MAC 438 - Programação Concorrente - Primeiro Semestre de 2006

Segundo Exercício-Programa: *POSIX threads* e semáforos

Entrega: até 10 de maio de 2006

Este exercício deve ser desenvolvido em equipes de duas pessoas, a fim de suscitar discussão. Mesmo sem ser grande, ele envolve detalhes que devem ser conhecidos por todos vocês. Não deixe seu colega de equipe fazer tudo sozinho!

Dúvidas sobre o enunciado devem ser enviadas para a lista de discussão de MAC-438.

1 O Problema da Montanha Russa

Suponha que existam n passageiros e um carro em uma montanha russa. Os passageiros, repetidamente, esperam para dar uma volta no carro. O carro tem capacidade para C passageiros, com $C < n$. O carro só pode partir quando estiver cheio. Após dar uma volta na montanha russa, cada passageiro passeia pelo parque de diversões e depois retorna à montanha russa para a próxima volta.

Tanto o carro como os passageiros devem ser representados por *threads*. As *threads* passageiro executam o seguinte pseudo-código:

```
thread passageiro {
    while (!fechouParque) {
        entraNoCarro();
        esperaVoltaAcabar();
        saiDoCarro();
        passeiaPeloParque(); // tempo variável
    }
}
```

A *thread* carro executa o seguinte pseudo-código:

```
process carro {
    while (existemPassageirosNoParque) {
        esperaEncher();
        daUmaVolta();
        esperaEsvaziar();
        volta++; // serve como parâmetro para fechar o parque
    }
}
```

2 Requisitos

- Este EP deve ser implementado no Linux, usando semáforos (funções `sem_init`, `sem_wait` e `sem_post`) e a biblioteca Pthreads (POSIX threads). Como o objetivo é usar semáforos, não devem ser usadas as funções da biblioteca Pthreads que dão suporte a sinalização/espera por condições (funções `pthread_cond*`).

- O principal no programa é o uso de semáforos para controle de concorrência e sincronização entre threads. A ordem de chegada dos passageiros na fila de embarque na montanha russa deve ser respeitada. Sua solução não deve deixar que passageiros furem essa fila.

- O carro só pode partir se estiver cheio.

- A saída do seu programa deve ser bem planejada, de modo a mostrar o que está acontecendo a cada momento, sem ficar carregada demais.

- **Bônus:** Faça seu programa lidar com o caso de m threads carro, onde $m > 1$. Como a montanha russa tem só um par de trilhos, nenhum carro pode ultrapassar outro. Em outras palavras, os carros devem terminar cada volta na montanha russa na ordem em que eles começaram essa volta. Como anteriormente, um carro só pode partir se estiver cheio. (Esta parte é opcional.)

3 Sobre a entrega

Você deverá entregar um arquivo tar.gz contendo arquivos-fonte, `makefile`, arquivo `README`, ... O desempacotamento do seu arquivo tar.gz deverá produzir um diretório contendo esses ítens. O diretório deve ter nome da forma `ep2-membros-da-equipe` (exemplo: `ep2-joao-maria`).

A entrega será via Internet. Detalhes adicionais serão divulgados na lista de discussão da disciplina.

EPs atrasados não serão aceitos.

Bom trabalho!