Projeto 2 - Animação Multithread com Semáforos

Cristiano (256352), George (216741),
Jhonatan (256444) e Mylena (222687)

TABELA DE CONTEÚDOS

01

Tema do Projeto

THE ROLLER COASTER PROBLEM

Suponha que existem n threads de passageiros e uma thread de carro. Os passageiros esperam repetidamente para andar no carro, que comporta C passageiros, onde C < n. O carro só pode dar a volta nos trilhos quando estiver cheio. Todos os passageiros devem andar pelo menos uma vez no carro.

**** Versão modificada do problema homônimo contido em "The Little Book of Semaphores", de Allen B. Downey.*

THE ROLLER COASTER PROBLEM

Detalhes adicionais:

- Os passageiros devem invocar **board** e **unboard**;
- O carro deve invocar **load**, **run** e **unload**;
- Os passageiros não podem embarcar sem **tickets** e sem que o carro tenha invocado **load**;
- O carro não pode partir até que C passageiros tenham embarcado;
- Os passageiros não podem desembarcar sem que o carro tenha invocado **unload**.

02

Solução Proposta

PONTOS PRINCIPAIS

- Uso de multithread, semáforo, mutex lock e variável de condição;
- Base construída conforme dicas de solução do problema original;
- Uso de **tickets** para organizar as corridas e evitar starvation de algumas threads de passageiros.
  - Cada passageiro começa com um ticket;
  - Caso haja assentos livres para a última corrida, os passageiros sem ticket competem para conseguir um ticket grátis e andar novamente no carro.

03

Detalhes da Implementação

STRUCT PASSENGER_DATA

```
typedef unsigned int ui;

typedef struct passenger_data
{
    ui pid;
    ui tickets;
} p_data;
```

FUNÇÃO CAR

```
void* car(void *arg)
{
    ui rides = (n % C) == 0 ? n/C : n/C + 1;
    ui rides_count = 0;

    char log[100];
    memset(log, 0, sizeof(log));

    if (rides == 1)
        last_ride = 1;

    while (rides--)
    {
        // load();
        update_ride_log(++rides_count);
        sprintf(log, "O carro está pronto o embarque.\n");
        update_log_message(log);

        for (ui i = 0; i < C; i++)
            sem_post(&boardQueue);

        sem_wait(&allAboard);

        // run();
        sprintf(log, "O carro está se movimentando.\n");
        update_log_message(log);
        move_car_scene();
```

FUNÇÃO CAR

```
        // unload();
        sprintf(log, "O carro está pronto o desembarque.\n");
        update_log_message(log);

        for (ui i = 0; i < C; i++)
            sem_post(&unboardQueue);

        // Avisa aos passageiros sobre a última corrida
        if (rides == 1)
        {
            pthread_mutex_lock(&lr_mutex);
            sprintf(log, "O carro está pronto para a última viagem.\n");
            update_log_message(log);
            pthread_cond_broadcast(&lr_cond);
            last_ride = 1;
            pthread_mutex_unlock(&lr_mutex);
        }

        sem_wait(&allAshore);
    }

    return NULL;
}
```

FUNÇÃO PASSENGER

```c
void* passenger(void *p)
{
    p_data p_info = *((p_data*) p);
    ui id = p_info.pid;
    ui tickets = p_info.tickets;

    char log[100];
    memset(log, 0, sizeof(log));

    sleep(rand() % 5);  // Delay na chegada do passageiro
    arrival_scene(id);

    while (tickets--)
    {
        sem_wait(&boardQueue);

        // board();
        sem_wait(&mutex1);

        boarders++;
        sprintf(log, "O passageiro %d embarcou.\n", id);
        update_log_message(log);
        boarding_scene();

        if (boarders == C)
        {
            sem_post(&allAboard);
            boarders = 0;
        }
```

FUNÇÃO PASSENGER

```
    sem_post(&mutex1);

    sem_wait(&unboardQueue);

    // unboard();
    sem_wait(&mutex2);
    unboarders++;
    sprintf(log, "O passageiro %d desembarcou.\n", id);
    update_log_message(log);
    unboarding_scene();

    if (unboarders == C)
    {
        sem_post(&allAshore);
        unboarders = 0;
    }

    sem_post(&mutex2);

    pthread_mutex_lock(&lr_mutex);

    if (!last_ride)
        pthread_cond_wait(&lr_cond, &lr_mutex);
```

FUNÇÃO PASSENGER

```
        if (free_tickets)
        {
            free_tickets--;
            tickets++;
            sprintf(log, "O passageiro %d conseguiu um novo ticket para embarcar.\n", id);
            update_log_message(log);
            new_boarding_scene();
        }

        pthread_mutex_unlock(&lr_mutex);
    }

    sprintf(log, "%d terminou a viagem\n", id);
    update_log_message(log);

    return NULL;
}
```

ANIMAÇÃO


PPPPP
DESEMBARQUE
[0]
[0]
[+]
[+]
[+]
\#/
EMBARQUE
Dados:
n: 05
C: 05
Corrida: 01
LOG: 4 terminou a viagem

04

Demonstração

CASOS DE TESTE

- n < C;
- n = C;
- n > C.

*** *n: número de passageiros*
*** *C: número de assentos no carro*

OBRIGADO!

Alguma pergunta?