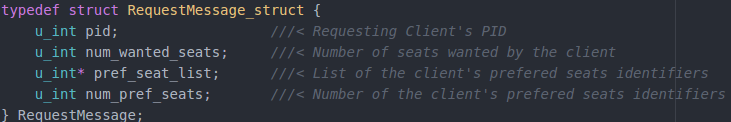
**Sistemas Operativos**

Relatório 2º Projeto

**Estrutura de Mensagens**

Após o programa do cliente ser iniciado e todos os argumentos serem processados e verificados, o pedido do cliente é guardado numa estrutura denominada ‘*Request Message*’.



No envio da mensagem para o servidor, esta mensagem é processada para uma única linha de texto, contendo toda a informação do pedido, com a seguinte estrutura:

**<pid> <numLugaresDesejados> <preferencia1> <preferencia2> … <preferenciaN>**

Após processamento do pedido por parte do servidor, este envia uma mensagem de resposta ao cliente. Esta mensagem pode ser correspondente a um erro, consistindo apenas num inteiro com o número negativo identificador do erro (**<idErro>**). Em caso de sucesso, é enviada uma mensagem contendo todos os identificadores dos lugares que foram efetivamente reservados:

**<lugarReservado1> <lugarReservado2> <lugarReservado3> … <lugarReservadoN>**

Estas mensagens são escritas atomicamente, a fim de evitar sobreposição de mensagens e prevenindo erros de leitura.

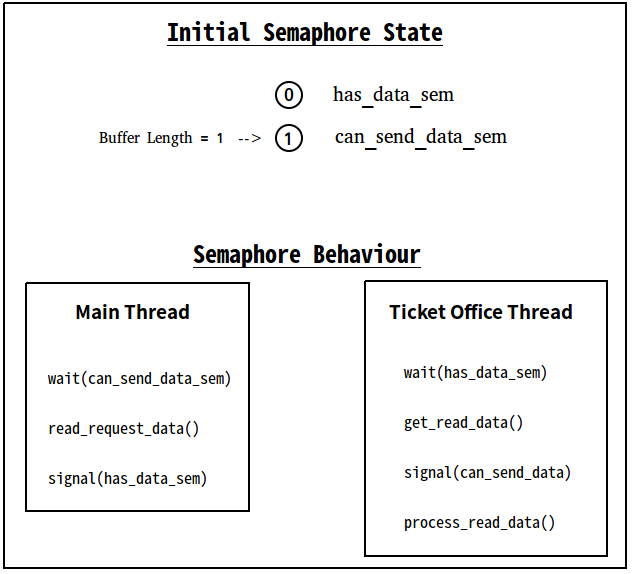
**Encerramento do Servidor**

Após o tempo de funcionamento do servidor terminar, todas as bilheteiras são notificadas. Após serem notificadas, as bilheteiras deixam de aceitar novos pedidos por parte dos clientes, **mas,** se estiverem a processar um pedido aquando o momento de encerramento do servidor, terminam de processar o pedido que estão a realizar nesse momento. Após terminarem o pedido em questão e enviarem a a resposta ao cliente, as bilheteiras encerram. O programa principal espera pela terminação de cada um dos *threads* (bilheteiras), escrevendo no ficheiro de *log* a terminação dos mesmos e, por fim, o programa principal encerra, escrevendo no ficheiro de *log* a mensagem “SERVER CLOSED”. Todos os descritores de ficheiros abertos são fechados

**Mecanismos de Sincronização Utilizados**

Foram utilizados semáforos e mutexes para proceder à sincronização entre *threads* (bilheteiras).

Quanto à sincronização do buffer unitário, foram utilizados dois semáforos, cujo funcionamento está descrito no seguinte esquema:



Quanto à sincronização de acesso ao array de seats, foi utilizado um mutex para garantir que o acesso a este é mutuamente exclusivo, isto é, apenas uma bilheteira pode aceder ao array de seats de cada vez. Esta sincromização é garantida através de um *mutex lock* no início da parte do processamento do pedido referente ao acesso ao array de seats, seguido de um *mutex unlock* no final da mesma.

**Notas relativas à Compilação**

Cada programa tem um *makefile* próprio. Para compilar a totalidade do programa (client + server), deve-se correr o script “build.sh”, que executa o *makefile* de cada programa e coloca o binário resultante de cada programa na mesma pasta (condição necessária para o funcionamento).