

---

## Resumo Organizado: Sistemas Operacionais (SO)

### 1. O que é um Sistema Operacional?

- É o **software básico** instalado na máquina física (hardware).
- Pode ser definido como o software colocado **entre o hardware e os programas do usuário** (aplicativos).
- Todos os programas que instalamos rodam "em cima" do Sistema Operacional.

### 2. As Duas Funções Principais do SO

O SO tem dois papéis fundamentais que definem sua operação:

#### A. Abstração (O Intermediário)

- O SO funciona como um **intermediário entre o hardware e o software**.
- **Para Desenvolvedores de Software (Devs):** Eles não precisam se preocupar com os detalhes específicos do hardware (como qual placa de vídeo ou tipo de disco) onde seu software será executado.
- **Para Desenvolvedores de Hardware:** Eles precisam fornecer os **drivers** (que funcionam como um "manual de instruções"), permitindo que o SO saiba como controlar aquele componente específico.

#### B. Gerenciamento de Recursos (O Gerente)

- O SO é responsável por **gerenciar o uso do hardware** (recursos) pelos softwares (aplicativos).
- Ele gerencia todos os subsistemas do dispositivo, como:
  - Processadores (CPU)
  - Memória (RAM)
  - Discos (Armazenamento)
  - Gerenciamento de Entrada e Saída (E/S - teclado, mouse, etc.)
- Uma tarefa crucial é a **distribuição otimizada da CPU** entre as várias tarefas que estão em execução ao mesmo tempo.

### 3. Como o SO Interage com os Programas

- Qualquer requisição que você faz na interface (como um clique do mouse ou digitar algo) passa primeiro pelo SO.

- O SO, então, traduz essa requisição e a implementa no hardware para que a ação aconteça.

#### 4. O SO como "Máquina Virtual"

- O SO pode ser visto como uma "**máquina virtual**".
  - Todo dispositivo de hardware possui uma linguagem de máquina que é muito primitiva e difícil para humanos entenderem.
  - O SO é o programa responsável por **esconder esses detalhes** complexos de implementação, gerando uma abstração de mais alto nível, tornando o computador mais fácil de usar.
- 

#### 5. Tipos de Programas Computacionais

Os programas podem ser divididos em duas categorias principais:

1. **Programas de Sistema:** São aqueles que manipulam a operação do computador.
  - O **Sistema Operacional** é o **programa de sistema mais importante**.
2. **Programas Aplicativos:** São aqueles que resolvem problemas específicos para os usuários (ex: navegador, editor de texto, jogos).



### Funcionamento de Sistemas Operacionais

#### 1. Processos (ou Tarefas)

- **Conceito:** É a ideia fundamental do sistema operacional. Basicamente, é um **programa em execução**.
  - **Função:** Está sempre solicitando recursos da máquina (como CPU, memória) e interagindo com outros processos.
- 

#### 2. Estados de um Processo

- **Executando:** Está ativamente usando a CPU para executar as instruções do programa.
- **Bloqueado / Em Espera:** Está parado, esperando pela ocorrência de algum evento externo (Ex: término de uma leitura do disco, espera por uma entrada do usuário).

- **Ativo / Pronto:** Está na fila, pronto para rodar, apenas aguardando a CPU ficar livre para executar.
- 

### 3. Gerenciamento e Interface

- **System Calls (Chamadas de Sistema):** É a forma como o Sistema Operacional gerencia os processos (como eles pedem recursos, etc.).
  - **Shell:** É o interpretador de comandos (CLI - *Command Line Interface*). Você interage com o SO digitando comandos.
  - **GUI (Interface Gráfica):** Permite ao usuário interagir visualmente (com mouse, janelas, ícones), sem precisar digitar comandos de texto.
- 

### 4. Sistemas Multusuário e Multitarefas

- **Multusuário:**
  - **O que é:** Permite que várias contas de usuário diferentes usem o mesmo computador.
  - **Como funciona:** Cada usuário pode usar o computador com seus próprios arquivos e configurações, sem bagunçar os dos outros.
  - **Técnica principal:** Utiliza a técnica de **multiprocessamento**.
- **Multitarefas:**
  - **O que é:** É a capacidade do Sistema Operacional de alternar rapidamente entre vários processos (ou tarefas) que estão em execução.
  - **Técnica principal:** Atua com a técnica de **multiprogramação**.



