UNIVERSIDADE SÃO JUDAS TADEU

Sistemas da informação

Projeto integrado

81724388 Gregory Dias

81728264 Augusto da Silva

818231656 Samuel Barreto de Oliveira

819143064 João Pedro Anunciação Silva

817127990 Mariana Silva Lima

81716917 Giulliano Pires Gianelli

Projeto Interdisciplinar

ESCALONAMENTO DE PROCESSOS

Sistemas Operacionais

São Paulo

2019

Sumário

1 Algoritmos de escalonamento de processos

Com a intenção de promover um *overview* sobre escalonamento de processos, o escalonador é responsável por decidir e priorizar qual momento cada processo irá contemplar a CPU, a lógica é feita pelos algoritmos de escalonamento e para que a CPU não fique muito tempo sem executar uma tarefa, o SO utiliza de técnicas para escalonar o processo que estão em execução na máquina.

O escalonamento é muito atuante nos sistemas em tempo real, onde o tempo é um fator extremamente crítico, como: aviões, hospitais, usinas nucleares, bancos, multimídia, etc. Em ambientes como estes, quando um sistema tem respostas com atraso, é tão ruim quanto não obter. Tipos de sistemas em tempo real:

- Hard Real Time onde atrasos não são tolerados (aviões, usinas nucleares, hospitais);
- Soft Real Time quando atrasos são tolerados (Bancos, Multimídia).

O escalonador do SO utiliza alguns critérios de escalonamento, como:

- A taxa de utilização de CPU, que é a fração de tempo durante a qual ela está sendo ocupada;
- Throughput que são números de processos terminados por unidade de tempo;
- Turnaround que é o tempo transcorrido desde o momento em que o software entra e o instante em que termina sua execução;
- Tempo de resposta: intervalo entre a chegada ao sistema e inicio de sua execução;
- Tempo de espera: soma dos períodos em que o processo estava no seu estado pronto.

OS responsáveis por essa tarefa são algoritmos de escalonamento. Os sistemas operacionais utilizam combinações deles para melhor escalonar os processos.
Neste documento, veremos como os principais algoritmos funcionam, seus processos, vantagens, desvantagens e seus comportamentos em determinadas situações em um SO.

1.1 Escalonamento SJF

O escalonamento de processos Shortest Job First (processo mais curto primeiro) seleciona para ser executado na CPU o processo com o menor tempo de execução.

O Algoritmo estima o tempo prévio de execução de todos os processos, com esse conhecimento o processo com o menor tempo de execução sai da fila de pronto e entra em execução na CPU. E Atrás do mesmo forma uma fila de processos por ordem crescente de tempo de execução. Processo com tempo iguais, utiliza–se a ordem de chegada (FCFS) o primeiro a entrar na fila que será o primeiro a ser executado.

