

# Processos

Gui / Eli / Adi / Filipe / Marcos

# O que é um processo?



Podemos definir processos como softwares que **executam alguma ação** e que **podem ser controlados** de alguma maneira, seja pelo usuário, pelo aplicativo correspondente ou pelo sistema operacional.

Os processos podem ser gerenciados ou não, dependendo do sistema operacional do equipamento.

Uma forma de visualizar os processos em execução, por exemplo, no Windows, é através do Gerenciador de Tarefas (Ctrl+Shift+Esc).

# Comunicação entre processos - IPC

O processo pode ser **independente** ou **cooperativo**. A cooperação requer que os processos comuniquem entre si e sincronizem suas ações.

Principais motivos para cooperação entre processos:

- Velocidade de computação
- Modularidade
- Compartilhamento de informações

Os processos podem se comunicar com outros através de dois modelos:

- Memória compartilhada
- Troca de mensagem

# Sincronização e operações de processos

## Estados

- |              |              |
|--------------|--------------|
| 1. Novo      | 4. Bloqueado |
| 2. Pronto    | 5. Encerrado |
| 3. Execução. |              |

## Semáforo

- Espera
- Bloqueado
- Avanço

## Processos

- Independentes
- Cooperativos

## Técnicas de planejamento

Principais:

- FIFO - First-In First-Out
- SJF - Shortest Job First
- SRTF - Shortest Remaining Time
- Round Robin



# Threads de execução

As **threads** são fluxos de um programa em execução, que podem ou não ser divididos para trabalhar em mais de um processador.

**Ex. 1:** Um programa que recebe dois números, multiplica um pelo outro e retorna, contém apenas um fluxo de execução.

**Ex. 2:** Um programa que transforma uma imagem em cinza, pode dividir a imagem em 4 quadrantes e processar cada quadrante em um thread (fluxo de execução) para depois retornar a imagem final.

# Problemas nas threads

**Espera ocupada:** Enquanto uma thread está executando a seção crítica, outras threads que querem acessar o mesmo recurso devem aguardar.

**Condição de corrida:** Quando mais de uma thread tenta acessar uma variável ao mesmo tempo temos uma condição de corrida. Ou seja, elas concorrem por algum recurso. O impacto dessa condição, é que o resultado da computação dessas threads depende de quem executou primeiro. Não há como garantir que independente do número de execuções o resultado será o mesmo.

# Métodos e módulos de sincronização de threads

**Exclusão mútua:** A exclusão mútua é uma propriedade que garante que exclusivamente apenas uma unidade de execução (thread) esteja executando.

**Mutex:** O mutex é uma implementação de exclusão mútua. Entretanto, ele gera espera ocupada.

**Semáforo:** O semáforo é outra implementação de exclusão mútua. Diferentemente do mutex, eles não tem espera ocupada.

**Barreira:** Elas estabelecem um ponto na execução de um programa onde cada thread é bloqueada até que todas as demais threads alcancem a barreira. Somente quando todas chegaram até a barreira é que elas podem continuar.