## Resumo: Arquitetura de Sistemas Operacionais

### 1. O Conceito de Máquina de Níveis (Abstração)

Um computador é extremamente complexo. Para facilitar o seu uso e desenvolvimento, ele é organizado em "níveis" ou "camadas" de abstração. Cada nível usa os serviços do nível inferior e oferece serviços ao nível superior, escondendo a complexidade.

Suas anotações listam estes níveis, que geralmente são vistos de baixo para cima:

- **Nível 6: Aplicação:** Os programas que o usuário final utiliza (Ex: seu navegador, editor de texto, jogos).
- **Nível 5: Aplicativo (Bibliotecas):** Muitas vezes se refere às bibliotecas de programação (APIs) que os programas de aplicação usam.
- **Nível 4: Sistema Operacional:** O grande gerente. Ele fornece uma interface mais simples para os programas acessarem o hardware.
- **Nível 3: Programação Assembler:** Uma linguagem de baixo nível que representa diretamente as instruções da máquina.
- **Nível 2: Micropogramação:** Um nível muito baixo que traduz as instruções (como "SOMAR") em sinais elétricos reais para a CPU.
- **Nível 1: Lógica Digital (Circuito Integrado):** O hardware físico, composto por portas lógicas (AND, OR, NOT) e circuitos que executam os cálculos.



## 2. O Coração do Sistema: O Kernel

O **Kernel** é o **núcleo** do Sistema Operacional. É a parte mais importante, que fica sempre em execução e controla tudo.

### Funções do Kernel

Conforme suas anotações, as funções principais do Kernel são:

- **Escalonamento e comunicação entre processos:** Decidir qual processo usa a CPU em cada momento.
- **Tratamento de interrupções (Chamadas de Sistema):** Lidar com pedidos dos programas (Ex: "abrir um arquivo") ou eventos de hardware (Ex: "tecla pressionada").
- **Gerenciamento do processador:** Alocar tempo de CPU para os processos.
- **Gerenciamento de memória:** Controlar quem usa qual parte da memória RAM.
- **Gerenciamento de sistemas de arquivos:** Organizar, ler e escrever dados no HD/SSD.
- **Gerenciamento de dispositivos de E/S (Entrada/Saída):** Controlar o acesso a hardware como mouse, teclado, impressora, etc.

## ET 4G

---

### 3. Modos de Acesso (Proteção)

Para proteger o Kernel (e o sistema todo) de falhas ou programas maliciosos, os computadores modernos usam dois modos de acesso:

- **Modo Usuário (Não-Privilegiado):**
  - É onde os aplicativos normais rodam (navegador, jogos, etc.).
  - Tem um acesso "de superfície", ou seja, **não pode** acessar o hardware diretamente nem executar "instruções privilegiadas".
  - Se um programa em Modo Usuário falhar, ele "quebra" sozinho, mas não derruba o sistema inteiro.
- **Modo Kernel (Privilegiado):**
  - É onde rodam as partes essenciais do sistema, ou seja, o próprio **Kernel**.
  - Tem acesso total e irrestrito a todo o hardware e a todas as instruções da CPU.
  - Uma falha no Modo Kernel é catastrófica e geralmente causa a "Tela Azul" (ou *Kernel Panic*).

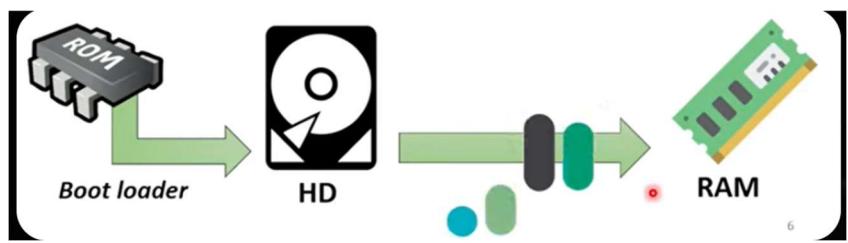
**Ponto-chave:** Quando um aplicativo (Modo Usuário) precisa de algo do hardware (Ex: ler um arquivo), ele faz uma "**Chamada de Sistema**" (**System Call**). Isso é um pedido formal para o Kernel (Modo Kernel) fazer o serviço para ele.

---

#### 4. Ativação do Sistema (Boot)

Quando você liga o computador, o SO não está rodando. Ele precisa ser "acordado" e carregado. Este é o processo de **Boot**:

1. O SO (Windows, Linux, etc.) está guardado "dormindo" na **memória secundária** (HD ou SSD).
2. Ao ligar, o computador primeiro lê um pequeno programa de uma memória permanente especial chamada **ROM** (ou firmware UEFI/BIOS).
3. Esse programa da ROM é o **Boot Loader**. Sua única tarefa é encontrar o SO no HD/SSD.
4. O Boot Loader então **copia** as partes essenciais do SO do HD/SSD para a **memória principal (RAM)**, que é muito mais rápida.
5. Uma vez na RAM, o Kernel do SO finalmente começa a ser executado e toma o controle do computador.