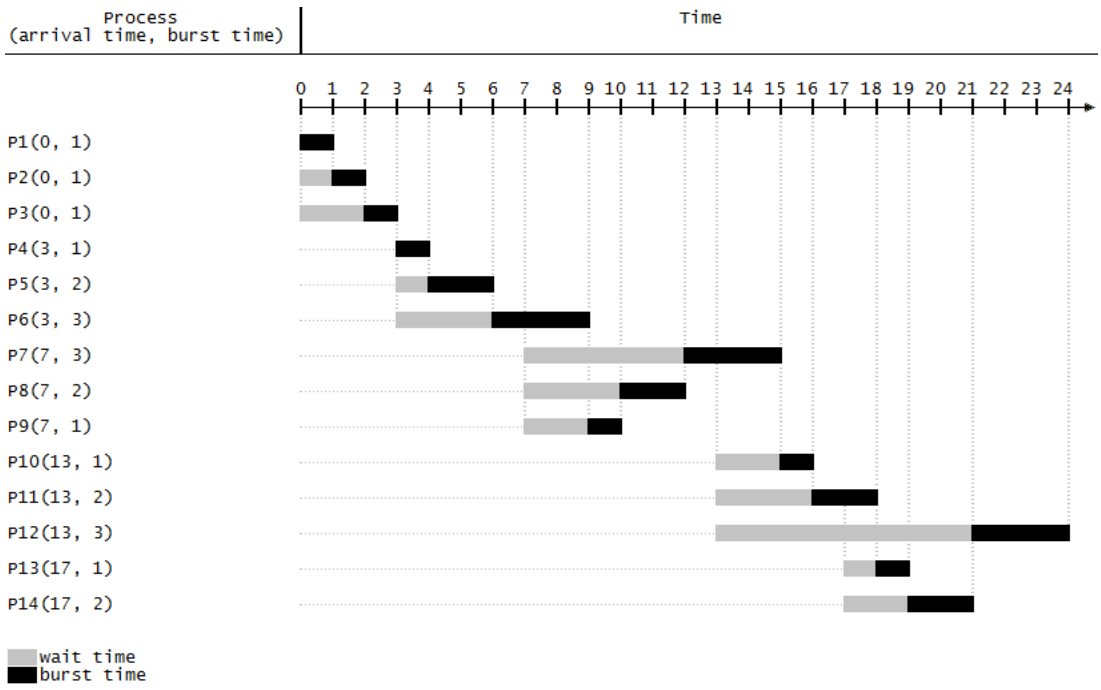


Figura 1: Execução do escalonamento SJT

SJF não–preemptivo – Uma vez atribuída um processo a CPU, este não pode ser desalojado antes de consumir o tempo previsto

SJF preemptivo – Se chegar um novo processo com tempo de uso de CPU inferior ao tempo que resta ao processo em execução, então desalojar o processo que está em execução e entra o outro processo com tempo menor.

O escalonamento SJF é vantajoso por sua simplicidade e porque minimiza o tempo médio que cada processo leva desde quando ele é criado até o fim de sua execução, incluindo o tempo de espera entre o momento em que ele é criado e o momento em que é selecionado para execução.

Essa estratégia pode levar à não execução de processos com longos tempos de execução caso processos curtos sejam continuamente adicionados ao escalonador. Outra desvantagem do SJF é a necessidade de saber previamente os tempos para execução dos processo. Embora seja impossível prever os tempos de maneira exata, existem diversos métodos que podem ser usados para estimá-los, tais como média ponderada ou uso dos tempos de execução anteriores para processos semelhantes.

1.2 Shortest Remaining Time First Scheduling Algorithm

Sobre o algoritmo de agendamento de tempo restante mais curto 'SRTF'. Veremos exemplos diferentes que demonstram o uso do menor tempo restante no primeiro agendamento *(Figura 2)*.

Como o nome sugere, ele seleciona o processo para execução que tem o menor tempo restante até a conclusão. Menor tempo restante O primeiro agendamento é uma versão antecipada do SJF (Shortest Job First). No SJF, uma

vez que o processo começa a execução, ele é executado até a conclusão. No SRTF, um processo em execução pode ser precedido por um processo do usuário com um menor tempo de execução estimado[3].

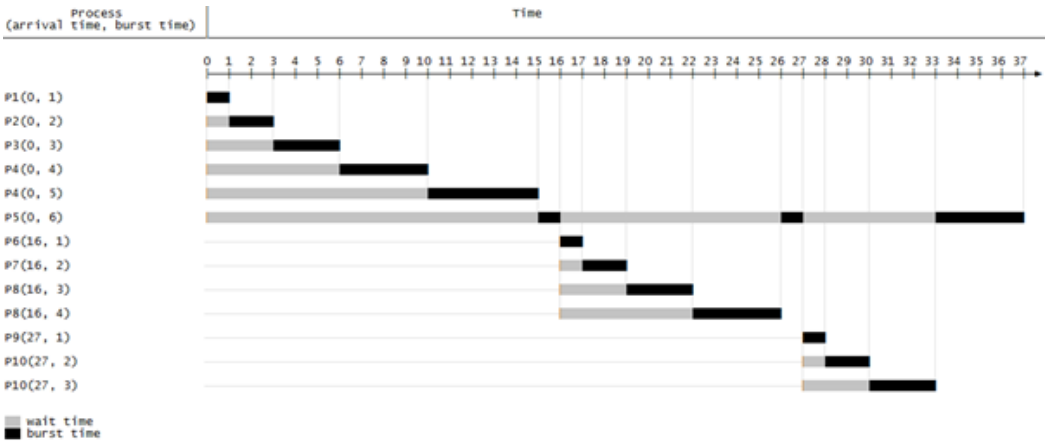


Figura 2: Representação do tempo estimado de execução

Muitos processos são executados em menor quantidade de tempo. Então, o rendimento é aumentado com isso, processos que possuem tempo de burst curto são mais velozes.