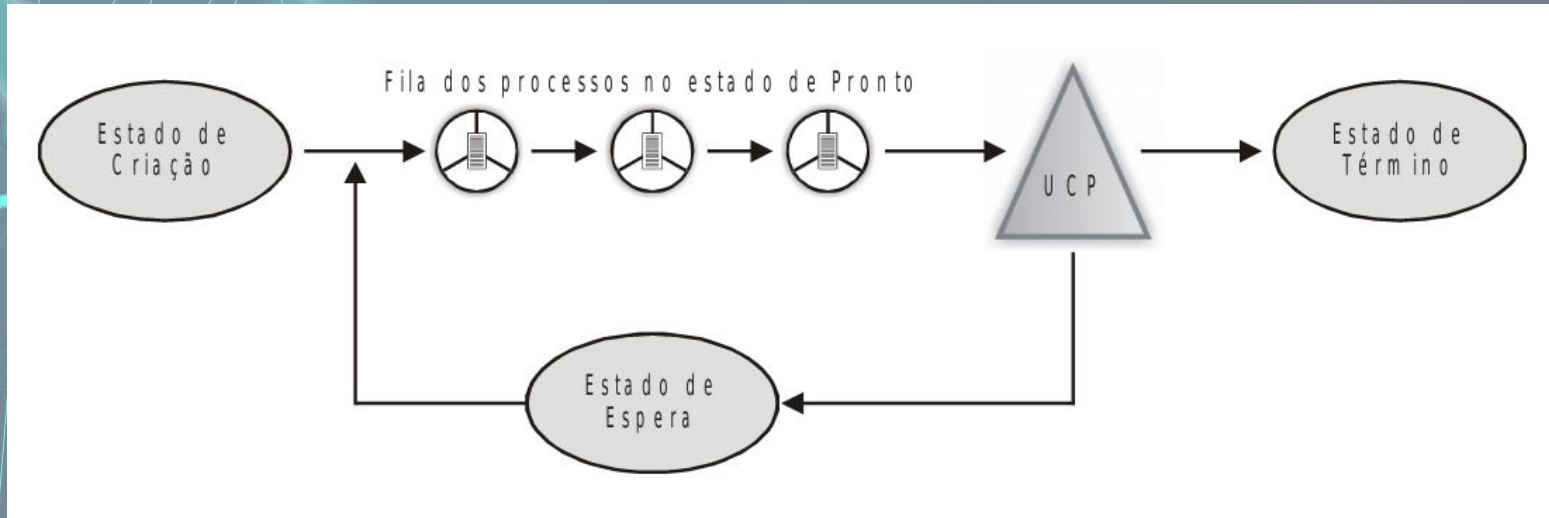
# Algoritmos de escalonamento

Um **Escalonador de Processos** é um subsistema do Sistema Operacional responsável por decidir o momento em que cada processo será executado pela CPU. É utilizado algoritmos de escalonamento que estabelecem como será tomada essa decisão.