## 5. Modelos de Arquitetura de SO

Existem diferentes formas de organizar as múltiplas partes de um Sistema Operacional.

### Arquitetura Monolítica

- **O que é:** O Sistema Operacional inteiro (gerenciamento de arquivos, memória, E/S, etc.) é um único e grande programa que roda em **Modo Kernel**.
- **Vantagem:** Muito rápido, pois a comunicação entre as partes é direta (são todas partes do mesmo "bloco").
- **Desvantagem:** Uma falha em qualquer parte (como um driver de vídeo) pode derrubar todo o sistema. É difícil de atualizar e depurar.

### Arquitetura Micronúcleo (Microkernel)

- **O que é:** O Kernel (Modo Kernel) é mantido o menor possível, fazendo apenas o mínimo (comunicação e escalonamento).
- Outras partes, como drivers e sistemas de arquivos, rodam como processos comuns em **Modo Usuário** (como suas notas corretamente apontam).
- **Vantagem:** Mais seguro e estável. Se um driver falhar (em Modo Usuário), ele pode ser reiniciado sem parar o sistema.
- **Desvantagem:** Mais lento, pois a comunicação entre o Kernel e os drivers (agora em espaços diferentes) exige mais "burocracia".

### Arquitetura de Camadas

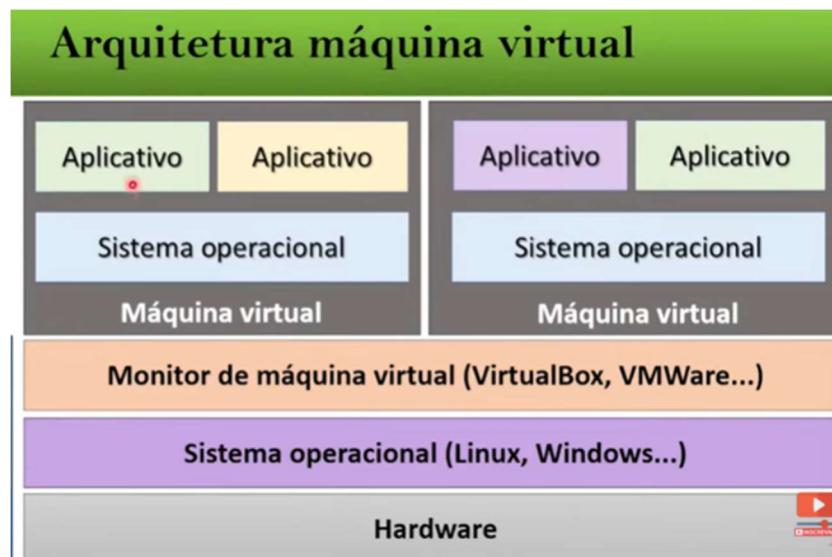
- **O que é:** O SO é organizado em níveis de camadas hierárquicas. A camada 0 é o hardware, a camada 1 gerencia a memória, a camada 2 gerencia os

processos, etc. Cada camada só pode usar os serviços da camada imediatamente abaixo.

- **Vantagem:** (Como anotado) **Isolamento de camadas**. Facilita muito o desenvolvimento e a depuração, pois você pode focar em uma camada de cada vez.
- **Desvantagem:** (Como anotado) **Desempenho**. Um pedido de um aplicativo pode ter que passar por várias camadas até chegar ao hardware, o que gera lentidão.

### Arquitetura de Máquina Virtual (VM)

- **O que é:** Um programa (chamado **Monitor de Máquina Virtual** ou **Hypervisor**, como o VirtualBox ou VMWare) cria uma **cópia virtual do hardware**.
- Isso permite que você instale e execute um Sistema Operacional (ou vários) como se fosse um aplicativo dentro do seu SO principal.
- Suas notas também mencionam **Emulador**, que é um conceito próximo: um emulador possibilita executar um aplicativo de *outra plataforma* (Ex: rodar um jogo de celular Android no seu PC Windows).



## 💻 Gerenciamento de Processos

### 1. O Contexto: Multiprogramação e Concorrência

Com o surgimento de sistemas **multiprogramáveis**, múltiplos processos podem permanecer na memória principal (RAM), **compartilhando o uso da CPU**.