Uma desvantagem é que praticamente não é possível prever o tempo de rajada e os processos que têm tempo de rajada longo terão que esperar por longo tempo para execução.

1.3 Round Robin (RR) Algoritmo

Round Robin é um algoritmo que prioriza o processo, ele monta uma fila de acordo com o processo que chegou primeiro, esse processo é executado de acordo com o quantum (é definido pelo sistema operacional), se o processo não é finalizado ele é jogado pro final da fila.

O Round Robin é um algoritmo muito usado e ao mesmo tempo muito antigo , a idéia é que todos os processos são armazenados em uma fila circular, logo, o escalonador da CPU percorre a fila, alocando a CPU para cada um dos

processos durante o seu quantum, ele pega o primeiro que chegou na fila e faz sua execução.

Entretanto, a tarefa é interrompida se o seu tempo quantum acabar e será retomada onde parou no próximo agendamento [1].

Esse processo de escalonamento circular é simples, mas pode trazer problemas se apresentar tempos de execução muito diferentes entre os processos, por exemplo: Quando muitas tarefas estiverem ativas e de longa duração, as tarefas de curta duração terão seu tempo de execução deteriorado devido às tarefas longas reciclarem continuamente na fila circular, compartilhando de maneira equivalente a CPU.

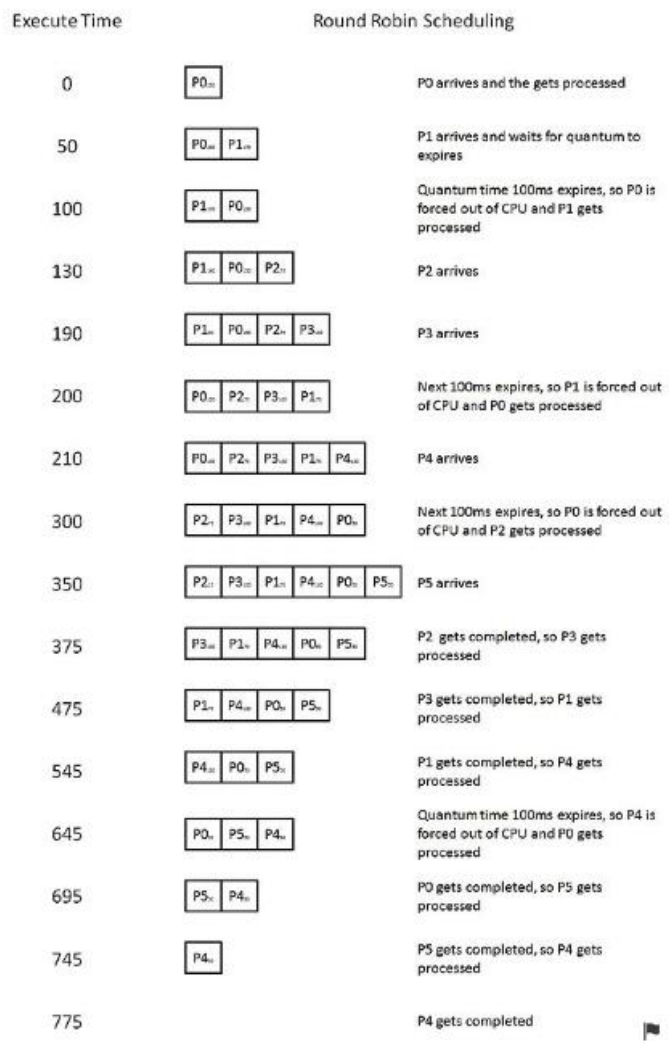


Figura 3: Exemplo do comportamento RR

Na imagem a seguir podemos entender o comportamento de quando o Quantum expira e outro processo assume a fila:

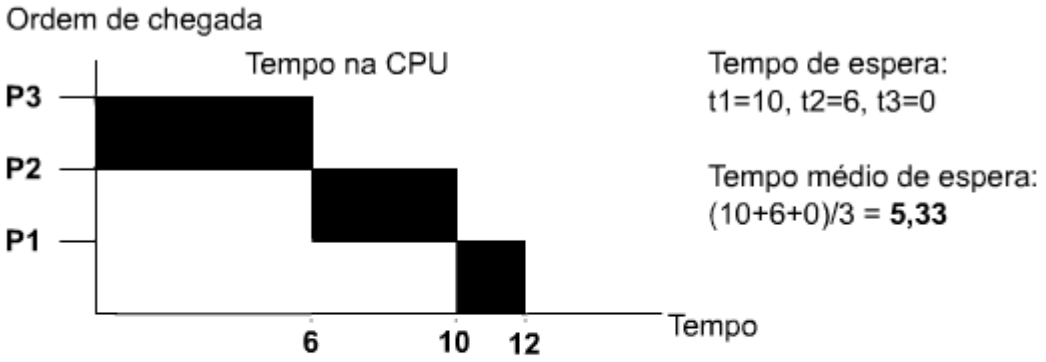
Figura 4: Comportamento Quantum expirado

