

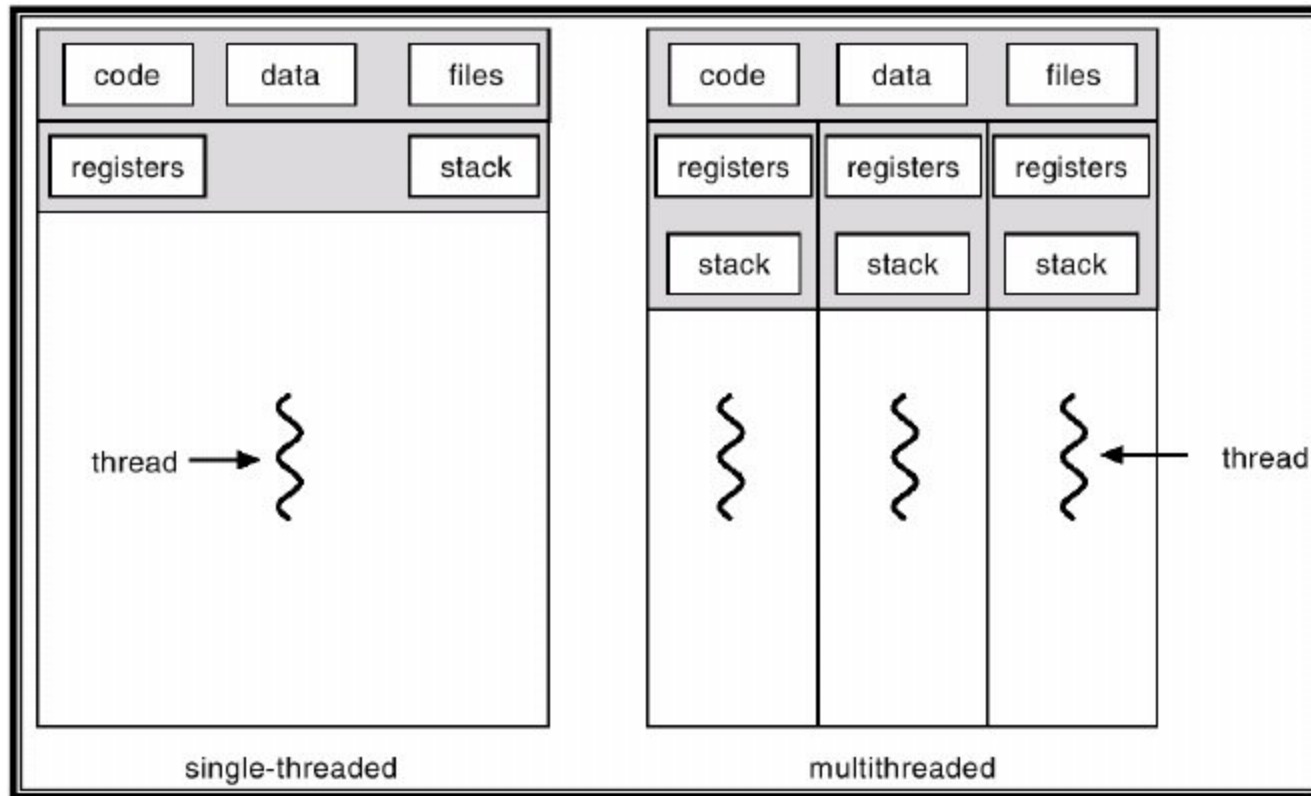
# Sistemas Hardware-Software

Aula 19 - Introdução a sincronização

2021 – Engenharia

Maciel Vidal  
Igor Montagner  
Fábio Ayres

# Processos e threads



# Processos e threads

- Processos
  - Comunicação entre processos
  - Possível distribuir em várias máquinas
- Threads
  - Mais barato de criar e destruir
  - Sempre pertencem a um único processo
  - Sincronização para acessar recursos compartilhados

Troca de contexto ocorre de maneira igual nos dois casos!

