**1. FIFO (First-In, First-Out) ou FCFS (First-Come, First-Served)**

**É o algoritmo mais intuitivo, tratando os processos como uma fila simples.**

* **Como funciona: Os processos são atendidos na ordem exata em que chegam. O primeiro a entrar na fila de prontos é o primeiro a ser executado e só libera a CPU ao terminar ou ao solicitar uma operação de I/O (Entrada/Saída).**
* **Categoria: Não preemptivo.**
* **Vantagens: Simplicidade de implementação e ausência de inanição (starvation).**
* **Desvantagens: Suscetível ao "Efeito Comboio", onde um processo longo atrasa todos os processos subsequentes, aumentando drasticamente o tempo médio de espera. É inadequado para sistemas interativos.**

**Exemplo Prático Detalhado**

**Considere os seguintes processos:**

|  |  |  |
| --- | --- | --- |
| **PROCESSO** | **TEMPO DE CHEGADA** | **TEMPO DE EXECUÇÃO** |
| **P1** | **0** | **7** |
| **P2** | **2** | **4** |
| **P3** | **4** | **1** |
| **P4** | **5** | **4** |

**Linha do Tempo (Diagrama de Gantt):**

**| P1 (7ms) | P2 (4ms) | P3 (1ms) | P4 (4ms) |**

**0 7 11 12 16**

**Cálculo do Tempo de Espera:**

**Tempo de Espera = Tempo de Conclusão - Tempo de Chegada - Tempo de Execução**

* **P1: (7 - 0) - 7 = 0 ms**
* **P2: (11 - 2) - 4 = 5 ms**
* **P3: (12 - 4) - 1 = 7 ms**
* **P4: (16 - 5) - 4 = 7 ms**
* **Tempo Médio de Espera: (0 + 5 + 7 + 7) / 4 = 4.75 ms**

**2. SJF (Shortest Job First)**

**Prioriza a execução dos processos mais curtos para otimizar a vazão e minimizar o tempo de espera geral.**

* **Como funciona: O escalonador seleciona o processo com o menor tempo de execução estimado.**
* **Categoria: Pode ser não preemptivo ou preemptivo. A versão preemptiva é conhecida como SRTF (Shortest Remaining Time First) e interrompe o processo atual se um novo processo mais curto chegar.**
* **Vantagens: Provadamente ótimo para minimizar o tempo médio de espera.**
* **Desvantagens: Requer conhecimento prévio do tempo de execução (impraticável na realidade) e pode levar à inanição de processos longos.**

**Exemplos Práticos Detalhados**

**Usando o mesmo conjunto de processos:**

**A) SJF Não Preemptivo**

* **No tempo 0, P1 inicia e executa até o fim, pois é não preemptivo.**
* **No tempo 7, P1 termina. P2, P3 e P4 já chegaram. O escalonador escolhe o mais curto entre eles: P3 (1ms).**
* **No tempo 8, P3 termina. Entre P2 (4ms) e P4 (4ms), ocorre um empate. O critério de desempate é FCFS, então P2 é executado.**
* **No tempo 12, P2 termina e P4 é executado.**

**Linha do Tempo (Diagrama de Gantt):**

**| P1 (7ms) | P3 (1ms) | P2 (4ms) | P4 (4ms) |**

**0 7 8 12 16**

**Cálculo do Tempo de Espera:**

* **P1: (7 - 0) - 7 = 0 ms**
* **P2: (12 - 2) - 4 = 6 ms**
* **P3: (8 - 4) - 1 = 3 ms**
* **P4: (16 - 5) - 4 = 7 ms**
* **Tempo Médio de Espera: (0 + 6 + 3 + 7) / 4 = 4.0 ms**

**B) SJF Preemptivo (SRTF)**

* **t=0: P1 inicia (tempo restante = 7).**
* **t=2: P2 chega (execução = 4). O tempo restante de P1 é 5. Como 4 < 5, P1 é interrompido e P2 inicia.**
* **t=4: P3 chega (execução = 1). O tempo restante de P2 é 2. Como 1 < 2, P2 é interrompido e P3 inicia.**
* **t=5: P3 termina. P4 chega (execução = 4). Processos na fila: P1(rest=5), P2(rest=2), P4(exec=4). O mais curto é P2, que volta a executar.**
* **t=7: P2 termina. Processos na fila: P1(rest=5), P4(exec=4). O mais curto é P4, que inicia.**
* **t=11: P4 termina. Apenas P1 resta. Ele volta a executar seus 5ms restantes.**
* **t=16: P1 termina.**

**Linha do Tempo (Diagrama de Gantt):**

**| P1(2) | P2(2) | P3(1) | P2(2) | P4(4) | P1(5) |**

**0 2 4 5 7 11 16**

**Cálculo do Tempo de Espera:**

* **P1: (16 - 0) - 7 = 9 ms**
* **P2: (7 - 2) - 4 = 1 ms**
* **P3: (5 - 4) - 1 = 0 ms**
* **P4: (11 - 5) - 4 = 2 ms**
* **Tempo Médio de Espera: (9 + 1 + 0 + 2) / 4 = 3.0 ms**