Isso gera a necessidade de gerenciar o **acesso concorrente**, que é um dos problemas mais difíceis da administração de recursos, pois muitos processos podem solicitar o uso da CPU (e outros recursos) simultaneamente.

---

## 2. Desafios do Acesso Concorrente

O SO precisa arbitrar o acesso para evitar caos. Os principais desafios são:

- **Proteção de Dados:** Processos concorrentes frequentemente precisam acessar o mesmo arquivo ou dado. O SO deve garantir que um processo não possa alterar um dado que outro processo esteja usando ao mesmo tempo.
- **Sincronização:** Ocorre quando um processo gera dados que outro processo usará. O SO deve garantir que o segundo processo não tente usar os dados *antes* que o primeiro tenha terminado de gerá-los.

Para regular essa ordem e resolver esses conflitos, o SO utiliza o **Escalonamento**.

---

## 3. O que Compõe um Processo?

Um processo possui duas partes principais:

- **Parte Passiva:** O programa em si, com seus dados e recursos.
  - **Parte Ativa (Processo Leve ou Thread):** É a unidade de execução do processo, o que "roda" efetivamente na CPU.
- 

## 4. Processos e a Memória (Swapping)

Nem sempre um processo pronto para executar está na memória principal:

- **Swapping:** É a técnica de mover processos da memória principal (RAM) para a **memória secundária (HD/SSD)**.
  - **Quando ocorre:** Acontece quando não existe espaço suficiente na RAM para todos os processos que estão em estado de "Pronto" ou "Espera". O SO "troca" processos de lugar para dar espaço.
- 

## 5. Classificação dos Processos

Os processos podem ser classificados por prioridade ou por comportamento:

- **Por Prioridade:**

- **Processo de Sistema:** Executa com prioridade mais alta para garantir o funcionamento do SO.
- **Processo de Usuário:** Programas comuns do usuário (navegador, jogos, etc.).

- **Por Comportamento:**

- **Processo CPU-Bound:** Passa a maior parte do tempo utilizando a CPU. (Ex: renderizando um vídeo, fazendo um cálculo matemático complexo).
  - **Processo I/O-Bound:** Passa a maior parte do tempo esperando por operações de Entrada/Saída (E/S). (Ex: esperando o disco ler um arquivo, esperando dados da rede, esperando o usuário digitar algo).
- 

## Escalonamento de Processos

Escalonamento é a atividade de **gerenciar o processador da máquina**, decidindo qual processo será executado a seguir.

### 6. Política de Escalonamento (Os Objetivos)

É o conjunto de **critérios** usados para decidir qual processo, entre os vários na fila de "Prontos", será escolhido. Os objetivos principais são:

- Manter o processador ocupado a maior parte do tempo.
- Equilibrar o uso da CPU entre os diferentes processos.
- Privilegiar a execução de aplicações críticas.
- Maximizar o desempenho (vazão) do sistema.
- Oferecer um tempo de resposta razoável para usuários interativos.

### 7. O Escalonador (Scheduler)

- O **Escalonador** é o componente do SO que **implementa** a política de escalonamento.
- É ele quem, de fato, utiliza os critérios para escolher o próximo processo a rodar.

- Cada SO (Windows, Linux, etc.) possui suas próprias políticas e escalonadores.

## 8. Tipos de Escalonamento (As Técnicas)

Existem duas formas principais de escalar:

### 1. Não-Preemptivo (ou Cooperativo)

O Sistema Operacional **não interrompe** um processo que está em execução. O processo só para de usar a CPU quando ele mesmo decide (porque terminou ou porque precisa esperar por E/S).

- **Exemplos de Algoritmos:**

- **FIFO (First-In, First-Out):** O primeiro a chegar na fila é o primeiro a ser executado (igual a uma fila de banco).
- **SJF (Shortest Job First):** Dá prioridade ao processo que for "menor" (que exige menos tempo de CPU para terminar).
- **HRN (Highest Response-ratio Next):** Um critério que considera há quanto tempo o processo está esperando na fila, evitando que processos longos fiquem "presos" para sempre.

### 2. Preemptivo

O Sistema Operacional **pode interromper** um processo em execução a qualquer momento para colocar outro em seu lugar (seja porque o tempo do processo acabou ou porque um processo mais importante chegou).

