2º Trabalho de Computação Concorrente

Aluno: Mauro Victor de Araújo Nascimento
DRE: 118090256

Para o trabalho, deve ser implementado um programa para o problema dos Leitores e Escritores. Entre os tópicos necessários estão:

- Mais de um leitor pode ler ao mesmo tempo
- Apenas um escritor pode escrever de cada vez
- Não é permitido ler e escrever ao mesmo tempo
- Evitar inanição (situação onde um processo não consegue ser executado de forma alguma pois sempre existem outros com maior prioridade).

Algoritmo

Entrada: Quantidade de Threads leitoras, quantidade de threads escritoras, número de leituras, número de escritas e nome do arquivo de log.

Saída: Arquivos com escritas geradas pelas threads leitoras.

1. Recebe argumentos de entrada.
2. As variáveis “qtdLeitores” e “qtdEscritores” recebem as respectivas quantidades dos dois tipos de threads, leitores e escritores.
3. A variável “Limite” recebe a quantidade de “escritas” que devem ser feitas.
4. Inicia as Threads Escritoras.
5. Inicia as Threads Leitoras.
6. Cada Thread Escritora recebe seu identificador, guardado na variável “tid”.

a. Enquanto o número de escritas não for alcançado, ela repete os seguintes passos:
    i. Entra na função “EntraEscrita()”, passando seu identificador como parâmetro.
    ii. Dá Lock para evitar o uso indevido das variáveis seguintes.
    iii. Verifica se existe alguém escrevendo ou lendo. Se existir, aguarda. Senão, prossegue.
    iv. Printa uma mensagem avisando que está escrevendo.
    v. Avisa logicamente que irá escrever, aumentando o valor da variável “escritores” em uma unidade.
    vi. Dá Unlock e sai da função “EntraEscrita()”.
    vii. Dá Lock novamente para alterar a variável “papel”, que obterá o valor do identificador da thread (mais tarde, um leitor lerá esse identificador).
    viii. Dá Unlock
    ix. Entra na função “SaiEscrita()”.
    x. Dá Lock
    xi. Avisa logicamente que deixará de escrever, decrementando a variável “escritores” em uma unidade.
    xii. Dá sinal com a variável de condição para que os leitores possam ler.
    xiii. Dá sinal para que o próximo escritor possa escrever.
    xiv. Dá Unlock
b. Thread termina execução.

7. Cada Thread Leitora recebe seu identificador, guardado na variável “tid”.
    a. Inicia a variável “vouEscreverNoArquivo”, que guardará o valor lido pelo Leitor, para que seja escrito em seu arquivo.
    b. Enquanto o número de escritas+1 não for alcançado (visto que após a última escrita, os leitores ainda terão algo para ler), ela repete os seguintes passos:

i. Entra na função “EntraLeitura”, enviando seu identificador e aguardando o retorno da mesma, que será guardado na variável “vouEscreverNoArquivo”.
ii. Dá Lock
iii. Verifica se a variável “escritores” é maior do que zero. Se for, significa que existe alguém escrevendo, e portanto, a thread aguarda. Quando for zero, prossegue.
iv. Printa uma mensagem avisando que está lendo.
v. Incrementa o valor da variável “leitores”, avisando que está lendo.
vi. A variável “vouEscreverNoArquivo” recebe o valor lido na variável “papel”.
vii. Dá Unlock e a função EntraEscrita() retorna “vouEscreverNoArquivo”.
viii. Leitor verifica o valor de seu identificador e cria um arquivo para si mesmo.
ix. Escreve o valor de “vouEscreverNoArquivo” no arquivo.
x. Executa o método SaiLeitura()
xi. Dá Lock
xii. Decrementa o valor de “leitores”
xiii. Se “leitores” tiver valor igual a zero, libera para o próximo escritor
xiv. Dá Unlock

c. Thread termina execução.