Trabalho Prático Nº 2

Simulação de um sistema de *home banking*

Mensagens Trocadas entre Utilizadores e Servidor

Tal como pedido no guião do trabalho prático, existe troca de mensagens entre um Utilizador (user) e um Servidor (server), de modo a simular um sistema de *home banking*. O Utilizador, que tanto pode ser o Administrador do Servidor como também apenas um Cliente, envia um pedido ao Servidor que, após executar a operação pedida, responde ao Utilizador com os dados obtidos através da operação.

Mensagens Utilizador->Servidor

Para um utilizador comunicar com um servidor, duas coisas têm que acontecer previamente: o servidor e também o canal de comunicação (**FIFO**) entre os dois têm que existir. Sempre que um utilizador pretende enviar um pedido, abre o FIFO em modo de escrita e coloca neste as informações necessárias para ser realizada a operação por parte do servidor, ou seja, coloca uma estrutura do tipo ***tlv_request_t***, fornecida pelos docentes da unidade curricular. Esta estrutura é suficientemente genérica para que o servidor possa executar diferentes operações apenas a partir desta.

Após receber esta estrutura, o servidor coloca-a numa fila de pedidos, que será posteriormente lida por um dos balcões eletrónicos (***threads***).

Mensagens Servidor->Utilizador

Após ser executada a operação no balcão eletrónico, este retorna ao utilizador uma estrutura do tipo ***tlv_reply_t***, também fornecida pelos docentes, que contém, entre outras coisas, um código que indica se a operação foi, ou não, bem sucedida. A reposta é enviada através de um novo FIFO, criado pelo utilizador, o que significa que para cada utilizador teremos um novo FIFO.

Mecanismos de Sincronização

Neste projeto utilizamos vários mecanismos de sincronização lecionados nesta unidade curricular, sempre que são acedidas variáveis partilhadas por vários processos/threads. Estes mecanismos são mutexes e semáforos.

De modo a utilizar uma abordagem de produtor/consumidor utilizamos semáforos, que sinalizam a "produção" de um pedido, ou seja, a colocação deste na fila de pedidos, e o seu "consumo" por parte dos balcões eletrónicos, sempre que um elemento é retirado da mesma fila. Deste modo, cada balcão eletrónico só lê um pedido quando este é colocado na sua fila e, caso não exista nenhum pedido, todos esperam que seja colocado outro pedido na fila.

O array de contas pode ser acedido por várias threads simultaneamente, pois podem ser executadas ações concorrentemente. Deste modo é obrigatória a utilização de mutexes, no nosso caso, de um array de mutexes, que restringe o acesso exclusivo a cada conta. Assim, é possível aceder a diversas contas simultaneamente por várias threads, mas não é possível aceder à mesma conta, evitando, por exemplo, que a visualização de um saldo de uma conta seja realizada ao mesmo tempo que uma transferência da mesma conta.

Outros mutexes são utilizados, por exemplo, no acesso à fila de pedidos, de modo a impedir que dois balcões leiam o mesmo pedido, e também para aceder à variável partilhada que indica se o servidor já teve ordem para ser encerrado.

Encerramento do Servidor

Após enviado um pedido de shutdown para o Servidor, caso este tenha sido enviado pelo Administrador e após a sua autenticação, são mudadas as permissões do FIFO para modo de leitura apenas (através da função ***fchmod()***), e processados os pedidos ainda na fila ou no buffer do FIFO. É também enviada uma estrutura do tipo ***tlv_reply_t*** para o Administrador com o número de balcões eletrónicos ativos no momento do encerramento.

Após processar todos os pedidos pendentes, procede-se à terminação e recolha dos threads ativos, destruição dos mecanismos de sincronização e comunicação, e terminação do programa.