OO – Engenharia Eletrônica

Programação em C/C++

Slides 19 – C : *Multithreading.*

Produtor /Consumidor

Exercícios.

Prof. Jean Marcelo SIMÃO – DAELN/UTFPR

Produtor - Consumidor

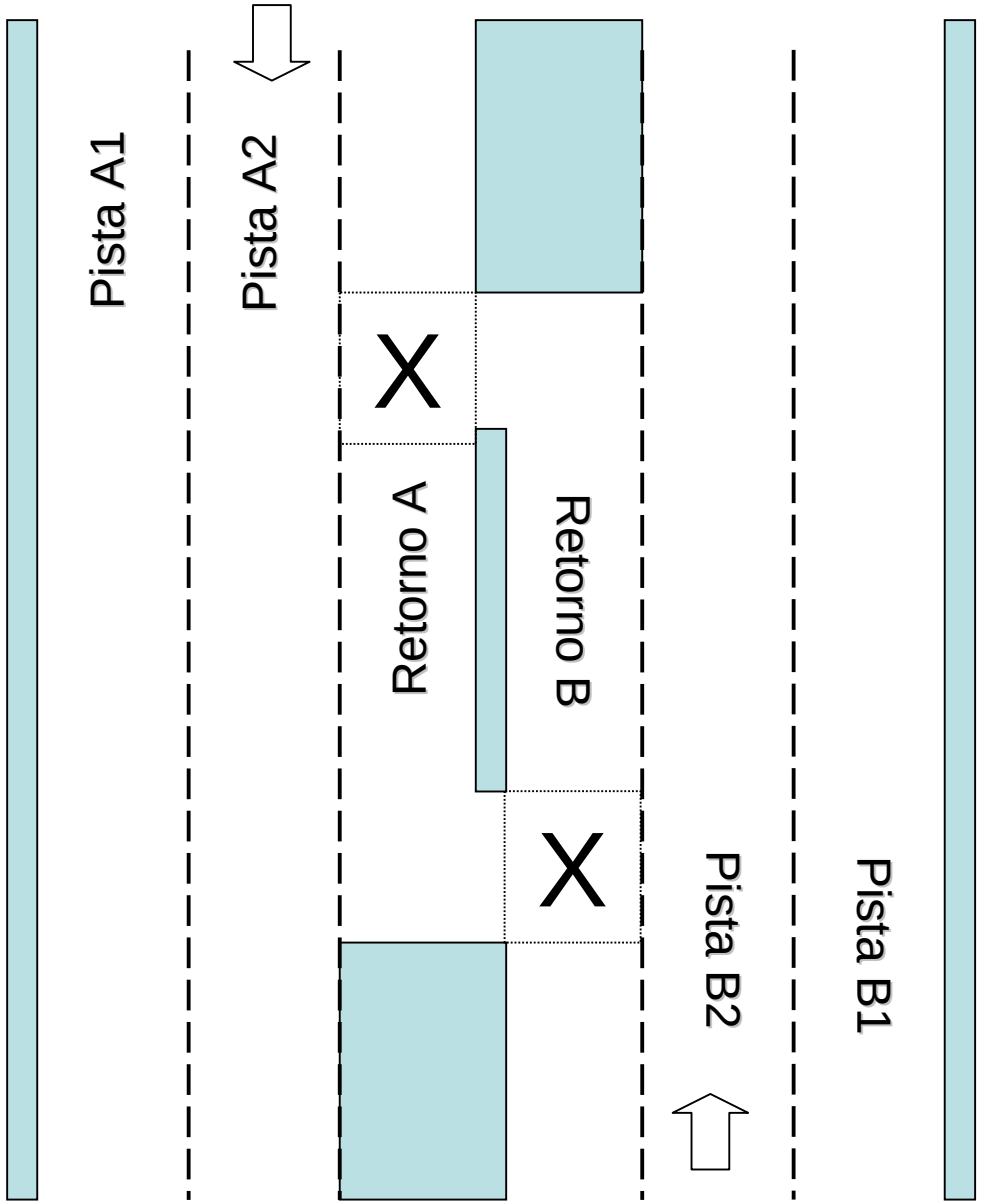
Exercícios

Pré-requisitos

- Ter compreendido e estudado as aulas anteriores, sobretudo as aulas de *threads.*
  - *Slides 16 e 17.*

- Eis outro material introdutório sobre *threads.*
  - http://200.17.137.110:8080/licomp/Members/rfidalgo/infrasw/aulas/tema6-introducao-a-thread.pdf
  - *http://200.17.137.110:8080/licomp/Members/anderson/infra-software/is-aula-07.pdf/download*

Considere o seguinte “mapa”.



Exercício 1

- Componha o cenário exposto em um programa C++ utilizando preferencialmente formulários (*form*) e componentes pré-prontos (e.g. componentes do *.net* ou da *STL*).

- Simule a circulação de carros nas Pistas A, de maneira tal que alguns carros tomem o Retorno A afim de irem no sentido contrário do qual vinham, i.e. Pistas B.

- Cada sentido deve ser gerenciado por uma (ou até mais) *threads*, preferencialmente no formato orientado a objetos.

  – Assim sendo, quando a *Thread A* envia carros para o Retorno A, esta se torna uma thread dita produtora. Similarmente, quando a *Thread* B absorve estes carros, ela se torna uma thread dita consumidora.

- Faça com que a fila de Retorno A lote.

- Permita carros na posição X do Retorno A.

Exercício 2

- Considere o Exercício 1.
- Simule a circulação de carros também nas Pistas B, de maneira tal que alguns carros tomem o Retorno B afim de irem no sentido contrário do qual vinham, i.e. Pistas A.
- Cada sentido deve ser gerenciado por uma (ou até mais) *threads*, preferencialmente no formato orientado a objetos.
  - Assim sendo, quando a *Thread B* envia carros para o Retorno B, esta se torna uma thread dita produtora do ponto de vista de A. Similarmente, quando a *Thread* A absorve estes carros, ela se torna uma thread dita consumidora do ponto de vista de B.
- Faça com que as filas de Retorno A e B lotem.
- Permita carros na posição X do Retorno A e do Retorno B.
- Quanto houver carro em posição X, carros não poderão passar por ali, bloqueando o retorno.
- Observe que um bloqueio total ocorrerá, o dito "*deadlock*".

Exercício 3

- Considere o Exercício 2.
- Exercício 3 A:
  - Impeça a utilização de cada posição X usando um *mutex* para cada qual.
- Exercício 3 B:
  - Impeça a utilização de cada posição X usando um semáforo para cada qual, sendo o limite do semáforo a quantidade de carros comportadas por cada retorno antes da respectiva posição X.
- Exercício 3 C:
  - Impeça a utilização de cada posição X relativa a uma *thread* a partir do momento em que a outra *thread* notificá-la que há carros no retorno de sua responsabilidade.
  - Permita a utilização de cada posição X relativa a uma *thread* a partir do momento em que a outra *thread* notificá-la que não há carros no retorno de sua responsabilidade.
- Obs.:
  - A notificação de *threads* dar-se-á por meio de mensagens trocadas entre elas.
  - Estude o assunto de troca de mensagens entre *threads*.
  - Vide próximo *slide* (transparência).

Dicas de sítios sobre *Thread Message*

- http://www.sixtyfourbit.org/mq4cpp.htm
- http://www.codeproject.com/KB/cpp/Win32_MQ_MultiThreading.aspx
- http://www.wehlou.com/Code/msgthreads/index.htm