Trabalho Sistemas Operacionais - Simulador de Escalonamento por Prioridade

Por: Maicon Almeida Mian | RA: 2023.1.08.013

Universidade Federal de Alfenas | Professor: Fellipe Rey

1 OBJETIVO

O objetivo deste trabalho é implementar um simulador do Escalonamento por Prioridades, onde existem n filas de prioridade e o processo que será executado é o primeiro da prioridade mais alta (no caso de minha implementação, a mais alta é a de número maior). Também é necessário implementar o Round-Robin e um sistema para que um processo que não foi executado recentemente suba de prioridade para evitar o Starvation.

2 TECNOLOGIA UTILIZADA

Para o desenvolvimento do trabalho, a linguagem de programação escolhida foi JavaScript. Isso se deu porque a linguagem, acompanhada de HTML, consegue facilitar o desenvolvimento de um simulador visual, em que os níveis de prioridade, as entradas e as saídas são mostradas em tempo real para o usuário, sem precisar de bibliotecas externas, apenas a linguagem pura, além de ser simples de executar em qualquer máquina.

3 DECISÕES TOMADAS

Para a implementação do trabalho, algumas decisões deveriam ser tomadas. Assim, para começar, defini que algumas entradas no programa deveriam ser solicitadas ao usuário, sendo elas: o número de níveis de prioridade, indo de 1 a 10, a possibilidade de criação de processos aleatórios, e caso positivo, a porcentagem de probabilidade (entre 1 e 100) e o tamanho máximo que um processo aleatório pode ter, aleatorizando um valor entre 1 e o limite definido, também solicitei o tempo (em u.t) que dura um quantum do Round-Robin e o tempo máximo (em u.t) que um processo deve ficar sem executar para subir sua prioridade. Por fim, deixei livre a criação de processos pelo próprio usuário, tendo que colocar o tamanho (em u.t) do processo e sua prioridade.

Ademais, cada processo tem seu contador de tempo sem executar, e quando esse tempo chega ao limite definido pelo usuário, o processo sobe (caso não esteja na maior prioridade), dessa maneira, a cada u.t, é verificado, em todos os processos, se ele já atingiu

seu limite e se deve subir. Além disso, decidi que cada unidade de tempo (u.t.) equivaleria a 1,3 segundos na vida real, permitindo que o acompanhamento do processamento seja feito com atenção. O ID do processo é gerado de acordo com a quantidade de processos criados.

É importante ressaltar que se a opção de criação de processos aleatórios for ativada, o processamento continuará infinitamente até que o usuário clique no botão de parar.

4 IMPLEMENTAÇÃO

Como dito anteriormente, a implementação foi feita em Javascript, mesclando orientação a objetos (com as classes: Processo, Nível e Gestor de Níveis) e programação estruturada (demais funções), além do arquivo HTML para gerar a visualização ao usuário e css para personalizar a página.

5 SAÍDA

Na página criada, na parte superior, é solicitado ao usuário as informações, já na parte inferior, é a parte da saída dos dados. Na saída, temos 3 colunas, a primeira são os níveis de prioridade, que se divide em mais colunas, uma para cada nível, e em que o processo mais ou topo é primeiro da fila daquele nível e assim sucessivamente. Essa tabela de níveis é dinâmica e é atualizada a cada nova u.t (mostrando processos que mudaram de prioridade, processos gerados aleatoriamente e processos que foram executados).

A próxima coluna mostra o último processo executado na última u.t, e a linha do tempo mostra todos os processos executados naquela simulação. É possível observar que a linha do tempo começa com 1 *u.t*, isso porque a visualização é atualizada ao fim de cada unidade de tempo, por isso, quando ***P1 1.u*** aparece, isso significa que o P1 foi executado do tempo 0 até o 1, se ***P2 4 u.t*** aparece, é porque nada executou no meio e ele foi executado de 3 até 4 *u.t*. É importante ressaltar isso para demonstrar que, por exemplo, se um processo deve subir sem executar a 2 *u.t*, assim que 2. *u.t* aparece na linha do tempo ele já sobe, pois 2 unidades de tempo já se passaram.

6 EXECUÇÃO

Para executar o programa, basta iniciar o arquivo *index.html* no navegador de sua preferência, o código foi testado no Chrome e no Mozilla.