Jonathan Heidy Kinjo RGA:201519060360
Renan Benatti Dias RGA:201519060602

Sistemas Operacionais
Trabalho 2

Universidade Federal de Mato Grosso do Sul - UFMS
21 de junho de 2017

Índice

Introdução

O presente trabalho é sobre o escalonamento da CPU, mais precisamente aborda sobre a implementação do conceito básicos do escalonamento e utilização da sincronização de processos.

O tema abordado deve se ao fato de que o escalonamento da CPU é base dos sistemas operacionais multiprogramados, ou seja, a multiprogramação afeta consideravelmente a utilização da CPU, tornando o computador mais produtivo. Assim, ao considerar isso detêm a motivação deste trabalho, que está em base no aprendizado e a prática. Este trabalho tem como objetivo aplicar estes assuntos abordados na resolução do problema proposto pelo trabalho.

Desenvolvimento

A ideia de implementação dos itens:

A. A implementação possui 4 classes (“ThreadCPU”, “MyProcess”, “ThreadProcess” e “MyCpu”) e uma “Main”. Para a simulação de fila de prontos, utilizamos o PriorityQueue, uma lista pronta da biblioteca java, que se comporta com uma FIFO de prioridade, ou seja, esta fila organiza os processos por prioridade e em ordem de chegada;
B. Para a identificação da thread criamos uma classe denominada “MyProcess”, possuindo como um dos atributos o “id” (identificador);
C. Há um condição no início da execução do sistema que trata do problema da falta de argumentos;
D. Guardamos o valor de *inter* em uma variável da classe “ThreadProcess” que implementa Runnable, que ao executar o método run é calculado o tempo em que a thread entrará na fila, auxiliado com o método sleep da Thread;
E. Esta unidade é guardada na variável “timeUnit”;
F. O tempo está convertido tudo em múltiplo inteiro de unidade milissegundo;
G. Esse cálculo é feito no construtor da classe “MyProcess” e o valor é guardada no atributo “cpuTime”;
H. O estado da thread é guardado em uma variável “state” da classe “MyProcess”;
I. O valor do tempo que resta para o processamento está guardada na variável “cpuTime”;
J. Esta prioridade está guardada em uma variável “priority” da classe “MyProcess”;
K. Este requisito é assegurada pela fila PriorityQueue e pela implementação do Comparable com o método compareTo;
L. O time quantum é guardada em uma variável chamada “timeQuantum” da classe “ThreadCpu”;
M. Esta especificação é realizada no método run() da classe “ThreadCpu”, na condicional “if(cpu.prioNext() >= running.priority){..}”, na qual faz a comparação se a thread que está sendo executada tem menor ou igual prioridade que a thread de maior prioridade na lista de pronto. Caso sim, a thread que está executando muda para o estado pronto e é alocada para a fila de prontos e a thread comparada, da fila de prontos, é alocada para ser executada. Senão, não é feita nenhuma ação e continua a execução da thread.
N. Esta especificação é realizada no método run() da classe “ThreadCpu” realizada dentro do laço “while(thisTimeQuantum > 0) {... running.priority--;..}”;
O. Esta especificação é realizada no método run() da classe “ThreadCpu”, na condicional “if(cpu.prioNext() >= running.priority){..}”, na qual faz a comparação se a thread que está sendo executada tem menor ou igual prioridade que a thread de maior prioridade na lista de pronto. Caso sim, a thread que está executando muda para o estado pronto e é

alocada para a fila de prontos e a thread comparada, da fila de prontos, é alocada para ser executada. Senão, não é feita nenhuma ação e continua a execução da thread;

P. Para realizar esta especificação é utilizada a condição “if(running.restaTime <= 0){break;}” no método run() da classe “ThreadCpu”, na qual irá parar de executar o processamento da thread se o seu tempo de processamento chegar a 0 ou ficar menor que 0;
Q. Para executar essa primeira iteração foi criada denominada “cpuThread”.
R. Especificação assegurada pelo laço “while(i > 0){...}”, na qual a variável i é definida pela quantidade de threads e somente se é decrementada quando a thread esgotou seu tempo de processamento;
S. Foi utilizado a linguagem java para implementação das threads, e uso do semaphore para a escrita e retirada das threads na fila;
T. A hora do sistema é realizada pelo new Date() juntamente com o System.out.println(..) com as outras informações.

Conclusão

Este trabalho foi muito importante para a nossa compreensão deste tema, pois visto que permitiu-nos a conhecer melhor sobre algumas peculiaridade de uma implementação do escalonamento da CPU, além de ter-nos permitido desenvolver competências de investigação, seleção, organização e pesquisa para a resolução de problema proposto pelo trabalho.