**3. Round Robin (RR)**

**Projetado para sistemas de tempo compartilhado, garante que nenhum processo monopolize a CPU.**

* **Como funciona: Cada processo recebe uma fatia de tempo, chamada *quantum*. Se não terminar nesse intervalo, é interrompido e movido para o final da fila, dando a vez ao próximo.**
* **Categoria: Preemptivo.**
* **Vantagens: Excelente tempo de resposta e justiça entre os processos.**
* **Desvantagens: A performance é sensível ao tamanho do *quantum*. Um *quantum* muito pequeno aumenta a sobrecarga com trocas de contexto.**

**Exemplo Prático Detalhado (Quantum = 3ms)**

**Linha do Tempo (Diagrama de Gantt):**

**| P1(3) | P2(3) | P1(3) | P3(1) | P4(3) | P2(1) | P1(1) | P4(1) |**

**0 3 6 9 10 13 14 15 16**

* **Execução: A CPU alterna entre os processos P1, P2, P1, P3, P4, P2, P1, P4 conforme eles chegam e usam suas fatias de tempo, até todos terminarem.**

**Cálculo do Tempo de Espera:**

* **P1: (15 - 0) - 7 = 8 ms**
* **P2: (14 - 2) - 4 = 8 ms**
* **P3: (10 - 4) - 1 = 5 ms**
* **P4: (16 - 5) - 4 = 7 ms**
* **Tempo Médio de Espera: (8 + 8 + 5 + 7) / 4 = 7.0 ms**

**4. Escalonamento por Prioridade**

**Associa uma prioridade a cada processo, alocando a CPU ao de maior prioridade.**

* **Como funciona: O escalonador seleciona o processo pronto com o maior valor de prioridade.**
* **Categoria: Pode ser preemptivo ou não preemptivo.**
* **Vantagens: Permite que processos críticos tenham precedência, essencial para sistemas de tempo real.**
* **Desvantagens: Risco de inanição para processos de baixa prioridade. Pode ser mitigado com a técnica de *aging* (envelhecimento), que aumenta a prioridade de processos que esperam por muito tempo.**

**Exemplo Prático Detalhado (Preemptivo, 1 = maior prioridade)**

|  |  |  |  |
| --- | --- | --- | --- |
| **PROCESSO** | **TEMPO DE CHEGADA** | **TEMPO DE EXECUÇÃO** | **PRIORIDADE** |
| **P1** | **0** | **7** | **2** |
| **P2** | **2** | **4** | **1** |
| **P3** | **4** | **1** | **4** |
| **P4** | **5** | **4** | **3** |

* **t=0: P1 inicia (Prio=2).**
* **t=2: P2 chega (Prio=1). Como 1 > 2, P1 é interrompido e P2 inicia.**
* **t=6: P2 termina. Na fila, temos P1(rest=5, Prio=2), P3(exec=1, Prio=4), P4(exec=4, Prio=3). P1 tem a maior prioridade e volta a executar.**
* **t=11: P1 termina. Na fila, P4(Prio=3) tem prioridade maior que P3(Prio=4). P4 inicia.**
* **t=15: P4 termina. P3 inicia.**
* **t=16: P3 termina.**

**Linha do Tempo (Diagrama de Gantt):**

**| P1(2) | P2(4) | P1(5) | P4(4) | P3(1) |**

**0 2 6 11 15 16**

**Cálculo do Tempo de Espera:**

* **P1: (11 - 0) - 7 = 4 ms**
* **P2: (6 - 2) - 4 = 0 ms**
* **P3: (16 - 4) - 1 = 11 ms**
* **P4: (15 - 5) - 4 = 6 ms**
* **Tempo Médio de Espera: (4 + 0 + 11 + 6) / 4 = 5.25 ms**

**Resumo Comparativo e Conclusão**

**Comparativo do Tempo Médio de Espera (Exemplos Acima)**

|  |  |  |
| --- | --- | --- |
| **ALGORITMO** | **TEMPO MÉDIO DE ESPERA** | **IDEAL PARA** |
| **SRTF (SJF PREEMPTIVO)** | **3.0 MS** | **MAXIMIZAR VAZÃO** |
| **SJF NÃO PREEMPTIVO** | **4.0 MS** | **SISTEMAS EM LOTE (BATCH)** |
| **FCFS** | **4.75 MS** | **SISTEMAS MUITO SIMPLES** |
| **PRIORIDADE PREEMPTIVO** | **5.25 MS** | **SISTEMAS DE TEMPO REAL** |
| **ROUND ROBIN** | **7.0 MS** | **SISTEMAS INTERATIVOS** |

**Qual algoritmo escolher?**

* **Sistemas Interativos (Desktops, Smartphones): O Round Robin é a escolha dominante, pois garante que a interface do usuário e os aplicativos permaneçam responsivos.**
* **Sistemas em Lote (ex: processamento de grandes tarefas): O FIFO/FCFS pode ser usado pela sua simplicidade, mas o SJF (Não Preemptivo) é superior se houver uma boa estimativa do tempo de execução das tarefas, pois ele maximiza a vazão (número de tarefas concluídas por unidade de tempo).**