Mestrado Integrado em Engenharia Informática e Computação

Sistemas Operativos
14/05/2018

Simulação de um Sistema de Reserva de Lugares

Trabalho realizado pelo Grupo 07 da turma 2MIEIC01, constituído por:

Fernando Coelho	up201605270	up201605270@fe.up.pt
Henrique Gonçalves	up201608320	up201608320@fe.up.pt
João Carlos Maduro	up201605219	up201605219@fe.up.pt

Implementação

Para a realização deste trabalho baseamo-nos na figura (“Arquitetura geral da aplicação a desenvolver”) fornecida pelos docentes nas instruções do trabalho.

Comunicação entre processos:

Em termos de estrutura de troca de mensagens entre o cliente e servidor foi usada uma struct ‘request’ com a informação do programa *client* que era transmitida ao server através de um FIFO (para todos os clientes) chamado ‘requests’.

Em termos da troca de mensagens entre o servidor e o cliente, de forma a responder à informação que entrava a partir do fifo referido anteriormente, cada processo *client* cria o seu fifo ‘ansXXXXX’ (sendo o XXXXX o PID do processo *client*) a partir do qual o servidor (uma das threads “bilheteira” mais especificamente) vai responder ao request de cada cliente individualmente. Essa resposta consiste numa sequência, cujo primeiro elemento representa o número total de lugares reservados caso seja positivo, e os restantes números são os identificadores dos lugares reservados. No caso do primeiro número ser negativo, quer dizer que não foi possível realizar o pedido feito ao servidor e o seu valor representa o tipo de erro ou o motivo de o pedido não ter sido realizado.

Struct ‘request’:

```
typedef struct{
        int client_id;  //pid do processo cliente
        int num_wanted_seats; // nr. de lugares pretendidos num request
        int prefered_seats[MAX_CLI_SEATS]; //lista dos id’s dos lugares preferidos
        int array_size; //tamanho da lista anterior
        int error; //flag que indica se o user inseriu valores inválidos

} Request;
```

Mecanismos de sincronização:

Os mecanismos de sincronização utilizados são os semáforos da library <*semaphore.h*>*.* Estes semáforos são usados no acesso ao pedido de reserva no buffer unitário, e cada 'seat' (que é uma struct) dos lugares disponíveis do evento tem o seu próprio semáforo, sendo que quando um 'seat' está a ser processado, os outros threads que o tentem processar ficarão à espera que o que chegou primeiro acabe.

A nossa abordagem inicial consistia em usar um semáforo para cada lugar, e dessa forma várias threads conseguiriam trabalhar ao mesmo tempo em "seats" separados e tornar o servidor mais rápido. No entanto, após várias tentativas e estratégias diferentes não fomos capazes de implementar essa abordagem, uma vez que que se demonstrou demasiado complexo no caso de haver pedidos que se sobrepunham. Por esse motivo, fomos forçados a usar uma abordagem mais simples que existe um semáforo para acesso ao array dos lugares.

Encerramento do servidor:

No início do servidor é inicializado um parâmetro 'start_time' do tempo atual(time_t) que vai ser utilizado numa função que define quando o servidor acaba. Essa função, que vai servir para ler do FIFO os 'requests' dos clientes, vai ser chamada infinitamente num while loop até ter passado 'open_time' segundos (passado como parâmetro no programa *servidor*) desde o 'start_time', rompendo o loop e obrigando os threads das bilheteiras a acabar o que estavam a fazer (as bilheteiras terminam o pedido que estão atualmente a processar antes de terminar). Assim no fim, o servidor espera que todos os threads terminem, através de um thread_join().