# POSIX threads

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

- Criação de threads
- Sincronização (usando semáforos)
- Controle a acesso de dados (usando mutex)

# Problemas limitados por CPU

- Roda tão rápido quanto a CPU puder
- Otimização de cache vale muito
- Faz pouca entrada/saída
  - Interage pouco com o sistema
- Pode ou não ter partes paralelas

# Problemas limitados por CPU

- Dividimos um problemas em partes
- Cada parte é independente (em sua maioria)
- Juntamos os resultados no fim
- Pouca ou nenhuma sincronização



# Tarefas paralelas (CPU-bound)

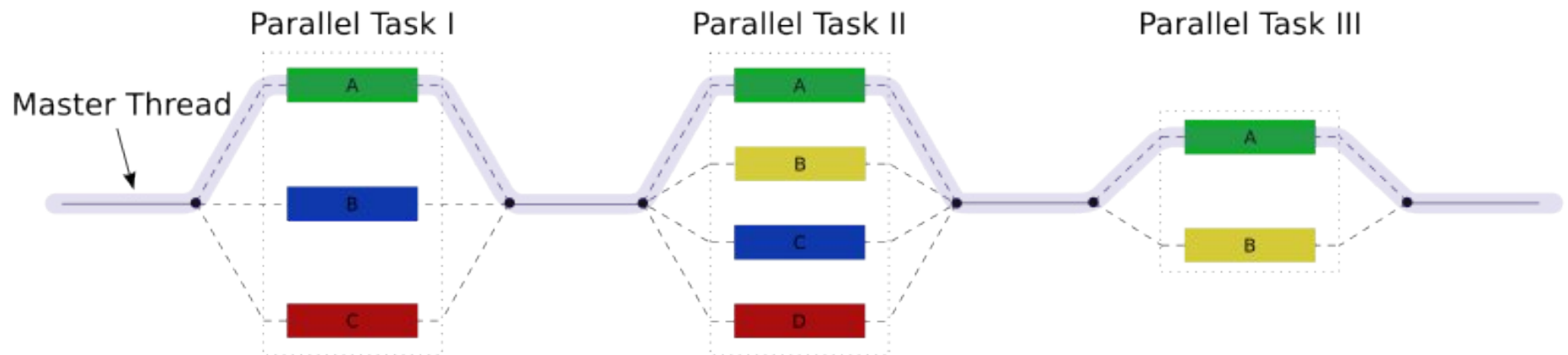
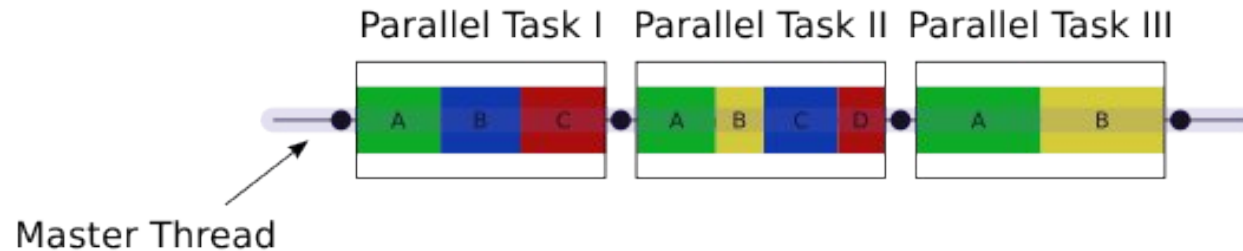


Figura: [https://en.wikipedia.org/wiki/File:Fork\\_join.svg](https://en.wikipedia.org/wiki/File:Fork_join.svg)





# Atividade prática

## **Divisão de trabalho (20 min)**

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

# Correção

## **Divisão de trabalho**

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

# Conceito : Race Condition

"Ocorre quando a saída do programa depende da ordem de execução das threads"

Em geral ocorre quando

- uma variável é usada em mais de uma thread e há pelo menos uma operação de escrita.
- trabalhamos com os mesmos arquivos simultaneamente em várias threads

# Conceito : Região Crítica

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

- elimina situações de concorrência
- elimina também toda a concorrência e pode se tornar gargalo de desempenho

# Mutex (Mutual Exclusion)

*Primitiva de sincronização para criação de regiões de exclusão mútua*

- Lock – se estiver destravado, trava e continua
  - se não espera até alguém destravar
- Unlock – se tiver a trava, destrava
  - se não tiver retorna erro



# Atividade prática

**Usando Mutex para sincronizar threads (20 minutos)**

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

# Correção

## Usando Mutex para sincronizar threads

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

# Atividade prática

## Usando (corretamente) Mutex para sincronizar threads

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



# Mutex

- 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

# Insper

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