1.4 First come, First served/ First in, First out

FCFS (“first come, first served” ou "primeiro a chegar, primeiro a ser servido") é um algoritmo de escalonamento para estruturas de dados do tipo fila de processos.

Apresenta o seguinte critério: o primeiro elemento a entrar é o primeiro a ser executado independente do tamanho, é um algoritmo de escalonamento não preemptivo que entrega à CPU os processos por ordem de chegada. Ele executa o processo como um todo do início ao fim sem interrupções, então quando um novo processo chega e existe um ainda em execução ele vai para uma fila de espera.

Figura 5: Primeiro a chegar, primeiro a ser servido!



Neste caso os processos chegaram na ordem P3, P2, P1. Os processos têm tempos iguais a 2, 4, 6.

Assim os tempos de espera de cada processo foram respectivamente 10, 6 e 0. Ao tirar à média obtemos o valor 5,33

Neste tipo de escalonamento, todos os processos tendem a serem atendidos, (por isso evita o fenômeno do starvation*), a menos que um processo possua um erro ou um loop infinito.

O *loop* infinito irá parar a máquina, pois o *FCFS* não terá como dar continuidade a execução dos processos que estão aguardando na fila de espera.

Ele também não garante um tempo de resposta rápido pois é extremamente sensível a ordem de chegada de cada processo e dos antecessores (se existirem) e se processos que tendem a demorar mais tempo chegarem primeiro o tempo médio de espera e o turnaround acabam sendo aumentados.

Pelo critério do primeiro a entrar, primeiro a ser servido, faz o agendamento de tarefas do sistema operacional dando a cada processo tempo de CPU na ordem em que as demandas são feitas. O oposto dele é o *LIFO* (*Last-In, First-Out*), que significa "o último a entrar é o primeiro a sair", aonde a entrada mais recente, ou o topo da pilha de processos, é processado primeiro. Já uma fila prioritária não é nem *FIFO*, nem *LIFO*, mas pode adotar comportamento similar temporariamente, ou mesmo por padrão[4].

As listas são amplamente utilizadas em programação para implementar *filas de espera*. A ideia fundamental desta fila é que só podemos inserir um novo elemento no final da fila e só podemos retirar o elemento do início.

É vantajoso por ser o mais simples entre os processos de escalonamento; e todos os processos tendem a serem atendidos.

Dentre as desvantagens estão: muito sensível a ordem de chegada; se processos maiores chegarem primeiro aumentarão o tempo médio de espera; não garante um tempo de resposta rápido; não é eficiente em sistemas de compartilhamento de tempo; e não é eficiente em sistemas em tempo real.

1.5 Escalonamento por Prioridade

Tabela 1 - Processo x Prioridade

Processo	Prioridade
A	3
B	1
C	3
D	4
E	2

O funcionamento do escalonamento funciona da seguinte forma, quem tiver prioridade maior será o primeiro a ser processado porém isso não significa que aquele que tiver o número ou a letra maior será o primeiro. Alguns sistemas operam da seguinte forma, eles fazem uma fila de 1 a 20 sendo o 1 a prioridade maior e o 20 a prioridade menor, mas em outros tipos de sistema o 20 pode ser considerado maior enquanto o 1 será considerado menor quem define este tipo de parâmetro é a mesma pessoa que desenvolveu o sistema. No exemplo acima o 1 é considerado o maior, logo todos que virão a seguir serão menores que ele.

