PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS E INFORMÁTICA
UNIDADE EDUCACIONAL PRAÇA DA LIBERDADE
Bacharelado em Engenharia de Software

Mateus Santos Fonseca
Yan Max Rodrigues Sette Pinheiro

Relatório – Trabalho Final de Sistemas Operacionais

Belo Horizonte
2018

SISTEMAS OPERACIONAIS – PROFESSOR: Lesandro Ponciano

1. Desenvolvimento do trabalho

No início do trabalho pensamos primeiramente em sua estruturação, uma vez que a primeira entrega (aquecimento) focaria bastante neste quesito. Além disso, havia o fato de que terminar a estruturação o quanto antes facilitaria o desenvolvimento do trabalho posteriormente. Portanto, o foco inicial do trabalho foi a estruturação e, somente após, o desenvolvimento dos algoritmos.

o Estruturação

O primeiro passo da estruturação foi a leitura dos dados. No momento da leitura, todas as linhas do arquivo de texto que informavam características dos Rs eram armazenadas em cada linha de um vetor e, logo após, os valores indesejados eram retirados das linhas. Após esta inserção no vetor, estes valores foram convertidos para o tipo inteiro e armazenados em uma matriz do tipo inteiro na qual em cada linha há um R e em cada coluna tem os seus respectivos valores A, B, C e D.

O segundo passo da estruturação foi a escrita dos dados. Antes de escrever no arquivo, um método é chamado para remover o conteúdo do out.txt. Após a limpeza, a escrita dos dados foi organizada em um método de forma que todos os algoritmos poderiam utilizá-la. Dentro deste método é feita toda a padronização de saída especificada no trabalho. Após estas definições o método basicamente exporta para o final do arquivo de texto toda vez que é chamado por um algoritmo, as informações do algoritmo que chamou e os tempos de acesso e espera médios calculados por ele.

O terceiro passo da estruturação foi a criação da classe R, cujos atributos são exatamente os valores de A, B, C e D. A classe R possui também um ordenador por posição do cilindro e os getters de cada atributo.

O quarto passo da estruturação foi a criação do método que calcula o tempo de espera médio. Tendo em vista que o tempo de espera médio é calculado da mesma forma para todos os algoritmos, em seu final, todo algoritmo o chamava.

o Desenvolvimento

Com a estruturação toda pronta, o desenvolvimento dos algoritmos se tornou uma tarefa mais simples. Uma arrayList com os valores de cada R foi criada para cada algoritmo visando auxiliar no desenvolvimento e não ser necessário trabalhar matriz. No final de cada algoritmo, é calculado os tempos de acesso e espera médios, e logo após este cálculo, os resultados são exportados.

- FCFS e SSTF

No FCFS, por se tratar de um algoritmo simples no qual o primeiro a chegar será o primeiro atendido, o método percorre os primeiros Rs a chegar e incrementa a soma total do tempo de acesso. Já no SSTF a lógica do cálculo do tempo de acesso médio é igual ao do FCFS. A grande diferença entre o SSTF e o FCFS é que, em vez de acessar os Rs que chegaram primeiro, será acessado o R cujo cilindro destino está mais próximo do cilindro atual. Isto foi feito no algoritmo desenvolvido através da comparação entre o módulo da distância entre todos os cilindros dos Rs ainda existentes (pois quando um R é lido, ele é retirado da lista) e o cilindro do R atual.

- SCAN, C-SCAN e C-LOOK

No SCAN, a forma de calcular o tempo médio de acesso é diferente das anteriormente vistas, primeiramente, os Rs são ordenados de acordo com a posição do cilindro e, outra diferença é o fato de que os Rs são acessados de forma diferente. Para acessar o R, é levado em consideração, além de verificar se ele chegou, a direção à qual ele está indo, ou seja, se ele estiver indo para a direita, somente os cilindros dos Rs que são maiores que o cilindro do R atual serão executados e, se ele estiver indo para a esquerda, somente os cilindros dos Rs menores que o cilindro do R atual serão executados. Quando um R é executado, a soma do tempo de acesso é incrementada de acordo com a fórmula. Além disso, é necessário lembrar que o algoritmo SCAN sempre vai até o final do disco quando está indo para a direita e para o começo quando está indo para a esquerda, portanto o algoritmo desenvolvido leva isto em consideração quando estiver calculando o tempo de acesso.

A lógica do cálculo do tempo de acesso médio do C-SCAN é igual ao do SCAN, a única e grande diferença é que ele não volta atendendo, ou seja, enquanto estiver indo para direita, a lógica do C-SCAN é igual ao do SCAN, mas no momento de retornar o C-SCAN não atende ninguém até chegar no começo do disco, para que, então, comece a ir para a direita e retorne a atender.

No algoritmo C-LOOK, a lógica de cálculo do tempo de acesso médio é igual ao do C-SCAN, a única diferença é que, em vez de ir até o final do disco, ele vai somente até a última requisição, e, no momento de retornar, ele retorna até a primeira requisição, em vez de retornar ao início do disco.

- MY

Em nossa política MY, primeiramente, os Rs são ordenados pela posição do cilindro. Logo após a ordenação, é diretamente procurado para ser executado o primeiro

R a chegar, ou seja, o R com menor tempo de chegada. Após a execução do primeiro R, é contado quantos Rs que chegaram estão à sua direita (em relação a seu cilindro) e quantos Rs que chegaram estão à sua esquerda. Tendo em mãos o lado que possui a maior quantidade de Rs que chegaram, o algoritmo parte para este lado atendendo aos Rs. Outra questão do nosso algoritmo é o fato de que, após a tomada de decisão do lado que irá começar atendendo, ele irá atender tanto indo para a direita quando indo para a esquerda, ou seja, nosso algoritmo atende nas duas direções. Também é um ponto importante informar que nosso algoritmo não vai até a borda do disco para mudar de direção, uma vez que ele muda de direção quando chega no último R daquela direção que havia chegado e o requisitado.

2. e 3. Bibliotecas pesquisadas e seus métodos utilizados

As bibliotecas pesquisadas foram: java.io.BufferedWriter, java.io.FileWriter. A utilização destas bibliotecas foram para, basicamente, ler e escrever nos arquivos de texto com os métodos: newLine(), write(), flush() e close().

4. Dificuldades experimentadas

No começo da estruturação, na etapa de leitura, remover os dados indesejados foi um desafio.

A escrita no arquivo também gerou certa dificuldade, devido ao fato de ter que seguir um padrão e, neste padrão, ter que truncar os valores decimais.

A grande dificuldade desse trabalho deu-se no início do desenvolvimento. Devido ao fato de que no começo obtivemos uma matriz com os valores de R, no início somente trabalhamos com esta matriz, ou seja, no desenvolvimento dos algoritmos manipulávamos a matriz. Com o passar do tempo, e com o feedback da entrega “Aquecimento”, percebemos que a manipulação da matriz seria muito mais complexa para conseguir executar os algoritmos. Em vez de manipular as matrizes, começamos a manipular uma arrayList de Rs, e, com isso, a execução dos algoritmos se tornou mais fácil e legível. Além disso, inicialmente tivemos dificuldades na evolução da ideia do MY, mas a ideia surgiu e foi possível desenvolvê-la.

5. Análises realizadas

Os testes e análises realizados consistiram, inicialmente, na execução à mão dos algoritmos e comparação com os resultados obtidos. Após a correção do trabalho no qual executamos os algoritmos em alguns arquivos de entrada, os resultados obtidos nesta correção foram fundamentais para determinar se o código estava gerando resultados corretos.