiplos de 1000. O objetivo é fazer a thread `ExecutaTarefa` pausar sua execução quando um múltiplo de 1000 é alcançado e somente continuar depois que o seu valor for impresso pela thread `extra`.
3. Execute o programa e confirme que todos os requisitos foram atendidos.

**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 responder às questões propostas.