Na tabela acima a maior prioridade é a letra B devido à prioridade 1 em seguida vem a letra E com prioridade 2, chegamos a um ponto onde temos duas prioridades com o mesmo nível, neste caso aquele processo que chegou primeiro irá ser processado primeiro sendo assim neste caso a letra A será processada antes que a letra C onde a mesma seria processada logo em seguida e para finalizar a última a ser processada seria a letra D com prioridade 4. Mas este tipo

de sistema de prioridade é basicamente inteligente não podemos nos basear em apenas 5 processos temos que imaginar em uma máquina. Em uma máquina existem vários processos e com isso para o processamento não ficar travado em apenas uma prioridade o sistema começa a rodar um pouco de cada um, para assim finalizar a tarefa, imaginemos que entrasse um processo com prioridade 20, mas na sua frente existem vários outros processos onde eles são bem maiores e demorados esta prioridade 20 nunca iria ser rodada se ela ficasse esperando a vez dela entrar, pois sempre entraria alguma prioridade na frente.

Neste tipo de escalonamento temos um termo chamado de starvation ele basicamente opera da seguinte forma imaginemos que temos um servidor em uma sala que está com problema no refrigerador este servidor está trabalhando normalmente com sua lista de prioridades e de repente o servidor por estar trabalhando com muitos processos acaba superaquecendo o sistema automaticamente entende que isso não pode acontecer e para completamente todos os seus processo tornando assim prioridade número 1 diminuir sua temperatura porém isso é temporário já que a temperatura geralmente não chega a superaquecer o servidor e nestes caso a prioridade número 1 se torna resfriar o servidor, assim que tudo se normalizar o servidor volta ao seu processamento normal, o starvation serve para certos casos onde ele entra em ação assim que é detectado alguma falha eminente aparece.

1.6 Escalonamento de Múltiplas Filas

É um algoritmo de escalonamento que cria N filas de processos com base nas características dos processos. As filas possuem um grau de prioridade e tem os processos ordenados internamente com base em outros algoritmos, podendo ser um para cada fila.

Pode ser utilizado em situações nas quais os processos podem ser facilmente classificados em diferentes classes com necessidades de escalonamento únicas.

Um caso comum é quando possuímos "*System Processes*" (Processos criados pelo sistema), "*Foreground Processes*" (Processos de interação direta do usuário) e "*Background Processes*" (Processos que não possuem intervenção do usuário), esses tipos de processos possuem necessidades distintas de escalonamento e podem ser agrupados separadamente.

Como é possível ver na Figura 6 podemos priorizar as filas e cada fila pode possuir seu próprio algoritmo de escalonamento

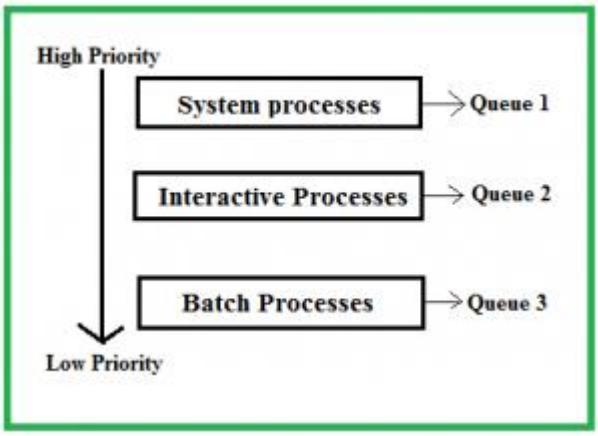


Figura 6: Múltiplas filas - Prioridade

Para definir a rotatividade entre as filas é necessário escolher entre o método de escalonamento preemptivo de prioridade fixa, no qual cada fila tem prioridade absoluta sobre as filas de menor prioridade, e o método de partição de tempo, no qual cada fila utiliza uma parte do tempo da CPU para escalonar seus próprios processos.

1.7 Escalonamento de CPU

- Algoritmo próximo processo mais curto

. Mesma idéia do Shortest Job First dos sistemas em lote;

. Problema: Processos Interativos não se conhece o tempo necessário para execução;

. Como empregar esse algoritmo: ESTIMATIVA de TEMPO.

- Algoritmo garantido:
    . Garantias são dadas aos processos dos usuários:
    . n usuários -> 1/n do tempo de CPU para cada usuários;
    .
- Algoritmo por loteria:
    . O Sistema Operacional distribui tokens (fichas), numerados entre os processos, para o escalonamento é sorteado um numero aleatório para que o processo ganhe a vez na CPU, processos com mais tokens têm mais chance de receber antes a CPU;
- Algoritmo por fração justa:
    . O dono do processo é levado em conta;
    . Se um usuário A possui mais processos que um usuário B, o usuário A terá prioridade no uso da CPU;

2. Gerenciamento e Alocação de Memória

Para aprimorarmos os estudos de sistemas operacionais e seus algoritmos de escalonamento, é preciso se aprofundar em um campo complexo da ciência da

computação que é o *gerenciamento de memória*, ou *gestão da memória*. Atualmente são desenvolvidas várias técnicas para aperfeiçoar deixando esse gerenciamento mais simples e mais eficiente.