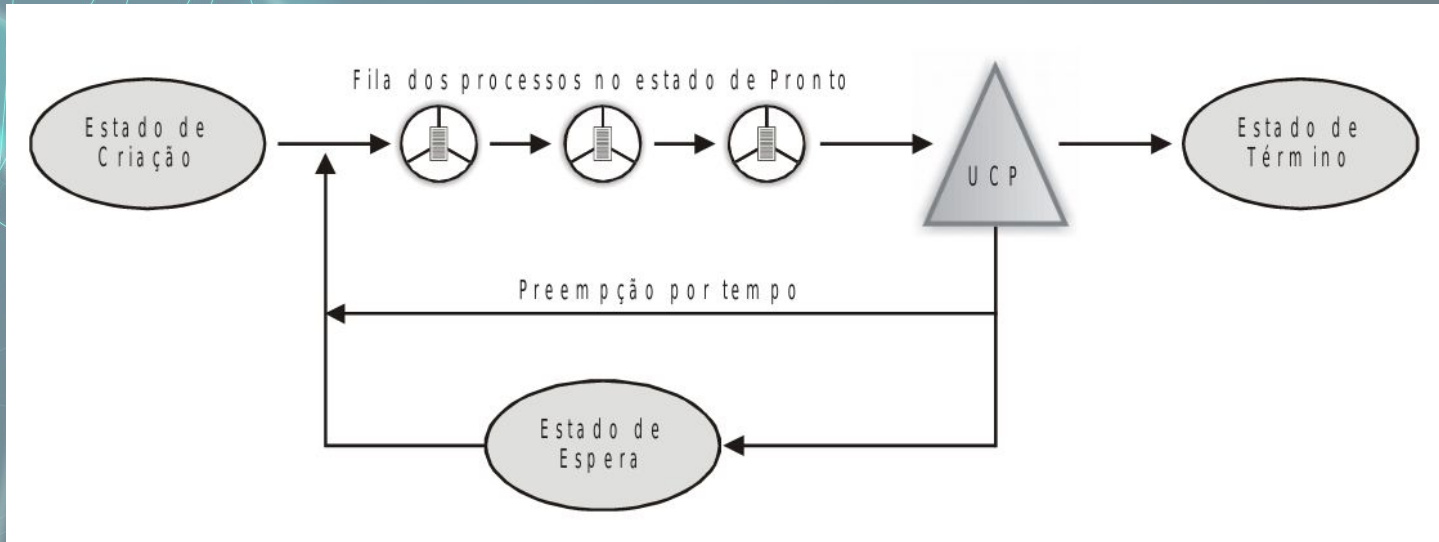
# Algoritmos de escalonamento

- Primeiro a chegar, primeiro a sair (FIFO)



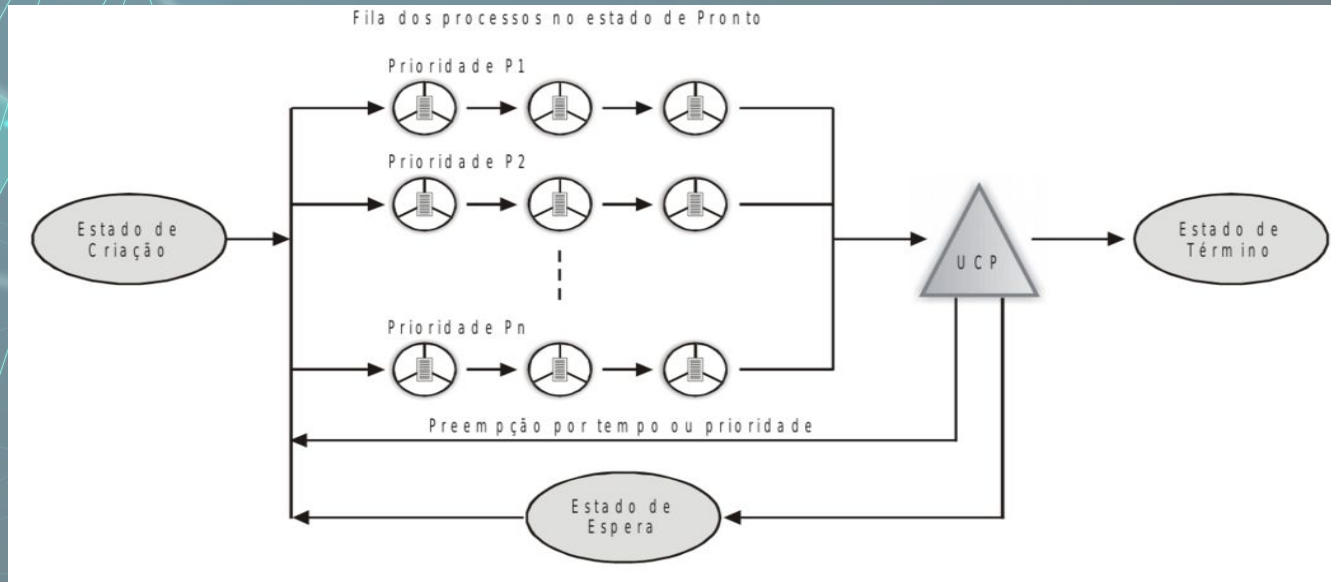
# Algoritmos de escalonamento

- Round-Robin (RR)



