

# Sistemas Hardware-Software

Aula 19 - Introdução a sincronização

**Engenharia**

Fabio Lubacheski

Maciel C. Vidal

Igor Montagner

Fábio Ayres

# Comunicação entre as Threads

- As threads podem trabalhar **cooperativamente** na resolução de um problema;
- As variáveis globais são utilizadas para trocar informações e influenciar na execução das outras threads.

# Tarefas paralelas

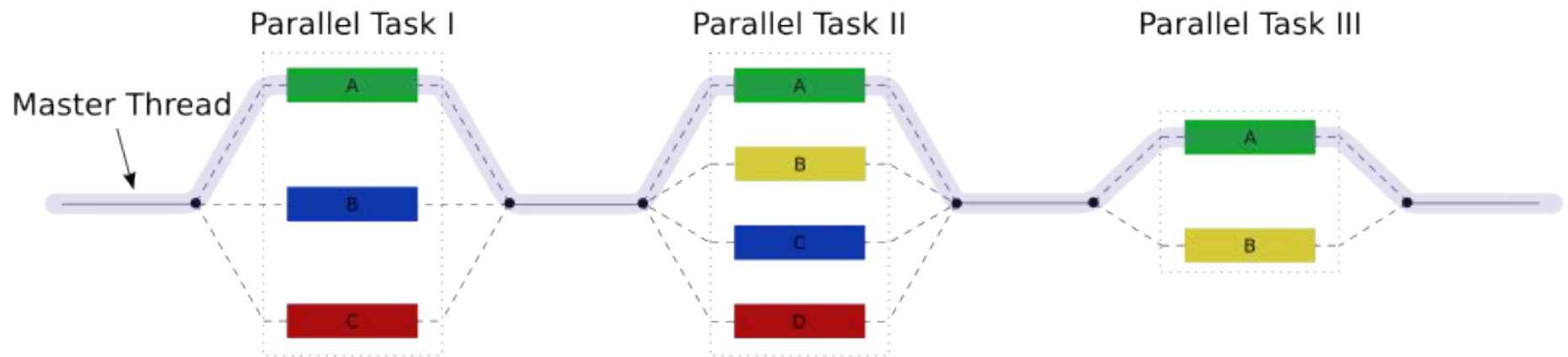
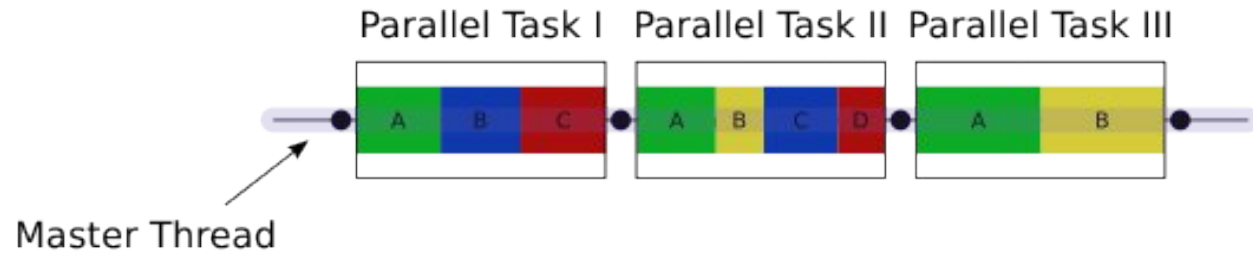


Figura: [https://en.wikipedia.org/wiki/File:Fork\\_join.svg](https://en.wikipedia.org/wiki/File:Fork_join.svg)

# POSIX threads

O padrão POSIX define também uma API de threads (*pthread*) que inclui

- Criação de threads
- Controle a acesso de dados (usando **mutex Semáforos Binários**)
- Sincronização (usando **Semáforos Contadores**)



