Trabalho Prático
Fundamentos de Sistemas Paralelos e Distribuídos

João Antonio Oliveira Pedrosa
Matrícula: 2019006752

[1] Universidade Federal de Minas Gerais
Belo Horizonte - MG - Brasil

joao.pedrosa@dcc.ufmg.br

1. Introdução

O objetivo do trabalho é implementar um sistema de sincronização entre threads que se assemelha a um sistema de controle de movimentação para um jogo digital, ou para o controle de robôs.

2. Estruturas e Funções

Para a xplicação das decisões de implementação, será explicado o objetivo de cada uma dos estruturas e funções definidos no código.

2.1. Estruturas

2.1.1. thread_id, group e path_s

Arrays simples que guardam, na posição i, qual o ID, o grupo e o tamanho do caminho percorrido da thread i.

2.1.2. path

Matriz que guarda, na posição (i, j), três inteiros, representando o X, o Y e o tempo de espera para o ponto j no caminho da thread i.

2.1.3. lock e cond

Matrizes de dois tipos, a primeira do tipo *pthread_mutex_t* e a segunda do tipo *pthread_cond_t*. Armazenam a condição de espera e o mutex de cada posição do tabuleiro. Esses valores são utilizados para implementar a exclusão mútua, garantindo que posições iguais não são acessadas ao mesmo tempo e para a emissão do sinal para checagem das posições.

2.1.4. occupied

Matriz bidimensional com 2 inteiros em cada posição. Guarda, para cada posição do tabuleiro, qual o grupo que está ocupando aquela posição e quantos indivíduos do grupo estão na posição atualmente. O grupo -1 significa que nenhum grupo está ocupando a posição atualmente.

2.2. Funções

2.2.1. passa_tempo

Função definida na especificação do trabalho.

2.2.2. enter

Recebe um X, um Y e o grupo da thread que chamou a função. Se occupied na posição (X, Y) pertencer ao mesmo grupo, adiciona 1 no número de indivíduos do grupo na posição. Caso esteja vazio, atribui o grupo correto ao inteiro que sinaliza o grupo e atribui 1 ao número de indivíduos na posição. Caso esteja ocupada por alguém de grupo diferente, espera enquanto não receber um sinal da posição (X, Y). Ao receber um sinal, testa novamente.

2.2.3. exits

Recebe uma posição (X, Y). Se houver apenas um indivíduo ocupando a posição, atribui -1 como o grupo daquela posição. Se houver mais de um indivíduo, diminui o número de indivíduos na posição. Em ambos os casos emite um sinal para a posição (X, Y).

2.2.4. walk

A função que executa cada thread. Recebe um inteiro indicando o indicador daquela thread. Para cada posição do caminho definido para esta thread, chama a função *enters* para a posição, chama a função *exits* para a posição anterior e chama a função *passa_tempo* com o tempo correto.

2.2.5. main

Realiza a leitura da entrada e a criação de cada thread.