Sistemas Operacionais
Trabalho I - Escalonador

Ramon Corrêa Fernandes

[1]Faculdade de Informática – Pontifícia Universidade Católica(PUCRS)
Porto Alegre / RS – Brasil
15 de Maio de 2018

Resumo. *Este artigo descreve uma solução para o problema apresentado no primeiro trabalho da disciplina de Sistemas Operacionais, dada uma sequência de processo executados com prioridade e tempo que precisam ser executados, o algoritmo deve solucionar quanto tempo cada processo executará e em qual processo estará executando em qual fatia de tempo, levando em consideração os valores dados além de troca de contexto e tempo de entrada e saída.*

1. Introdução

O problema apresentado mostra uma sequência de x processos e tamanho da fatia de tempo (ciclo), seguindo dos x processos com informações de tempo de chegada, tempo que o processo ficará em execução e prioridade de cada processo, dadas estas informações o programa deve executar os processos mostrando o tempo em que estes serão executados, média de tempo de espera e os tempos médios de resposta e de espera dos processos citados.

2. Leitura do Arquivo

O arquivo de teste apresentado possui duas linhas fixas no topo sendo a primeira o número de processos que serão executados e a segunda e o tamanho da fatia de tempo de execução de um processo antes da troca de contexto obrigatória, ambos os valores são guardados nas variáveis *nProcessos* e *fatiaTempo* e serão utilizadas na sequência para o resto do algoritmo.

As linhas em sequncia apresentam os processos cada linha com três valores, o tempo de chegada, tempo de execução, prioridade, estes valores são armazenados num objeto *Processo* que possui os atributos *id, tempoChegada, tempoExecucao, prioridade, duracaoRestante*, sendo que id é auto-incremental e define uma ordem númerica sequencial para cada novo processo criado, tempo de chegada, execução e prioridade são recebidos diretamente do arquivo e duração restante inicialmente recebe um valor igual ao do tempo de execução do processo e terá um papel crucial no desenvolvimento do algoritmo.

Os processos são armazenados em uma estrutura de fila encadeada *listaAguardando* e ordenado por tempo de chegada desta forma é mais fácil verificar qual próximo processo será executado ou estará pronto para ser executado

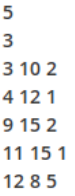

Figura 1. Exemplo de Arquivo Teste

3. Algoritmo

3.1. Prioridades

As prioridades são definidas de 1 à 9 sendo 1 o maior privilégio e 9 o menor, deve ser escolhido um critério de desempate para caso de processos com a mesma prioridade. Neste algoritmo foi escolhido como critério de desempate o tempo que falta para a execução do processo, ou seja, dado dois processos 1 e 4 com prioridade 1, ambos possem tempo de execução 12, porém o processo 1 ja executou por 2 medidas de tempo, faltando apenas 10 para terminar, este processo terá preferência na execuçãoa sobre o processo de mesma prioridade com 12 medidas de tempo ainda por executar.

3.2. Execução

Dados as estruturas devidamente construídas o algoritmo começa a executar, uma variável *tempo* é incrementada a cada começo de ciclo para simular tempo de execução do programa, esta inicia em zero, uma outra variável *proximoProcesso* recebe e remove o primeiro valor de listaAguardando, a cada novo ciclo é verificado se o tempo atual é maior ou igual ao tempoChegada do processo armazenado em próximoProcesso, caso seja, este processo vai para uma segunda lista encadeada de nome *listaExecução* onde são armazenados os processos prontos para ser executados, quando o proximoProcesso vai para nova lista ele ja recebe o próximo valor da listaAguardando e já é verificado novamente se o tempoChegada deste novo processo é menor ou igual ao tempo atual, para evitar que processos de mesmo tempo sejam esquecidos na lista e entrem em momentos errados. Uma vez que existe um processo em listaExecução o programa está pronto para executa-lo, desta forma a listaExecução é ordenada por prioridade e em caso de empate o desempate é feito por meio do tempoDuracao, essa ordenação é realizada apenas durante a **troca de contexto** ou quando a listaExecução vazia recebe um novo processo. Após a ordenação da lista o processo que estiver na posição zero da lista, isso é o com maior prioridade para execução, tem seu tempoDuração decrementado em um, para simular o tempo de execução, quando este chega em zero o processo é removido da lista indicando que o processo terminou sua execução.

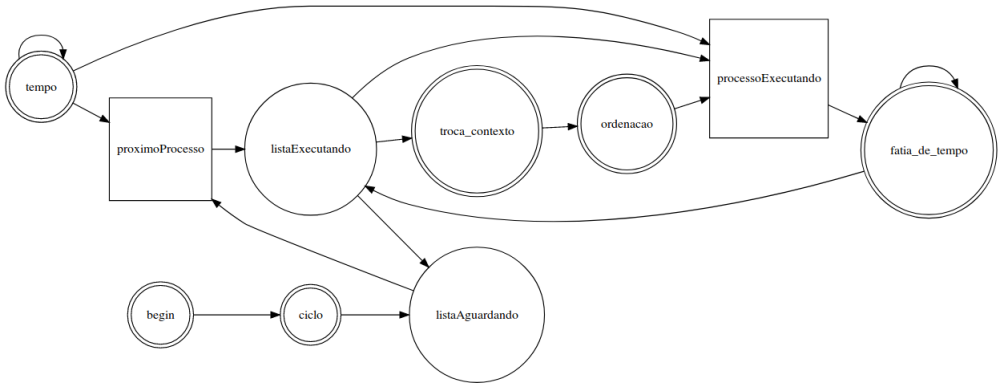

Figura 2. Ciclo de Execução do Algoritmo

3.3. Troca de Contexto

A troca de contexto é o processo computacional que armazenar e restaurar o estado de uma CPU de forma que múltiplos processos possam compartilhar uma única instância de CPU. Neste algoritmo ela serve para marcar o momento em que é realizada a troca de um processo para o outro, seja porque o processo terminou sua execução ou porque um ciclo (fatia de tempo) terminou e a listaExecução, agora reordenada, decidiu que o processo atual não é mais prioridade e agora outro é mais importate, a troca de contexto foi definida para a simulação que tem tempo x, dado pela segunda linha do arquivo armazenada na variável fatiaTempo e uma segunda variável ciclo é incrementada junto com o tempo e quando ele alcança a fatiaTempo ocorre uma troca de contexto, nenhum processo é executado durante este ciclo, a lista é reordenada e a variável ciclo é zerada.