A memória é um dos principais recursos de um computador, e a cada avanço na ciência da computação a memória exigida é maior, portanto, vamos entender como funciona a hierarquia de memórias:

1. Memória cache:
    - Pequena quantidade, volátil, muito rápida, alto custo.
2. Memória principal (RAM):
    - Quantidade razoável, volátil, velocidade média, custo médio.
3. Memória secundária
    - Grande quantidade, não volátil, velocidade baixa, custo baixo.

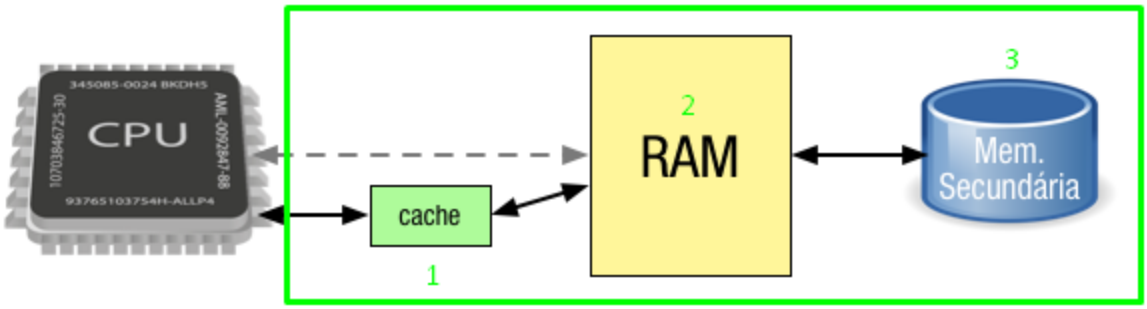


Figura 7 - Hierarquia das memórias

O gerenciador de memória é parte do sistema operacional que tem como objetivo ser responsável por cuidar das memórias que estão em uso, ao mesmo tempo deve ser capaz de cuidar das que estão livre e alocar a memória para quando os processos precisarem e por fim deslocar essa memória quando o processo não necessita mais dela.

Nós temos algumas maneiras de gerenciar a memória:

1. Gerenciamento sem Troca ou Paginação: Método utilizado de movimentação da memória para o disco e vice versa durante a execução do processo.
2. Monoprogramação sem Troca ou Paginação: Único processo sendo executado uma vez usando toda memória disponível (com exceção da parte reservada ao SO)
3. Gerenciamento de espaço: Duas formas de utilização da memória:

A) Gerenciamento com Mapa de Bits

B) Gerenciamento com Lista Encadeada

Gerenciamento com Mapa de Bits:

As memórias são divididas em unidades de um certo tamanho e essa unidade é associada a um bit. O tamanho precisa ser meticulosamente escolhido, pois a desvantagem de quando um novo processador ocupa K unidades de memória o gerenciador deve percorrer esse mapa e encontrar os bits consecutivos o que não é um processo simples.

Gerenciamento com Lista Encadeada:

Lista encadeada de segmentos alocados e livres, sendo que cada segmento é um buraco entre dois processos. A lista apresenta-se em ordem de endereços e quando um processo termina, ou é enviado para o disco, a lista é atualizada. Alguns algoritmos que utilizam essa técnica são:

- First Fit (primeiro encaixe): Procura na fila o primeiro espaço que caiba o processo, é um algoritmo de velocidade rápida.

- Next Fit (próximo encaixe): O mesmo que o First Fit só que procura sempre a partir do último ponto em que encontrou, é um algoritmo de velocidade rápida.
- Best Fit (melhor encaixe): Verifica toda a lista e procura o buraco com espaço mais próximo, é um algoritmo de velocidade lenta com desempenho pior que o First Fit.
- Worst Fit (pior ajuste): Pega sempre o maior buraco disponível, é um algoritmo de velocidade lenta e desempenho ruim.

