INSTITUTO FEDERAL DO RIO GRANDE DO NORTE

SISTEMAS PARA INTERNET

SISTEMAS OPERACIONAIS

Douglas da Silva Confessor

Lucas Venício de Araújo Nunes

**Algoritmos de escalonamento do braço do Disco**

Parnamirim**,** 2016

Douglas da Silva Confessor

Lucas Venício de Araújo Nunes

**Algoritmos de escalonamento do braço do Disco**

Relatório apresentado como requisito parcial para obtenção de aprovação na disciplina de Sistemas Operacionais, no Curso de Sistemas para Internet, no Instituto Federal do Rio Grande do Norte de Parnamirim.

Prof. Tadeu Ferreira Oliveira.

Parnamirim**,** 2016

**Comparação dos três algoritmos apresentados: *First Come First Served, Shortest Seek First e Algoritmo do Elevador***

O algoritmo *First Come First Served*, é uma estrutura de dados que apresenta o seguinte critério: O primeiro elemento a ser retirado é o primeiro que tiver sido inserido, é um algoritmo de escalonamento não preemptivo que entrega a CPU os processos pela ordem de chegada. Ele executa o processo como um todo do início ao fim não interrompendo o processo executado até ser finalizado, então quando um novo processo chega e existe um ainda em execução ele vai para uma fila de espera. Esta fila de espera nada mais é do que uma fila que organiza os processos que chegam até eles serem atendidos pela CPU.

Já o *Shortest Seek First* outro algoritmo escalonador, funciona a partir de um conceito bem simples: os processos menores terão prioridade, ou seja, serão executados primeiro. Isso tem como resultado um tempo médio mínimo de espera para cada conjunto de processos a serem executados. O cálculo de cada tempo médio é feito a partir da próxima alocação de CPU, ou seja, o processo que utilizar a CPU por menos tempo será executado primeiro.

O *Algoritmo do Elevador,* também conhecido como SCAN, é um algoritmo de escalonamento que opera de forma bastante simples. Ele estabelece um sentido de varredura do cabeçote. Por exemplo, em um dado momento o SCAN pode determinar que o cabeçote se movimentará da parte interna dos pratos para a parte externa. Neste caso, as requisições serão atendidas de acordo com qual cilindro deseja-se acessar: os mais internos serão atendidos primeiro. Quando o cabeçote chega à parte mais externa dos pratos, o sentido de varredura é invertido. Desta forma, a prioridade dos atendimentos também. O SCAN continua neste ciclo até que todas as requisições sejam atendidas. Uma maneira de visualizar o funcionamento do SCAN é pensando no elevador de um prédio comercial, daí o nome “*Algoritmo do Elevador”.*

Cada algoritmo apresenta suas vantagens e desvantagens, sendo comparadas uma ao outro. A vantagem do algoritmo *First Come First Served*, como sempre é a sua simplicidade, e todos os processos tendem a serem atendidos de forma “justa”, ou seja, as requisições são atendidas na ordem. Por outro lado, tem seus pontos fracos, se processos maiores chegarem primeiro aumentarão o tempo médio de espera, com isso não garantindo um tempo de resposta rápido, e além disso não é eficiente em sistemas de tempo compartilhado e nem em sistemas de tempo real. O *Shortest Seek First,* seleciona a requisição com o menor tempo de *Seek* em relação a posição atual do cabeçote de leitura/escrita, com isso tem à redução do tempo de busca e o tempo médio tende a ser mais baixo. Podemos considerar uma desvantagem, a variância alta (ruim para sistema interativos, aceitável para sistemas batch), e também o fato do SJF ser preemptivo, que acontece o bloqueio por tempo indefinido. Um algoritmo de escalonamento por prioridade (preemptivo) pode deixar que alguns processos de baixa prioridade fiquem esperando indefinidamente pela CPU. O *Algoritmo do Elevador,* atende requisições em uma direção preferencial, sendo assim, oferece bons tempos médios de resposta, um bom rendimento, e ainda mais tem uma variância menor se comparado ao SJF, devido a esses motivos o *Algoritmo do Elevador* pode ser considerado o melhor algoritmo no caso médio. Seu ponto fraco é o fato dele não “ser justo”, pois as trilhas das extremidades são “visitadas” menos frequentemente que as trilhas internas. Podemos afirmar que em quase todos os aspectos, o *Algoritmo do Elevador* (e suas variações) tem desempenho superior aos demais.

**Descrição das implementações feitas, dos principais métodos e classes**

No algoritmo First Come First Served, criamos uma classe de mesmo nome, onde a primeira parte do trabalho é ler o dadosDisco.txt. Para isso criamos uma String com o diretório do arquivo que será lido, em seguida uma variável auxiliar que captura a linha do arquivo, número de Setores do HD (int), a posição inicial do braço do HD, e a indicação da trilha do HD. Na segunda parte ele irá lê a primeira linha e converter para inteiro (nSetores), e depois fará o mesmo processo, lendo a segunda linha e convertendo para inteiro (pInicial), feito isso será mostrado na saída o número de setores do HD e a posição inicial do braço do HD. Posteriormente ele irá percorrer todo o resto do arquivo. A última parte do será o somatório de trilhas percorridas pelo braço do HD, em um (for), onde será mostrada as trilhas percorridas pelo braço a cada movimento feito, e por fim mostrará o total das trilhas percorridas.

No Shortest Seek First, podemos considerar a primeira parte feito no First Come First Served, pois a mudança vem logo em seguida, neste caso, o braço do HD irá para a posição mais próxima da atual e não para próxima posição da lista. Criamos variáveis chamadas menorDistancia, trilhaAuxTemp, trilhaAuxTemp2 e positionAux que são variáveis auxiliares que nos ajudaram a organizar o array de trilhas (trilhasPercorridas) de forma que de uma posição para outra do array, teremos a menor distancia possível. Em seguida no (for) irá procurar a posição mais próxima da atual feito isso, fazemos uma troca de posições do array de trilhas para podermos obter a sequência de movimentos que será feita pelo braço do HD. E por fim será feito o somatório de trilhas percorridas pelo braço do HD, processo já explicado anteriormente no First Come First Served.

No Elevator Algorithm, a movimentação do braço de leitura é realizada em apenas uma direção, até que a última trilha seja encontrada. Após isso a movimentação passa a ser no sentido contrário. Sabendo disso, em um (for) iremos encontrar o índice da posição inicial do braço. Logo em seguida serão dois (fors) um para percorrer as trilhas na direção decrescente em relação a posição inicial do braço e outro para percorrer as trilhas na direção crescente em relação a posição inicial do braço. Vale ressaltar, que ele só vai para as trilhas a direita do braço, depois de percorrer todas as trilhas a esquerda, ou seja, comportamento similar ao de um elevador. E para finalizar será exibindo as trilhas percorridas em cada movimento do braço e o somatório de todas as trilhas percorridas.