

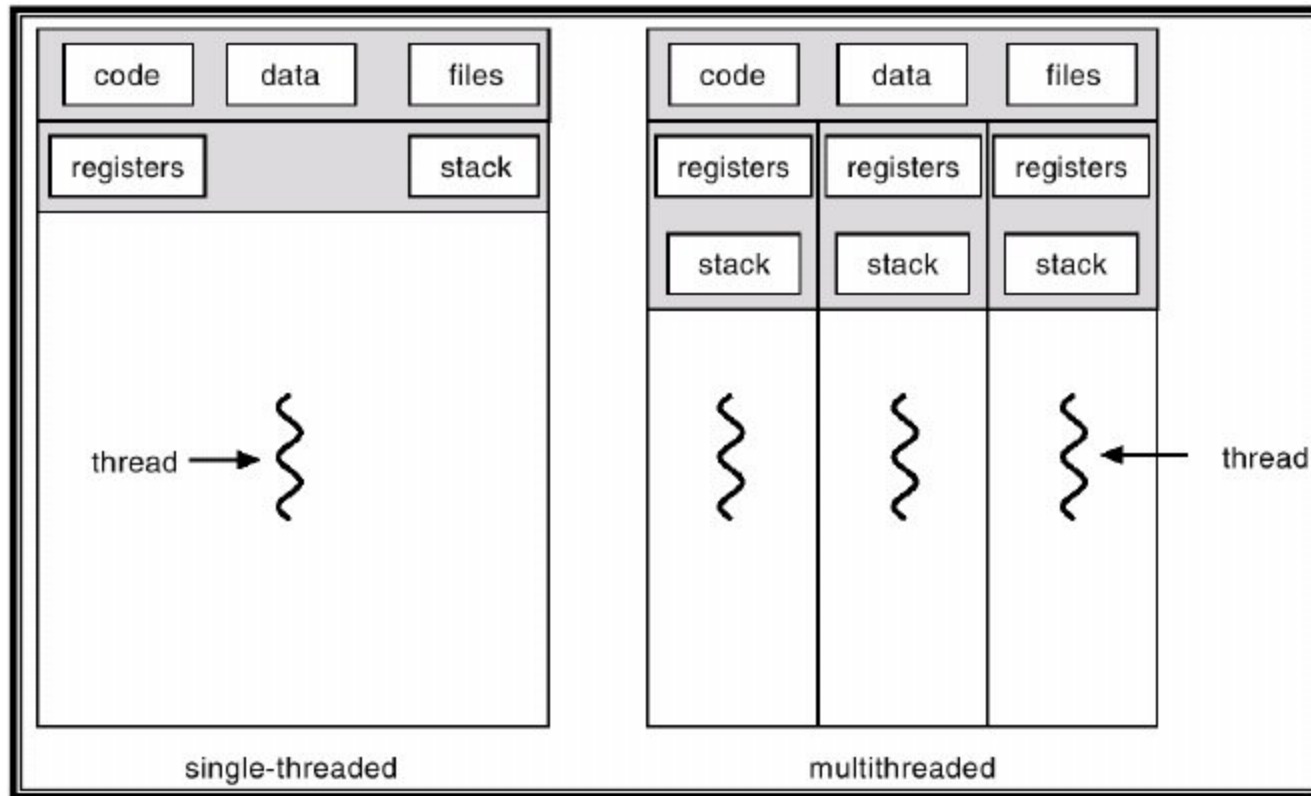
Sistemas Hardware-Software

Aula 20 – Threads III: semáforos

2020 – Engenharia

Igor Montagner , Maciel Calebe, Fábio Ayres

Processos e threads



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"

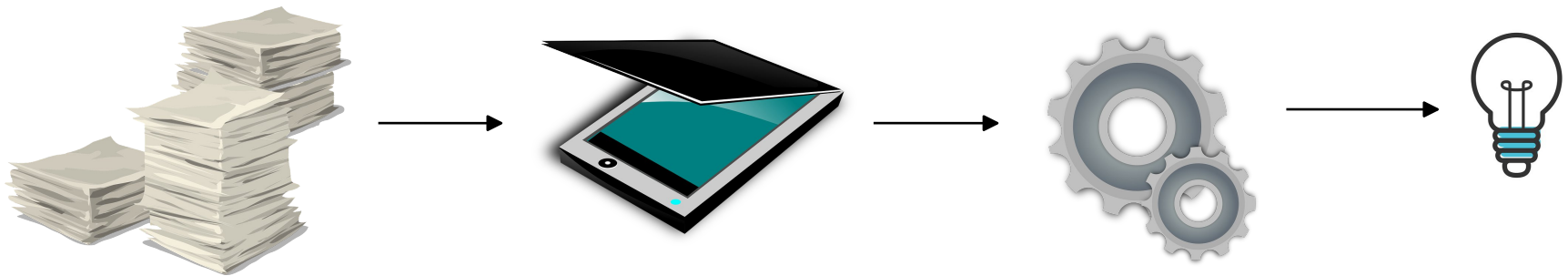
- elimina situações de concorrência
- elimina também toda a concorrência e pode se tornar gargalo de desempenho

Mutex (Mutual Exclusion)