- **Exemplos de Algoritmos:**

- **Round Robin (Circular):** Cada processo recebe uma pequena "fatia de tempo" (*quantum*) para rodar. Se não terminar, é interrompido, vai para o fim da fila e espera sua vez de novo.
- **Prioridades:** O processo com a maior prioridade é executado. Se um processo de prioridade maior chegar, ele "toma" a CPU de um processo de prioridade menor.
- **Múltiplas Filas:** O sistema gerencia várias filas (Ex: uma fila de alta prioridade para sistema, uma de média para interativos e uma de baixa para processos em lote), e cada fila pode ter seu próprio algoritmo (Ex: Round Robin na fila interativa).

## Tipos de Escalonamento

Existem duas categorias principais de escalonamento, que definem se o Sistema Operacional pode ou não interromper um processo.

- **Preemptivo:** O SO **pode interromper** um processo em execução, se for necessário (ex: seu tempo acabou ou um processo mais importante chegou).
- **Não-Preemptivo:** O SO **não interrompe** um processo. O processo só para de usar a CPU quando ele mesmo decide (porque terminou ou porque precisa esperar por algo, como E/S).

---

### 1. Escalonamento Não-Preemptivo

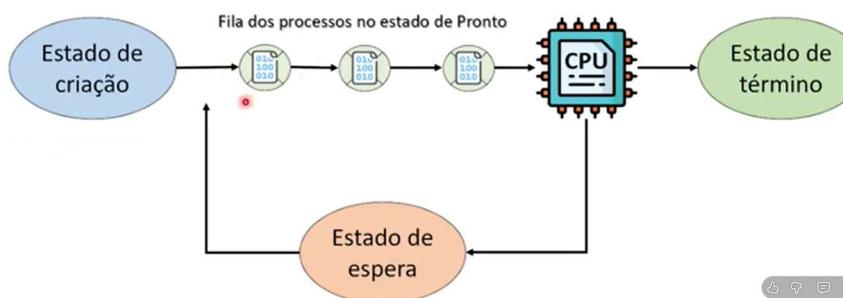
O processo roda até que ele mesmo se libere.

- **FIFO (First-In, First-Out):**

- O primeiro processo que entra na fila é o primeiro a ser executado.
- É o mais simples, como uma fila de banco.

---

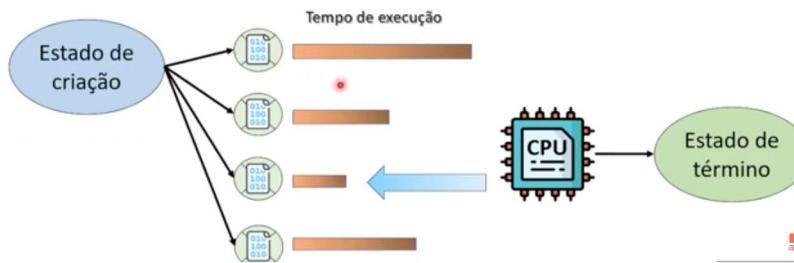
#### 1) Escalonamento Primeiro a entrar, primeiro a sair (FIFO)



- **SJF (Shortest Job First):**

- O "menor" processo (aquele que precisa de menos tempo de CPU) é executado primeiro.
- Busca otimizar o tempo médio.

## 2) Escalonamento Menor Job Primeiro (SJF)

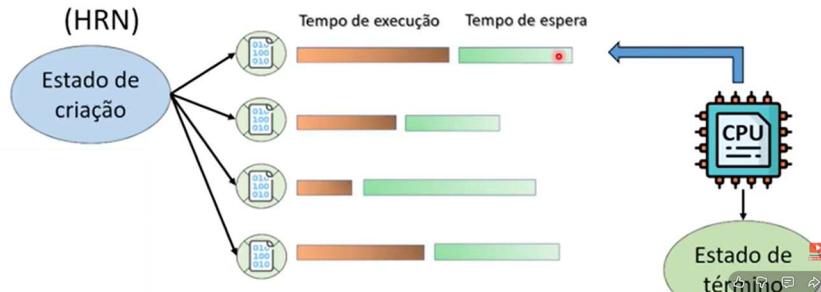


### • HRN (Highest Response-ratio Next):

- Escalonamento pela próxima taxa de resposta mais alta.
- É um algoritmo que considera o tempo que o processo já passou esperando na fila, evitando que processos longos sejam "esquecidos".

