Relatório

Mensagens entre Programas

A troca de mensagens entre Clientes e Servidor é feita de forma bidirecional, isto é, o Cliente envia mensagens ao Servidor (pedidos), e este também envia mensagens ao Cliente (respostas aos pedidos). De seguida iremos abordar cada uma destas trocas de mensagens.

Mensagens Cliente - Servidor

Quando um cliente tenciona comunicar com o Servidor, isto é, realizar um pedido, é necessária a abertura de uma via de comunicação entre estes. Para isso é aberto um FIFO de escrita no Cliente e um de leitura no Servidor, com o nome requests. Depois da abertura do FIFO o Cliente envia a informação necessária para realizar um pedido de forma sequencial, do tipo inteiro:

- PID do cliente.
- Número de lugares pretendidos.
- Tamanho da lista dos lugares preferidos.
- Lista dos lugares preferidos (enviados 1 a 1).

Uma vez recebido o pedido na extremidade de leitura o servidor redireciona-o para uma das bilheteiras disponíveis.

Mensagens Servidor – Cliente

Quando o Servidor tenciona comunicar uma resposta ao pedido de um determinado Cliente este comunica através da criação de um FIFO com o Cliente. O Servidor cria um FIFO novo para cada cliente diferente a que seja necessário enviar a resposta, com o nome ans[PID do Cliente]. O Servidor envia ao Cliente a resposta ao pedido de reserva que pode ter dois formatos:

Pedido com insucesso:

- Código do motivo de falha.

Pedido com sucesso:

- Número de lugares reservados.
- Conjunto dos lugares reservados (enviados 1 a 1).

Todos os dados são do tipo inteiro.

Mecanismos de Sincronização

Como mecanismos de sincronização, neste projeto foram utilizados mutexes e variáveis de condição. Estes aparecem quando:

- uma bilheteira necessita de aceder aos pedidos de cada cliente.
- ao tratar cada lugar da sala onde irá decorrer o evento.
- é necessário escrever nos ficheiros.

Assim que é recebido pelo servidor um determinado pedido é criado, guardado e posto em espera. Quando isto acontece é assinalada a variável de condição associada ao mutex dos requests. Isto fará com que apenas uma das bilheteiras acorde e leia o pedido, prevenindo assim que o mesmo pedido seja atribuído a várias bilheteiras e que as bilheteiras tenham de dar poll constantemente ao request guardado (prevenindo assim *busy waiting*).

Ainda assim, bilheteiras diferentes podem ter pedidos que envolvam os mesmo lugares. Deste modo cada um dos lugares da sala possui um mutex. Assim, antes de fazer qualquer alteração a um lugar, o seu mutex é devidamente bloqueado prevenindo conflitos que pudessem surgir com as diferentes bilheteiras a tentar aceder ao mesmo lugar.

Por fim, cada bilheteira, antes de reportar um acontecimento escevendo num ficheiro, bloqueia o seu mutex referente a logs.,impedindo outras bilheteiras de tentar aceder a um ficheiro que já estava a ser acedido por uma outra bilheteira para escrita.

Encerramento do Servidor

O encerramento do Servidor é feito após o tempo de funcionamento das bilheteiras expirar. Para isso foi utilizada a função ***alarm()*** da biblioteca ***unistd.h*** para possibilitar a implementação desta funcionalidade.

A função ***alarm(int seconds)*** irá fazer com que o sistema gere o sinal SIGALRM ao fim de um número de segundos especificado por seconds ter passado. Ao ser gerado o sinal SIGALRM, o handler deste sinal irá apenas alterar o valor de uma variável global (***bool timeout***) indicando o encerramento das bilheteiras. Esta alteração vai fazer com que a thread principal espere que cada uma das bilheteiras termine antes de sair do programa. Assim, os pedidos atuais de cada bilheteira serão ainda tratados e só no final destes é que as bilheteiras fecham.

Ao encerrar o Servidor tanto a thread principal como as bilheteiras têm de encerrar as suas estruturas de dados e libertar a memória previamente alocada por cada uma delas. No caso das bilheteiras, em cada uma delas é fechado o FIFO que comunica com o Cliente atual e é apagado o pedido que estava a ser tratado. No caso da thread principal, esta é responsável por eliminar todos os lugares do evento em questão, destruir todos os mutexes e fechar o FIFO requests que recebe os pedidos dos clientes.