Universidade do Porto
FEUP Faculdade de Engenharia

TRABALHO PRÁTICO Nº 2

Simulação de um sistema de home banking

Implementação de uma arquitetura cliente/servidor baseada em FIFO

1. Estrutura de mensagens trocadas entre cliente e servidor

As mensagens trocadas entre o programa do cliente e o programa do servidor resumem-se às duas structs principais fornecidas: **tlv_request_t** e **tlv_reply_t**. As mensagens têm o formato **TLV** (type, length, value), sendo value também uma struct com membros que variam consoante o tipo de operação pedida.

```
typedef struct tlv_request {
  enum op_type type;
  uint32_t length;
  req_value_t value;
} __attribute__((packed)) tlv_request_t;
```

```
typedef struct tlv_reply {
  enum op_type type;
  uint32_t length;
  rep_value_t value;
} __attribute__((packed)) tlv_reply_t;
```

2. Mecanismos de sincronização utilizados

Para a sincronização e gestão do acesso às contas bancárias foi utilizado um **mutex** por cada conta. Para isso, foi criado um array de mutexes, que são inicializados aquando da criação da conta e cujo índice corresponde ao id da conta. É feito um mutex_lock antes do processamento do pedido para aceder à(s) conta(s) e um mutex_unlock depois, garantindo, assim, que o acesso a cada conta é mutuamente exclusivo, isto é, que duas threads não acedem à mesma conta simultaneamente.

```
pthread_mutex_t accountsMutex[MAX_BANK_ACCOUNTS];
```

```
mutex_lock_account(threadID, SYNC_ROLE_ACCOUNT,
request->value.create.account_id);
```

```
mutex_unlock_account(threadID, SYNC_ROLE_ACCOUNT,
request->value.header.account_id);
```

Já a sincronização e gestão da fila de pedidos foi interpretada como sendo um caso típico da situação produtor-consumidor. Para isto, foram criados dois **semáforos sem nome**, sendo que um representa o número de pedidos da fila e o outro o número de vagas da mesma. Este mecanismo permite que o produtor (que será a função main, neste caso) não consiga colocar mais pedidos na fila caso esta já tenha atingido a capacidade desejada e, por outro lado, que os consumidores (que serão as threads, neste caso) apenas acedam à fila de pedidos quando esta não estiver vazia.

Turma nº 6
Grupo nº 4

João Praça
Lucas Ribeiro
Silvia Rocha

3. Forma de encerramento do servidor

Após o servidor receber um pedido válido de encerramento do administrador, deixa de poder receber novos pedidos de operações. Todos os balcões (threads) em funcionamento terminam o pedido que estão a processar e, caso necessário, processam ainda todos os pedidos pendentes da fila de pedidos. Quando a fila de pedidos estiver vazia, os balcões fecham (ou seja, as threads terminam) e de seguida o programa principal termina.

João Praça
Lucas Ribeiro
Silvia Rocha