Luiz Gustavo Akazawa Nora - 2018.1.08.025

Vitor Hugo da Costa Luz - 2018.1.08.023

O código foi implementado na linguagem Java. O tamanho da memória usado foi 1024. Para a representação da memória foi usada um vetor onde cada célula representa um Byte da memória. O projeto foi dividido nas classes Memoria, Bloco, Processo, Main e as classes responsáveis pela implementação da lógica de cada técnica.

A classe Memoria fica responsável por funções como limpar a memória, gerar processos, cuidar da ordem lógica para a execução do algoritmo e exibir a saída de texto. A classe Processo tem como atributos o tamanho e o número do processo.

A classe Bloco foi necessária para trabalhar com um conjunto de células, sendo possível por exemplo guardar os blocos de memória livre e analisar qual deles possui o menor ou maior tamanho para o caso dos algoritmos Worst-Fit e Best-Fit.

Para a execução dos algoritmos foi feita uma interface chamada AlgoritmoInsercao que possui o método abstrato execucao. As outras quatro classes (FirstFit, NextFit, WorstFit, BestFit) implementam essa interface e sobrescrevem o método abstrato cada um com sua funcionalidade específica.

O FirstFit com a característica de inserir o processo no primeiro espaço encontrado com tamanho suficiente, onde buscamos o primeiro bloco que tem tamanho suficiente para inserir nosso processo no momento.

O NextFit com a diferença de memorizar a posição na hora de percorrer a memória, mas ainda coloca o processo no primeiro bloco que couber o processo após a marca de posição atual.

O WorstFit fazendo a inserção no maior espaço de memória em que couber o processo e o Best-Fit no menor espaço em que couber o processo, este dois algoritmos foram implementados com a seguinte lógica: encontrado uma célula do vetor memoria com valor NULL buscamos iterativamente o final do bloco de NULL e caso ele tenha tamanho maior ou igual ao do processo a ser inserido armazenamos este bloco em uma lista de blocos em potencial, após percorrer toda a memória percorremos a lista de blocos em potencial buscando o bloco que atende a lógica do algoritmo .

Quando um processo não cabe na memória um processo aleatório é removido e este é enviado para uma fila de processos em espera. Então é feita uma tentativa de inserir o primeiro processo da fila na memória.

O código apresenta um menu simples para a seleção de qual algoritmo a ser simulado. O código foi desenvolvido no IDE Eclipse e também pode ser executado pelo terminal do Linux.

CONSIDERAÇÕES FINAIS:

Os algoritmos de First-Fit e Next-Fit tem o mesmo desempenho assíntotico mas estes dois em relação aos algoritmos de Best-Fit e WorstF-Fit são melhores pois Best-Fit e Worst-Fit percorrem toda a memoria para encontrar os blocos que suportam o processo e colam em uma lista de candidatos e ao final do algoritmo perccorem esta lista buscando o que atende a definição do algoritmo, por causa disso a complexidade ganha o tempo de percorrer a memoria e gerar os blocos mais o tempo de percorrer a lista de blocos, assim prejudicando o tempo de execução.

Sobre o o tempo de demora para um processo em esperar entrar na memoria, existe a possibilidade de que ele nunca entre pois sempre que é feita uma tentativa de colocar um processo na memoria e ele não é suportado no momento ele volta para o final da lista de espera, este problema pode ser solucionado fazendo-se um lógica de ranking de tentativas onde se um processo foi tenta inserir até X vezes e não deu certo nesta itercação atual forçaremos sua entrada removendo quantos processos forem necessários até ele passar pela memória.

EXECUCAO CODIGO:

Como executar o código, indicamos a utilização do SO Linux para execução do projeto.

Baixe o arquivo compactado e extraia ele em algum diretório, após isso vá até a pasta src dentro de projetoSO e abra o terminal.

Compile o arquivos no package projetoSO, com o seguinte comando javac projetoSO/*.java e depois execute o arquivo Main com o seguinte comando, java projetoSO.Main.

Também é possível executar o projeto através de um IDE eclipse, importanto o projeto em arquivo compactado.