Relatório de SOPE (Sistemas Operativos)

Projeto Final: Reserva de Lugares

TURMA: 2MIEIC02

Luís Mendes up201605769@fe.up.pt

Patrícia Janeiro up201605946@fe.up.pt

José Martins up201504761@fe.up.pt

Mecanismos de sincronização:

Para este trabalho recorremos essencialmente a mutexes para fazermos a sincronização, como tal, usamo-los para trancar as threads no início do programa (mal são criadas), garantindo que só começarão a tentar fazer reservas depois de se ter enviado o primeiro pedido.

(ver: *pthread_mutex_lock(&threads_lock)*)

São também utilizados quando se escreve para um ficheiro, certificando que as threads não interrompem a escrita uma da outra.

(*pthread_mutex_lock(&writing_lock)*)

Outro uso seria para manter as threads à espera da altura de terminar (assim que se chega a open_time segundos no servidor).

(*pthread_mutex_lock(&end_ticketer_lock)*)

Mais um detalhe era nas funções atómicas de reservar, freeSeat, bookSeat, isSeatFree, são utilizados para isolar dentro de uma secção critica a macro DELAY que simula um certo atraso nas operações de escrita.

(*pthread_mutex_lock(&delay_lock)*)

E, finalmente, também são utilizados para trancar todo o array de lugares (seats) cada vez que se faz uma operação de tentativa de reserva, impedindo as threads de se intrometerem num processo de reserva onde não deviam de interferir.

(*pthread_mutex_lock(&seats_lock)*)

Quanto ao main, este realiza uma espera ativa ate que uma das threads retire o pedido que colocou no buffer partilhado. Ao acontecer isto, a função main volta a colocar lá outro pedido que, por sua vez, vem do FIFO.

Enceramento do servidor:

O servidor encerra ao fim de open_time segundos, parâmetro especificado na linha de comando.

Para tal, utiliza funcionalidades da biblioteca <ctime>, contando durante esse intervalo de tempo dentro de um ciclo while.

Nesse ciclo while o main tenta constantemente ler novos pedidos do FIFO, passando-os para um buffer que ser lido pelas threads.

Assim que se chega a open_time segundos, sai-se do ciclo while e o main informa as threads que está na altura de terminar, utilizando-se pthread_join para esperar por elas. Fecha-se e destrói-se o FIFO, terminando o programa.

Estruturas:

Seat: Estrutura usada para simular um lugar, que pode ou não estar livre, assinalado pela flag *occupied*, se estiver a 'y', está ocupado, se estiver a 'n', não está. Contem também o número do lugar e o identificador do cliente que o ocupa.

Nota: O array seats utilizado no programa e um array deste tipo de estruturas.

Answer: Estrutura que é utilizada para enviar informação desde o servidor para o cliente, contem uma flag de erro (que será um número qualquer negativo caso não se possa ter efetuado uma reserva por um dado motivo) chamada error_flag e um array reservedSeats, em que o primeiro número representa a quantidade de lugares reservados, se a reserva teve sucesso, e os restantes são os identificadores desses lugares.

Request: Estrutura utilizada para enviar uma mensagem do cliente para o servidor, simboliza um pedido. Contem o identificador do cliente, o número de lugares que este pretende, um array com identificadores das cadeiras pretendidas e uma flag answered que indica se foi atendido ou não.