---

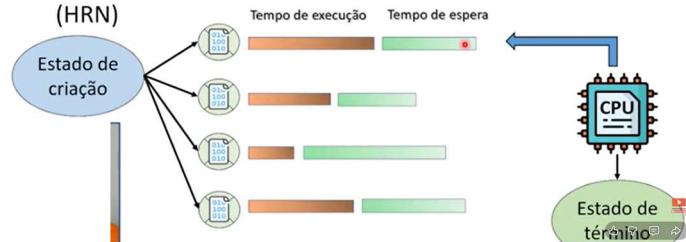
## 3) Escalonamento pela Próxima Taxa de Resposta Mais Alta (HRN)



### • Escalonamento Cooperativo:

- Os processos "cooperam" e avisam ao SO quando podem ser parados. É o processo que decide quando "devolver" a CPU.

### 3) Escalonamento pela Próxima Taxa de Resposta Mais Alta (HRN)



## 2. Escalonamento Preemptivo

### Preemptivo

- 1) Escalonamento Round Robin (ou Circular)
- 2) Escalonamento por Prioridades
- 3) Escalonamento por Múltiplas Filas

O SO tem o poder de interromper processos.

- **Round Robin (Circular):**

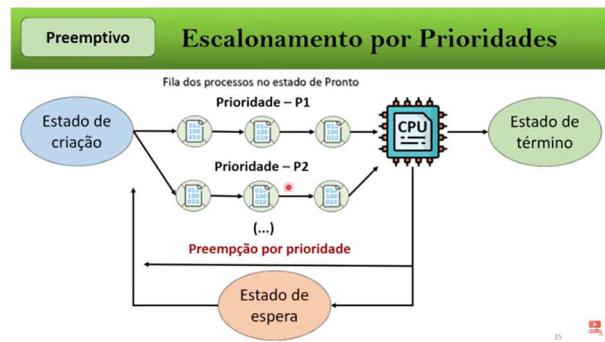
- Parecido com o FIFO, mas todos os processos são colocados em um círculo.
- Cada processo recebe uma pequena "fatia de tempo" (*quantum*).
- Se o processo não terminar nesse tempo, ele é **interrompido**, vai para o final da fila e espera sua próxima vez.

### 1) Escalonamento Round Robin (ou Circular)



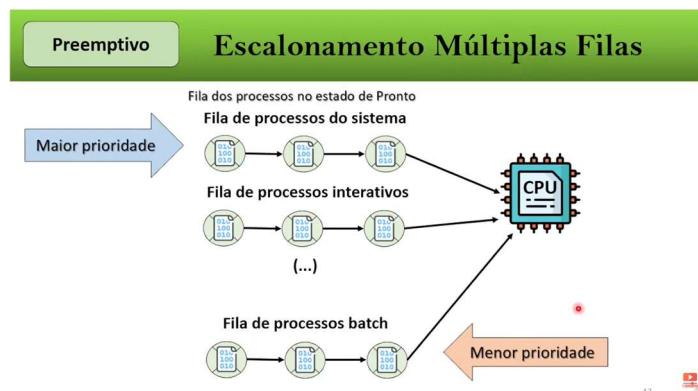
- **Escalonamento por Prioridade:**

- Cada processo recebe um nível de prioridade (alta, média, baixa, etc.).
- O SO sempre executa o processo de maior prioridade que estiver pronto.
- Se um processo de prioridade maior chegar, ele **interrompe** (preempta) o processo de prioridade menor que estava rodando.



- **Escalonamento de Múltiplas Filas:**

- O sistema cria várias filas separadas, geralmente baseadas em prioridade ou tipo de processo (ex: fila de "sistema", fila "interativa", fila "em lote").
- Cada fila pode usar seu próprio algoritmo (ex: a fila de alta prioridade pode ser *Round Robin*, e a de baixa pode ser *FIFO*).



## Gerenciamento de Memória

### 1. Memória Principal (RAM)

- É onde residem (ficam) os **programas em execução**.
- É uma memória volátil (perde os dados quando o computador é desligado).

### 2. Memória Secundária (HD ou SSD)

- É uma **memória permanente** (não volátil).
- Guarda os dados e programas quando eles **não estão mais em execução**.

### 3. GM (Gerenciamento de Memória)

- É a função (do Sistema Operacional) que **monitora** quais partes da memória estão disponíveis ou em uso.
- É responsável por **alocar** (reservar) e **liberar** (desocupar) espaço na memória para os processos.

### 4. MMU (Unidade de Gerenciamento de Memória)

- É um componente de hardware (geralmente parte da CPU).
- Atua como um "**tradutor**".
- Converte os endereços **lógicos** (usados pelo programa) para os endereços **físicos** (a localização real na memória RAM).