# Atividade prática

## Aquecimento (20 min)

1. Utilização da API pthreads
2. Dividir uma tarefa em pedaços para executar.

# Conceito : Região Crítica

"Parte do programa que só pode ser rodada uma thread por vez"

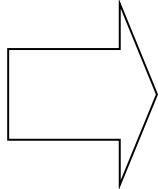
- Elimina situações de concorrência
- Elimina também todo o paralelismo e pode se tornar gargalo de desempenho

## Exemplo de Região Crítica

```
--  
12  double soma = 0;  
13  void *soma_parcial(void *_arg) {  
14      struct soma_parcial_args *spa = _arg;  
15  
16      for (int i = spa->start; i < spa->end; i++) {  
17          soma += spa->vetor[i];  
18      }  
19  
20      return NULL;  
21  }  
--
```

A variável **soma** é **compartilhada globalmente** e incrementada por duas ou mais threads ao mesmo tempo

## Exemplo de Região Crítica

`soma+=spa->vetor[i];`  `mov 0x0(%rip),%rax  
add %rax,%esi  
mov %esi,0x0(%rip)`

O que acontece se as threads **T1** e **T2** forem escalonados na linha acima no momento que estão executando a instrução `soma+=spa->vetor[i];` ?



## Exemplo de Região Crítica

```
mov 0x0(%rip),%rax //thread 1
add %rax,%esi      //thread 1
mov 0x0(%rip),%rax //thread 2
add %rax,%esi      //thread 2
mov %esi,0x0(%rip) //thread 1
mov %esi,0x0(%rip) //thread 2
```

# Implementando uma Região Crítica

- A **região crítica** é implementada através de um **protocolo** de sincronização denominado **Exclusão Mútua**.
- O protocolo de **Exclusão Mútua** deve garantir que se um processo estiver usando um recurso compartilhado os demais serão impedidos de fazer a mesma coisa.
- Implementação do **Protocolo de Exclusão Mútua**
  - Adquire o controle Exclusivo
  - **Região Crítica**
  - Libera o controle Exclusivo

# Semáforo Mutex (Mutual Exclusion)

- Para implementar o **Protocolo de Exclusão Mútua** utilizaremos **semáforos binários**, que são mecanismos de sincronização que permitem gerir o acesso a recursos em **modo exclusivo**;
- **Semáforos Mutex** são denominados **semáforos binários** pois assumem somente dois valores **0** ou **1**.
- Um **semáforo Mutex** é representado por uma variável inteira **não negativa** que só pode ser manipulada pelas primitivas **Lock** e **Unlock**.

# Mutex – Representação Conceitual

**Lock**(Mutex s)

```
{  
    if( s = 1 )  
        s = 0  
    else  
        "Bloqueia a thread"  
}
```

**Unlock**(Mutex s)

```
{  
    if("existe uma thread  
        bloqueada" )  
        "desbloqueia a  
        thread"  
    else  
        s = 1  
}
```

# Mutex – Representação Conceitual

- Para implementar o nosso exemplo de soma global poderíamos criar um semafóro mutex `s` e colocar o trecho de seção crítica entre `Lock()` e `Unlock()`.

```
//Adquire o controle Exclusivo
```

```
Lock(s)
```

```
//Região Crítica
```

```
soma+=spa->vetor[i];
```

```
//Libera o controle Exclusivo
```

```
Unlock(s)
```

# Semáforo Mutex (Mutual Exclusion)

- Caro, mas muito útil quando somos obrigados a compartilhar um recurso
- Ideal é usar **Lock/Unlock** o mínimo possível
- Criar cópias privadas de uma variável compartilhada pode ajudar

# Atividade prática

## Sincronização usando mutex (20 minutos)

1. Utilização da API pthreads para criar mutex
2. Entender quando usá-los e como diminuir seu custo

# Correção

## Sincronização usando mutex

1. Utilização da API pthreads para criar mutex
2. Entender quando usá-los e como diminuir seu custo



# Insper

[www.insper.edu.br](http://www.insper.edu.br)