O que é um processo?

Resumidamente, podemos dizer que um processo é um programa em **execução** e que um **programa executável** é um conjunto de instruções e dados armazenados em um **arquivo**. Portanto, quando o conteúdo desse programa é carregado na memória e executado, ele se torna um processo.

Estagio de um processo

1 – Preparado: Quando faz o pedido em um fast food

2 – Em espera: Quando está esperando o lanche ser finalizado

3 – Em execução: Quuando o lanche fica pronto e você vai pegar

4 – Finalizado: Quando se termina de comer o lanche

Existem 2 tipos de processos:

* Processo Independente: é o processo que não precisa de outra para ser executado.
* Processo Cooperativo: é o processo que necessita que outro **compartilhe informações** com ele, para que ele possa ser executado.

Mecanismos básicos de comunicação:

Memória compartilhada

Baseia-se nos processos que desejam se comunicar compartilhando a mesma região de memória física. Para realizar a comunicação, um escreve e o outro lê da região de memória compartilhada. Os processos usam serviços do sistema operacional para compartilhar a região.

Passagem de mensagens

Os processos usam um par de serviços do sistema operacional para se comunicar. Esses serviços são comumente conhecidos como enviar e receber. Para realizar a comunicação, um processo executa a função enviar e o outro recebe, trocando assim um bloco de informações que recebe o nome da mensagem.

O que é uma thread?

As threads representam uma sequência simples de **instruções** executadas em **paralelo** com outras **sequências**, são uma forma de dividir um programa em várias tarefas que são executadas **simultaneamente**. Um processo pode conter um ou mais threads.

Todos os threads de um processo compartilham o **mesmo** **ambiente** de **execução** (variáveis ​​globais, espaço de endereço, arquivos abertos, etc.). Para ordenar a maneira como os threads acessam os dados comuns alguns mecanismos de **sincronização** devem ser **usados**.

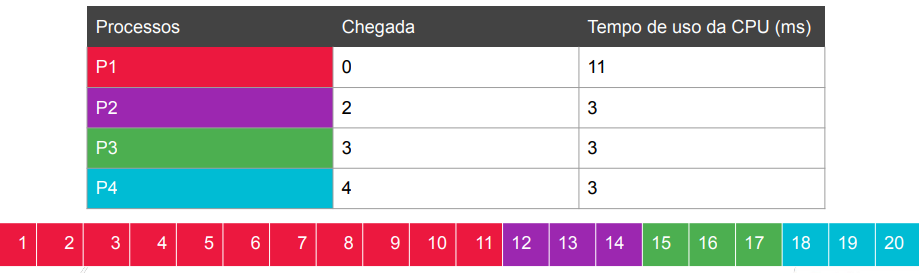
Algoritmos de Escalonamento

O **escalonador** do processador tem como missão **atribuir** o processador aos **processos** que estão **preparados na fila.**

Políticas de Escalonamento

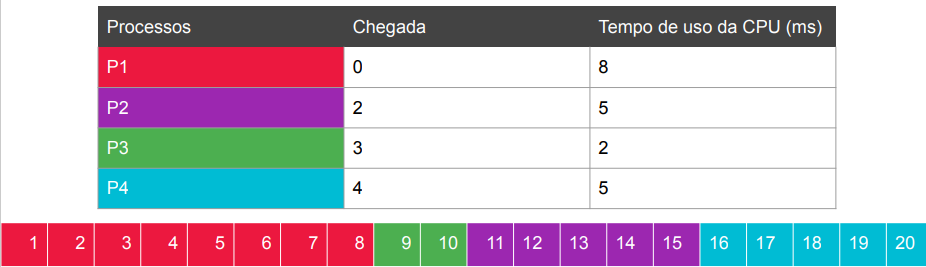
Primeiro a chegar, primeiro a sair (**FIFO**)

Nessa política o **processador** **executa** cada processo até seu término, os processos que estão na fila de processados preparados permanecerão enfileirados na **ordem em que chegaram até a hora de executá-los.** Também conhecido como “**primeiro a entrar, primeiro a sair**”.



O próximo processo mais curto (**SJF**)

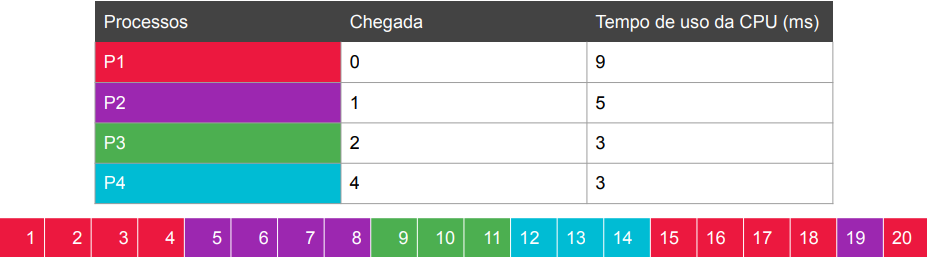
Essa política tira da fila de processos preparados aqueles que precisam de menos tempo de execução para fazer seu trabalho. Ele tem o principio do FIFO, mas caso chegue algum processo com **menos tempo de execução**, ele entra na fila **com prioridade**.



Round-Robin (**RR**)

Tem **um tempo para o processo terminar**, caso não termine, ele volta para o final da fila de processos preparados.

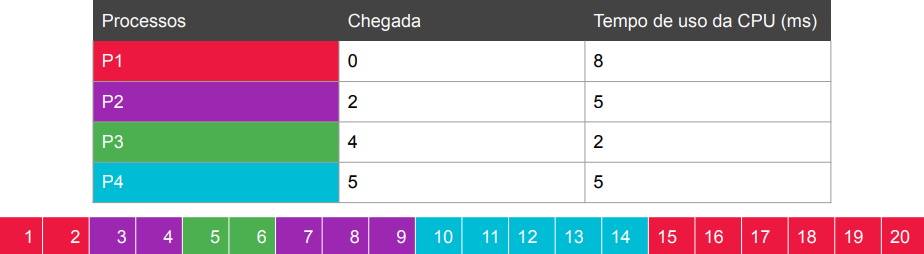
Nessa política cada processo de execução **tem um determinado período de tempo**, após o qual, se o processo **não tiver terminado**, ele é **devolvido** ao **final da fila** de processos preparados, sendo o **processador cedido** ao **próximo processo** por seu quantum correspondente.



Próximo processo, o menor tempo restante (**SRTF**)

Basicamente, quando aparece um **processo que leva menos tempo** para ser **executado** que aquele que está em execução, o **processador** dá **prioridade** para o que leva **menos tempo e altera**.

Essa técnica altera o processo em execução, uma vez que você executa um processo com um requisito de tempo de execução total inferior do que aquele em execução no processador.



Múltiplas Filas

Essa política **divide** a fila em processos **preparados** em **várias** filas **separadas**, de modo que os processos sejam **atribuídos** a uma determinada **fila com base em suas necessidades e tipos.**