# Algoritmos de escalonamento

- Múltiplas filas

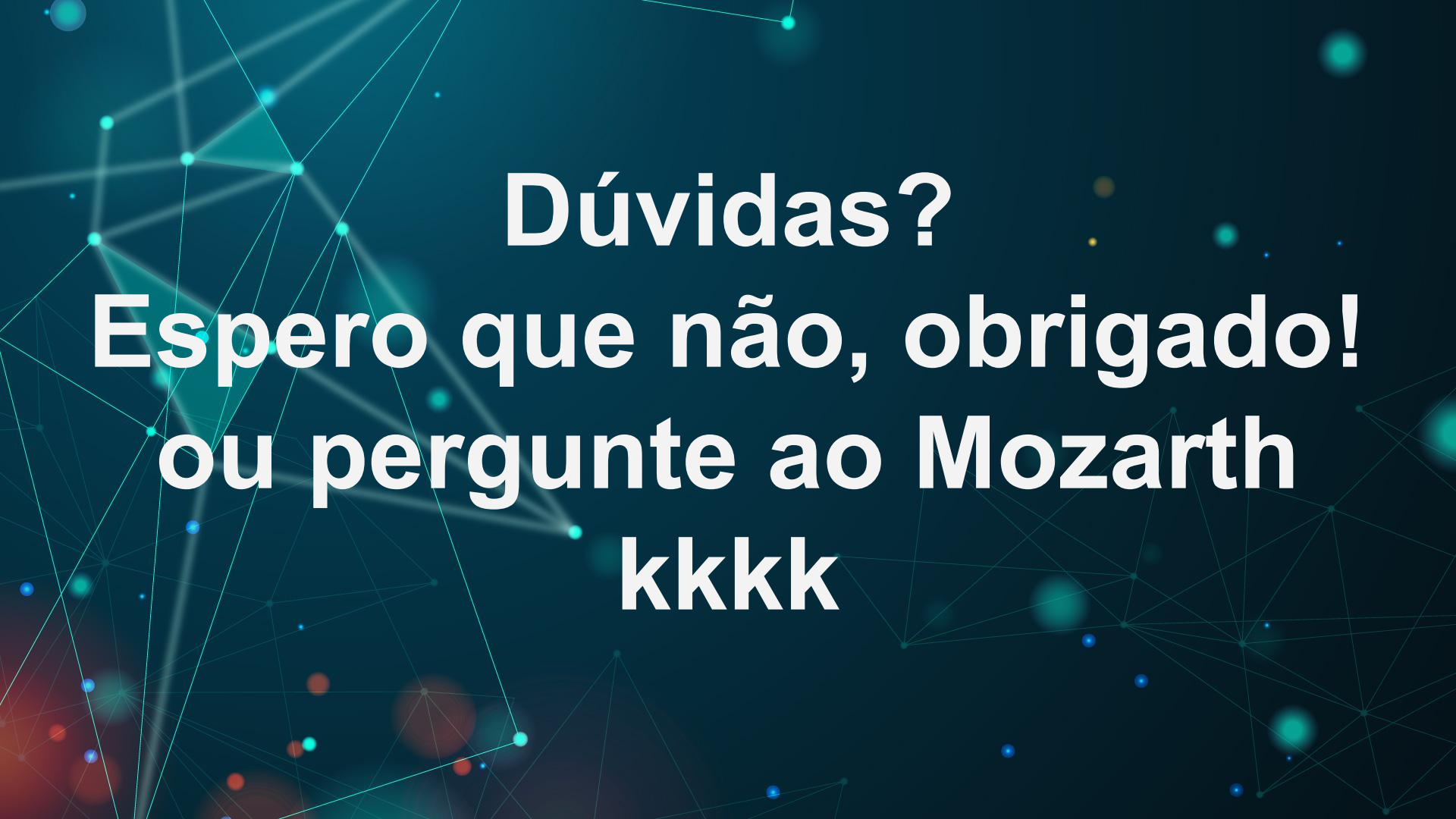


# Algoritmos de escalonamento

Outro exemplos:

- O próximo processo mais curto (SJF) - Não preemptivo
- Próximo processo, o menor tempo restante (SRTF) - Preemptivo





**Dúvidas?**  
**Espero que não, obrigado!**  
**ou pergunte ao Mozarth**  
**kkkk**