Brasília 13 de Novembro de 2018

Universidade de Brasília
Departamento de Ciência da Computação

Trabalho Prático

Autor
Cristiano Cardoso - 15/0058349

Disciplina
Programação Concorrente - Turma A
Prof Dr. Eduardo Alchieri

Introdução

Este trabalho visa apresentar uma abordagem de se aprender os conceitos de programação concorrente utilizando elementos visuais. O objetivo é criar um ambiente 2D onde o aluno possa manipular objetos na tela a fim de verificar o comportamento de seu algoritmo *multithread* em tempo real.

Formalização do Problema

Implementar utilizando a biblioteca *pthreads* exemplos que auxiliam o programador a verificar a corretude do algoritmo.

Descrição do Algoritmo

O algoritmo foi retirado de um exercício dado em sala de aula intitulado “Estudo Dirigido 3”. Se tratava do problema dos macacos, dado a seguir:
Existe uma ponte e uma quantidade X de macacos no lado esquerdo e Y, no direito. O programador deve implementar um algoritmo utilizando locks de forma que não haja macacos passando dos dois lados. Isto é, existe um turno para o uso da ponte

Nesta etapa do desenvolvimento, obtemos a seguinte solução implementada em C:

```
// realiza o lock de um lado da ponte
pthread_mutex_t global_bridge_access_lock[2];
// lock para calibramento das variáveis
pthread_mutex_t global_va_state_lock;
void * bridge_manager (void *arg){
      int monkey_id = *(int*)arg;
      // diz o lado em que o macaco está E ou D
      uint side = monkey_state[monkey_id];
      pthread_mutex_lock(&global_va_state_lock);
      // aumenta o número de macacos aguardando para passar
      monkey_count[side] += 1;
      pthread_mutex_unlock(&global_va_state_lock);
```

```
// caso o primeiro macaco passar
if (monkey_count[side] == 1){
      pthread_mutex_lock(&global_bridge_access_lock[side]);
// se eu posso ter acesso exclusivo a ponte
      pthread_mutex_lock(&global_bridge_access_lock[1-side]);
// então eu bloqueio o acesso para os macacos do inverso
printf(open_text, state_text[side], state_text[1-side]);
      pthread_mutex_unlock(&global_bridge_access_lock[side]);
// somente o “meu” lado poderá passar
}

      pthread_mutex_lock(&global_bridge_access_lock[side]);
      monkey_count[side] += -1;
      monkey_transfer_count[side] += 1;
      monkey_state[monkey_id] = 1-side; // neste momento, ao
chegar no outro lado, o macaco muda de lado
      if (monkey_count[side] == 0){
      monkey_transfer_count[side] = 0;
      // libera o acesso ao lado inverso
      pthread_mutex_unlock(&global_bridge_access_lock[1-side]);
      }
      pthread_mutex_unlock(&global_bridge_access_lock[side]);

      pthread_exit(0);
}
```

As modificações neste algoritmo estão na adição de rotinas para desenho dos elementos da tela. Também foi adicionada uma variável de condição a fim de aguardar os elementos chegarem ao outro lado da tela. E só assim o acesso ao outro lado da ponte é liberado.

Conclusão

Neste projeto foi possível observar os casos em que certas implementações não estavam corretas. Isto é, era possível ver objetos cruzando na tela. E outras onde havia o controle de acesso exclusivo da ponte.