MC514 – Sistemas Operacionais

1s2009 – Laboratório 3

Rollercoaster

Anderson Philip Birocchi *072787*

David Burth Kurka *070589*

Felipe Elterman Braga *070803*

Neste laboratório escolhemos o algoritmo da montanha-russa para implementar utilizando várias threads controladas por semáforos e mutexes. Resumidamente o projeto implementa uma animação em ASCII do gerenciamento de uma montanha-russa que segue os seguintes requisitos:

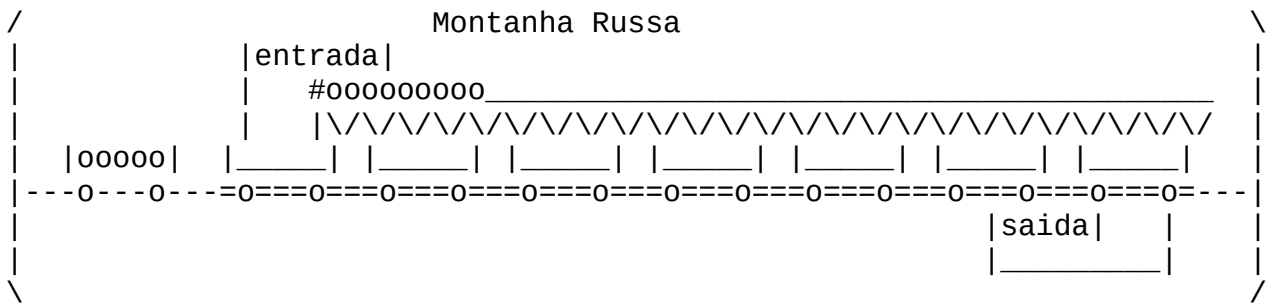
1- REQUISITOS DA MONTANHA RUSSA

- Apenas um carro carrega por vez;
- Os passageiros só embarcam quando o carro estiver carregando;
- O carro só sai quando estiver cheio;
- Vários carros podem passear ao mesmo tempo;
- Os carros chegam na mesma ordem na qual sairam;
- Só um carro pode descarregar por vez;
- Os passageiros só desembarcam quando o carro estiver descarregando.

Respeitando estes requisitos, faremos com que cada carro e cada passageiro seja controlado por uma thread, para assim termos maior paralelismo e consequentemente um aumento no realismo do algoritmo. Abaixo segue um exemplo de como é o display da animação:

2- ANIMAÇÃO

Esquema de 8 carros com capacidade para 5 pessoas:



Em cada figura, são mostrados os carros que estão aguardando, a fila de pessoas à espera por embarcar e a fila de pessoas que vão desembarcando. Quando um ou mais carros estão passeando, estes não são mostrados.

A animação tem tempos aleatórios e pode confundir. Pra ver tudo, é recomendável salvar a saída em um arquivo e ir percorrendo figura por figura.

Além disso, limitamos o algoritmo para que ele termine em algum tempo finito, ou seja, não gera indefinidamente os passageiros até segunda ordem, portanto, pelo requisito do algoritmo de um carro só sair se estiver cheio, caso o número de passageiros não seja um múltiplo da capacidade do carro, o programa irá ficar parado esperando e nada mais acontece.

3- IMPLEMENTAÇÃO

Como já foi dito, utilizamos neste laboratório uma implementação com semáforos.
Cada carro possui seu próprio semáforo: um na área de embarque e outro na de desembarque. Esses semáforos foram armazenados em 2 vetores, um para embarque e outro para desembarque.
Há também um semáforo na entrada e na saída da montanha russa, controlando o acesso dos passageiros ao brinquedo e mais 2 para saber quando um carro está cheio ou vazio, permitindo assim sua movimentação. Para garantir o embarque e desembarque de apenas um passageiro por vez, utilizamos mutexes. Mais detalhes podem ser encontrados nos comentários do código.

4- CONSIDERAÇÕES FINAIS

Este laboratório foi importante para mostrar como um algoritmo de multi-threads é interessante para simular um paralelismo pseudo-real (ou real em caso de multi-processadores), o quanto é eficiente o controle através de semáforos e mutexes e também para ensinar um jeito mais dinâmico para representar uma saída de um programa ao invés de textos apenas.