Sistemas Operativos - Relatório 2ºProjeto (T1G09)

Estrutura de mensagens

Para a troca de mensagens entre clientes e servidor, utilizamos as estruturas de dados que nos foram fornecidas, tlv_request e tlv_reply, para pedidos efetuados pelo user e respostas enviadas pelo server, respetivamente. Estas encontram-se no formato TLV (type, length e value) e incluem outras estruturas (req_value), cujo tamanho irá variar conforme o tipo de operação.

Após todos os argumentos terem sido processados, ao iniciar o programa do user, os seus pedidos são guardados numa estrutura tlv_request e enviados através do FIFO “secure_srv”. O server, por sua vez, valida estes pedidos e processa-os, guardando as respostas numa estrutura tlv_reply e enviando-as pelos FIFOs “secure_XXXXX”.

Mecanismos de sincronização

Para tratar os pedidos do user, aplicamos os mecanismos de sincronização de acordo com o problema do produtor consumidor. Inicializamos assim dois semaforos, “full” e “empty” com os valores respetivamente de 0 e o número de balcões abertos pelo server. Antes de ler o próximo request do user, a função main espera que o valor no semáforo empty seja superior a 0 (sem_wait(&empty)), indicando que há um balcão online livre para tratar do pedido. Assim que o pedido for acrescentado à fila de pedidos, é incrementado o valor do semáforo full (sem_post(&full)). De forma oposta, no lado dos balcões, estes apenas tratam um pedido quando o valor do semáforo full for superior a 0, indicando que há pedidos na fila para serem tratados e no final do seu tratamento notificam a main que estão livres para tratar novos pedidos, incrementando o valor de empty (sem_post(&empty)).

Para a realização das operações bancárias, em vez de utilizar um array de contas, utilizamos um array de estruturas, cada uma contendo uma conta (bank_account_t) e um mutex. Corresponde-se, então, um mutex a cada conta, possibilitando a diferentes balcões eletrónicos (threads) realizar operações com diferentes contas. Para evitar a ocorrência de deadlocks nas operações de transferência, procede-se ao lock aos mutexes de ambas as contas, realiza-se a operação, e segue-se o unlock dos mutexes envolvidos.

```c
request.type = 0;
do
{
    int bytesRead;
    bytesRead = read(serverFifo, &request.type, sizeof(op_type_t));
    if (bytesRead > 0)
    {
        int getVal;
        sem_wait(&empty);
        sem_getvalue(&empty, &getVal);
        logSyncMechSem(server_log_file, MAIN_THREAD_ID, SYNC_OP_SEM_WAIT, SYNC_ROLE_PRODUCER, getpid(),getVal);
        read(serverFifo, &request.length, sizeof(uint32_t));
        read(serverFifo, &request.value, request.length);
        logRequest(server_log_file, MAIN_THREAD_ID, &request);
        enQueue(requestQueue, request);
        sem_post(&full);
        sem_getvalue(&full, &getVal);
        logSyncMechSem(server_log_file, MAIN_THREAD_ID, SYNC_OP_SEM_POST, SYNC_ROLE_PRODUCER, getpid(),getVal);
    }
} while (!terminate);
```

```c
            logReply(server_log_file, pthread_self(), &reply);
        }
        else logReply(server_log_file, pthread_self(), &reply);
        write(user_fifo, &reply, sizeof(op_type_t) + sizeof(uint32_t) + reply.length);
        sem_post(&empty);
        sem_getvalue(&empty, &val);
        logSyncMechSem(server_log_file, pthread_self(), SYNC_OP_SEM_POST, SYNC_ROLE_CONSUMER,
    }
    logBankOfficeClose(server_log_file, x, pthread_self());
    pthread_exit(0);
}
```

```c
void *officeprocessing(void *requestQueue)
{
    int x = thread_count;
    logBankOfficeOpen(server_log_file, x, pthread_self());
    while (!(terminate && ((reqQ_t *) requestQueue)->front != NULL)){
        sem_wait(&full);
        int val;
        sem_getvalue(&full, &val);
        logSyncMechSem(server_log_file, pthread_self(), SYNC_OP_SEM_WAIT,
        if(((reqQ_t *) requestQueue)->front == NULL && terminate) {
```

Encerramento do servidor

Quando o encerramento é solicitado pelo admin do sistema, deixa de ser possível ao user enviar pedidos através do FIFO “secure_srv” (alteramos as permissões deste FIFO para apenas leitura), processando, contudo, os que ainda estiverem pendentes. O programa termina então assim que todos os balcões (threads) tiverem terminado os pedidos e encerrado.