

**Cap.5 – Scheduling e CPU**

1. Qual a função de um Scheduler de Curto Prazo? E um Dispatcher?

**R-** O Scheduler de Curto Prazo tem como função tirar os processos da fila de Prontos (READY QUEUE). Ele seleciona entre os processos que estão em estado de pronto na memória, para serem executados pelo processador, após a interrupção de um ciclo, uma interrupção de E/S, uma chamada de sistema ou outra forma de sinal. Assim o escalonador de curto prazo faz decisões de escalonamento muito mais frequentemente que os de médio e longo prazo. Uma decisão de escalonamento deve ser feita no mínimo a cada fatia de tempo, e estas são bem curtas. Já a função do módulo Dispatcher é passar o controle da CPU para o processo selecionado pelo Scheduler de Curto Prazo.

1. De quais maneiras um balanceamento de carga pode ser mantido em sistemas computacionais Multicore?

**R-** Existem duas abordagens para o balanceamento de carga:

**Migração por expulsão:** uma tarefa fica verificando constantemente a carga de cada processador, e quando encontra um desequilíbrio, tenta equilibrar a carga entre processadores.

**Migração por absorção:** quando um processador ocioso extrai uma tarefa que está esperando em um processador ocupado.

\*Ambas as técnicas podem ser usadas de forma conjunta em um SO.

1. Geralmente, quais são os critérios utilizados para comparação de algoritmos? Cite 3 (três) destes critérios e descreva sucintamente o que eles avaliam.

**R-** **Throughput:** Número de processos que são completados por unidade de tempo;

**Tempo de Espera:** Total de tempo que um processo fica na ready queue;

**Tempo de Resposta:** Intervalo de tempo que vai da submissão de um processo até quando este processo produz o seu primeiro resultado.

1. Quais são os principais algoritmos utilizados pelo SO para Scheduling de Processos? Escreva de forma sucinta como funciona cada um deles.

**R- FCFS (First-Come First Server):** O primeiro processo a chegar é o primeiro a ser servido.

**SJF (Shortest Job First):** Nesse algoritmo, é selecionado para ser executado o processo com o menor tempo de execução (Processo que tem o próximo pico de CPU mais curto).

**PS (Priority Scheduling):** Nesse algoritmo, uma prioridade é associada a cada processo. Os processos com maior prioridade devem ser executados primeiro, enquanto os trabalhos com prioridades iguais são executados em rodízio ou FCFS.

**RR (Round Robin Scheduling):** Nesse algoritmo, fatias de tempo (também conhecidas como quantum) são atribuídas a cada processo em partes iguais e em ordem circular, manipulando todos os processos sem prioridade (também conhecido como executivo cíclico).

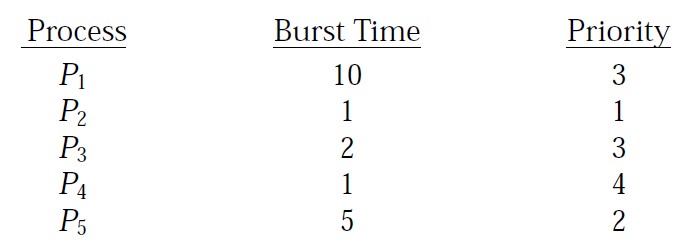
**MQS (Multilevel Queue Scheduling):** Algoritmo para situações em que os processos são facilmente classificados em diferentes grupos. Por exemplo, uma divisão comum é feita entre processos de Foreground (iterativos) e Background (batch). Nesse algoritmo, a ready queue é dividida em várias filas separadas e cada processo é permanentemente designado para somente uma das filas, de acordo com suas características. Cada fila pode ter seu próprio algoritmo de Scheduling e cada fila tem prioridade absoluta sobre filas de menor prioridade. Existe um Scheduler principal entre as filas, de prioridade fixa e com preempção.

**MFS (Multilevel feedback-queue Scheduling):** Esse algoritmo tem o funcionamento parecido com o MQS, porém, permite a alternância de processos entre filas. Os Processos são separados de acordo com seus CPU-Bursts (Picos de CPU). Processos de CPU-Bound vão se deslocando para as filas de mais baixa prioridade e Processos de IO-Bound e interativos vão se deslocando para filas de mais alta prioridade.

1. O que você entende por Processos CPU-Bound e IO-Bound?

**R-** Um processo com longos picos de CPU é denominado CPU-Bound e um processo com curtos picos de CPU é denominado IO-Bound. (IO-Bound refere-se a cargas de trabalho de IO (disco, redes, dispositivos) e CPU-Bound às cargas de trabalho de CPU (algoritmos, cálculos)).