Pra finalizar temos mais duas técnicas de gerenciamento de memória:

Alocação de espaço de troca (Swap):
Espaço ocupado no disco pelos processos que estão guardados, que foram retirados da memória devido a uma troca. Os algoritmos para gerenciar o espaço alocado em disco para swap são os mesmos apresentados para gerenciamento de memória. O que difere, é que alguns sistemas, cada processo tem um espaço reservado no disco e a memória é constantemente mudada de lugar.

Memória Virtual:
Foi a primeira solução absorvida para programas muito grandes comparados a quantidade de memória, chamada *overlays*. Essa técnica consistia em que o programa era dividido em *overlays* que podiam ser executadas separadamente e quando um *overlay* terminava a execução um outro poderia ser carregado na mesma posição de memória. O problema é a complexidade de divisão do programa em *overlay*, não é simples de se fazer. Dentro da memória virtual temos duas técnicas:

- Paginação: Forma de conseguir grande espaço de endereçamento linear em uma quantidade finita de memória física.

- Segmentação: Espaço bidimensional, dividido em um certo número de segmentos, cada um com dado número de bytes.

3. Conclusão

Escalonamento de processos é o ato de realizar o chaveamento dos processos ativos, de acordo com regras bem estabelecidas, de forma que todos os processos tenham chance de utilizar a CPU. O escalonador é a parte do SO encarregada de decidir entre os processos prontos, qual será colocado em execução.

Não existe apenas um escalonador, assim como não existe apenas um processo. E toda essa organização, quando bem distribuída, torna produtivo, o desempenho dos processos.

Entendemos o conceito de "processos" como vital para um Sistema Operacional, a maioria deles utiliza de sistemas multitarefas, e assim vários desses processos de programas estão usando ao mesmo tempo o Quantum de uma CPU.

O uso aprimorado do tempo da CPU, vem sendo estudado ao longos dos anos, e vários algoritmos foram criados para promover a melhor forma de um processo utilizar a performance da CPU. Com esse estudo, pudemos observar mais de perto como alguns algoritmos funcionam, suas especificidades, peculiaridades e como eles se comportam.

Então podemos concluir a eficiência de alguns escalonadores, e a ineficiência de outros, e notar também, de forma pratica no dia-a-dia, que praticamos filas de escalonamento, mesmo que de maneira indireta.

4. Lista de Figuras e Tabelas

5. Referência Bibliográfica

[1] BRENO, RIBA. **"A simplicidade e a importância do Round Robin como técnica de balanceamento"**. IMASTER. Disponível em: <https://imasters.com.br/apis-microsservicos/simplicidade-e-importancia-round-robin-como-tecnica-de-balanceamento>
Acesso em 9 de mar. 2019.

WIKIPEDIA - AUTORIA LIVRE. **Inanição(Computação) - Starvation.** Disponivel em: <http://www.inf.ufsc.br/~bosco.sobral/ensino/ine5645/Inanicao.pdf>
Acesso em 17 mar. 2019.

[3] WIKIPEDIA - AUTORIA LIVRE. **Shortest Job First**. Disponível em: <https://pt.wikipedia.org/wiki/Shortest_job_first>
Acesso em 12 mar. 2019.

[2] **MULTILEVEL QUEUE SCHEDULING**. Study Tonight. Disponível em: <https://www.studytonight.com/operating-system/multilevel-queue-scheduling>
Acesso em 17 mar. 2019

[4] **FIRST COME FIRST SERVE**. Study Tonight. Disponivel em: <https://www.studytonight.com/operating-system/first-come-first-serve>
Acesso em 03 abr. 2019

OPERATING SYSTEM. GeeksForGeeks. Disponivel em: em<https://www.geeksforgeeks.org/operating-system-multilevel-queue-scheduling/>
Acesso em 10 mar. 2019.

COLLETA, ALEX "**Gerenciamento de memória**" Disponível em: <https://alexcoletta.eng.br/artigos/gerenciamento-de-memoria/>
Acesso em 16 abr. 2019

GONÇALVES, ANDRÉ LUIZ "Escalonamento de processos - SJF (Shortest Job First) "Disponível em:
<https://www.youtube.com/watch?v=1F58irY068s>
Acesso em 17 mar. 2019

SOARES, RODRIGO "Sistemas Operacionais escalonamento por prioridades" Disponível em:
< https://www.youtube.com/watch?v=Suy28jqWS-0>
Acesso em 17 mar. 2019