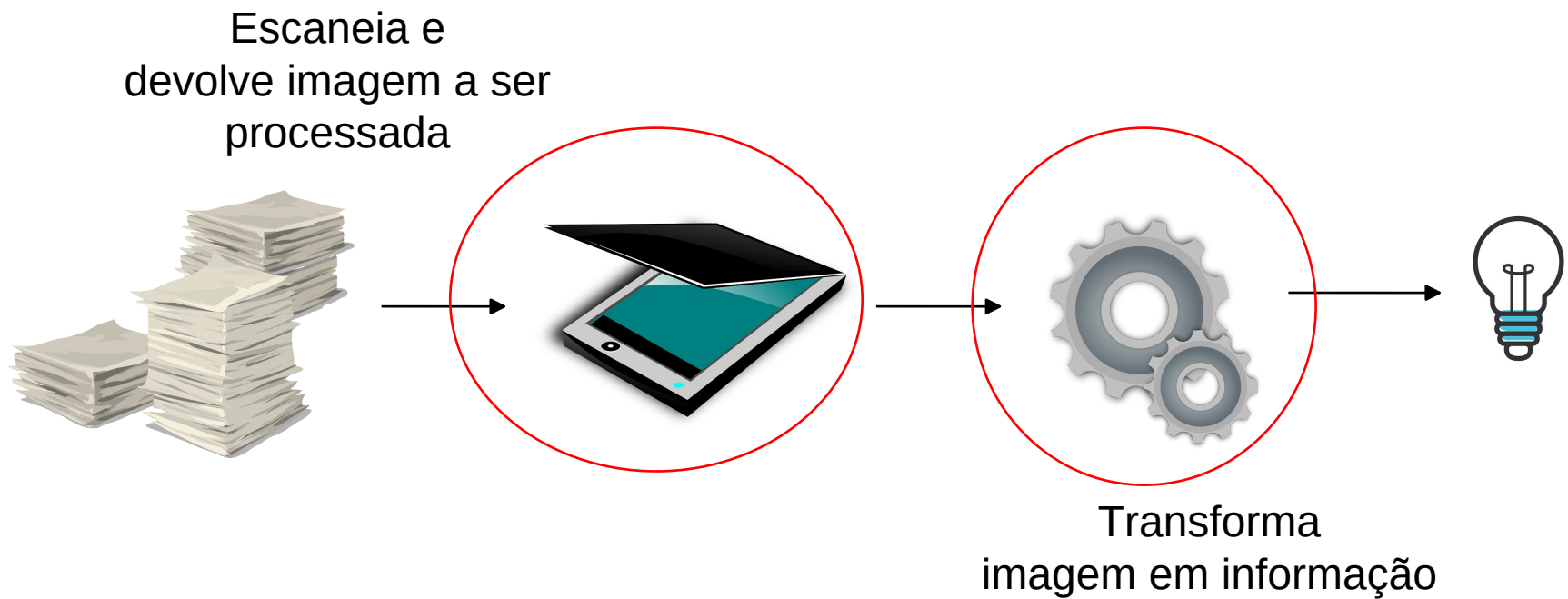
Primitiva de sincronização para criação de regiões de exclusão mútua

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

Problema – leitura de informações



Exemplo – produtor consumidor



Exemplo – produtor consumidor

Dois conjuntos de threads

- Produzem tarefas a serem executadas

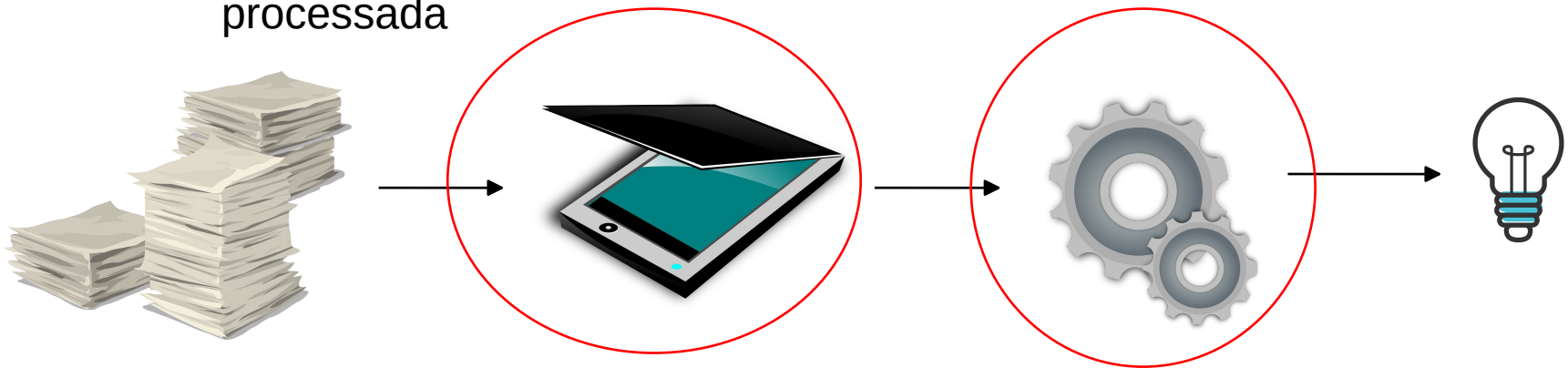
Pode depender de um recurso compartilhado controlar tamanho das tarefas.

- Consomem as tarefas e as executam.

Cada consumidor não depende dos produtores nem de outros consumidores.

Exemplo 1 – produtor consumidor

Produtor: Escaneia e devolve imagem a ser processada



Sincronização

1. Consumidor: espera produtor enviar item
2. Produtor: cria item e avisa Consumidor

Consumidor:
transforma
imagem em informação

Semáforos

"Inteiro especial que nunca fica negativo"

Só pode ser manipulado por duas operações atômicas

POST:

- Aumenta o valor em 1

WAIT:

- Se for positivo, diminui em 1
- Se for 0 fica esperando;



Atividade prática

Usando semáforos para sincronizar tarefas

1. Rendez-vous
2. API POSIX para semáforos

Insper

www.insper.edu.br