1. Considere o conjunto de Processos a seguir, com duração do pico de CPU dada em milissegundos:

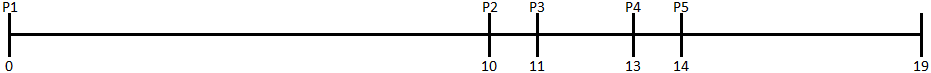


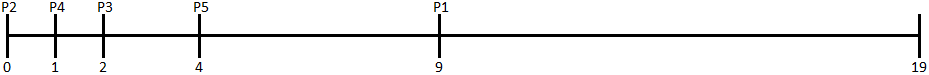
Presume-se que os processos tenham chegado na ordem P1, P2, P3, P4 e P5, todos no momento 0.

a. Desenhe 4 (quatro) gráficos de Gantt que ilustrem a execução desses processos usando os Algoritmos de Scheduling a seguir: FCFS, SJF e PS sem Preempção e RR (Quantum = 1);

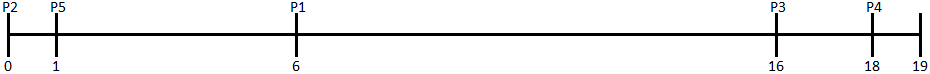
b. Qual é o tempo de espera de cada processo para cada um desses algoritmos de Scheduling?

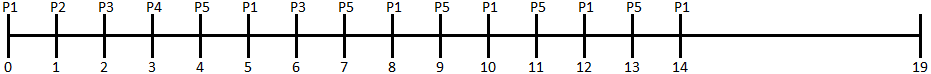
c. Qual dos Algoritmos resulta no menor tempo médio de espera (para todos os processos)?

**1 – FCFS**

**2 – SJF**

TEPx = última vez que entrou p/ executar – o que já executou antes – arrival time

**3 – PS Não-Preemptivo**

**4 – RR (Quantum = 1)**

b)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Processo | FCFS | SJF | PS | RR |
|  | 0 | 9 | 6 | 9 |
|  | 10 | 0 | 0 | 1 |
|  | 11 | 2 | 16 | 5 |
|  | 13 | 1 | 18 | 3 |
|  | 14 | 4 | 1 | 9 |

c) FCFS: 9,6ms / SJF: 3,2ms / PS Não-Preemptivo: 8,2ms / RR (Quantum = 1): 5,4ms

**R-** SJF (Shortest Job First).

1. Considere o conjunto de processos a seguir que chegam em momentos distintos, com os picos de CPU dados em milissegundos. Qual é o tempo médio de espera destes processos utilizando o algoritmo PS:

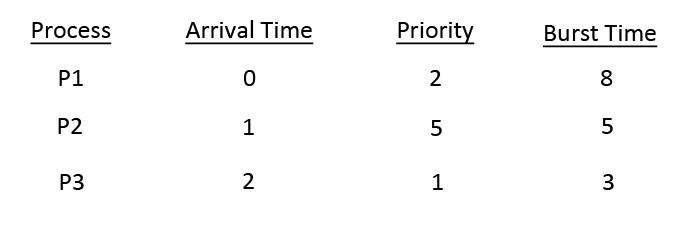
a. Não-Preemptivo.

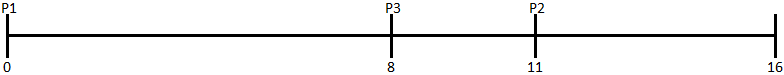
b. Preemptivo.

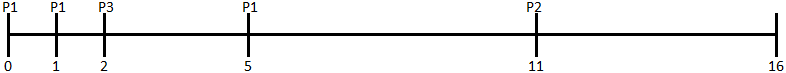
Obs:

• Para facilitar o desenvolvimento, desenhe dois gráficos de Gantt;

• Atenção com o detalhe do Arrival Time dos processos.



**a. PS Não-Preemptivo**

**b. PS Preemptivo**

1. O que significa um Processo ter afinidade com o Processador?

**R-** Essa afinidade entre Processo e Processador significa que um Processo buscará sempre ser executado em um mesmo processador para tentar, sempre que possível, aproveitar dados deixado por ele na cache em outros momentos.

1. O que você entende por Queda de Memória (Memory Stall)? O que pode ser feito/utilizado em um processador para evitar que esta etapa atrase o processamento de dados?

**R-** Quando um Processador acessa a memória, ele gasta um tempo significativo esperando até os dados ficarem disponíveis. Essa situação é conhecida como Queda de Memória.

Para evitar que esta etapa atrase o processamento de dados pode-se atribuir 2 Threads de Hardware a cada núcleo. Assim, se um Thread for interrompido enquanto espera a memória, o núcleo poderá passar para outro Thread (Hyper-Threading).