Faculdade de Engenharia
Universidade do Porto

Servidor de Aplicações

Mini Projeto 2 - Parte 2
Sistemas Operativos

Turma 5 Grupo 6
Mário Travassos up201905871@edu.fe.up.pt
José Costa up201907216@edu.fe.up.pt
Tomás Fidalgo up201906743@edu.fe.up.pt
Flávio Vaz up201509918@edu.fe.up.pt

20 de Maio, 2021

Contents

Chapter 1

Introdução

As duas tarefas deste projeto consistem na elaboração de uma aplicação do tipo cliente-servidor, capaz de lidar com pedidos de execução de tarefas. Estas tarefas são comunicadas por um processo cliente multithreaded a um servidor, também ele multithreaded. Os pedidos de tarefas ficam numa fila de atendimento até terem vez de serem processados. O servidor multithread gere os pedidos recebidos e transfere-os à biblioteca que os vai executar. Após serem recolhidos os resultados, estes são enviados ao cliente.

Esta etapa incidiu no desenvolvimento da porção relativa ao servidor desta aplicação, tendo-nos sido disponibilizado o código referente à implementação do cliente, juntamente com um script de teste, a fim de validar a nossa implementação. Mais uma vez, a bilbioteca atua como caixa negra, tendo-nos sido também disponibilizada.

Chapter 2

Detalhes da implementação

A nossa implementação cumpre com sucesso todos os requisitos do projeto:

- O Servidor é um programa multithread, funcionando com o máximo de paralelismo possível, e evitando situações de deadlock, colisão ou esperas ativas;
- As tarefas a serem executadas no servidor, tal como na etapa anterior, são devidamente identificadas e todos os seus atributos são recebidos, e transmitidos ao consumidor corretamente;
- A resposta é enviada corretamente ao cliente;
- As operações efetuadas são registadas com sucesso na saída padrão.

2.0.1 Invocação

A invocação do cliente é realizada da seguinte forma:

$$c < -tnsecs > [-lbufsz] < fifoname >,$$

onde nsecs é o número aproximado de segundos que o programa deve funcionar, bufsz é o tamanho do buffer ou armazém onde os resultados dos pedidos serão armazenadaos, e fifoname é o nome absoluto ou relativo do canal público de comunicação pelo qual o cliente envia pedidos ao servidor.

2.0.2 Componente multithreaded

De modo a obter o paralelismo desejado na operação do Servidor, foram utilizadas detatched threads, tanto para a thread consumidora como para as threads produtoras,, visto que esperar manualmente por cada uma das threads criadas implicaria, mais uma vez, ou limitar o número de threads possíveis de gerar, ou fazer uma implementação excessivamente complexa.

Estas threads são inicializadas como threads detatched, e não destacadas manualmente. Isto deve-se à possibilidade (se bem que bastante improvável) da thread criada ser tão rápida que termina a sua execução antes sequer de a thread principal a conseguir destacar. Isto foi feito definindo os atributos das threads antes da criação destas mesmas.

A thread consumidora é criada antes de o servidor começar a leitura de pedidos enviados pelo cliente através do fifo público.

A thread principal procura ler as mensagens enviadas pelo cliente através do fifo público. Aquando da receção de uma, o servidor tenta criar uma thread produtora, que tratará de processar este pedido. Esta tentativa é apenas realizada um número finito de vezes. Caso não seja possível, por uma qualquer razão, criar esta thread produtora, este pedido é ignorado, e a thread principal volta a tentar ler pedidos subsequentes.

Cada uma das threads produtoras está encarregue de enviar o resultado do seu pedido para o armazém. Este pedido é posteriormente enviado para o armazém, de onde a thread consumidora retira o resultado, enviando a resposta ao utilizador, através de um fifo privado pela thread de pedido.

O acesso ao armazém é gerido utilizando dois, semáforos, um responsável por bloquear a thread consumidora quando o armazém se encontra vazio (impedindo esta de tentar retirar de lá valores), e outro responsável por bloquear as threads consumidoras quando o armazém se encontra cheio.

Todas os registos pertinentes referentes às atividades das threads são registados devidamente na saída padrão.

2.0.3 Comunicação com o cliente

A comunicação com o cliente é executada através de FIFOS:

- um público, único, criado pelo servidor, através do qual os processos cliente lhe enviam mensagens relativas a pedidos de execuções de tarefas;
- vários privados, cada um criado por uma thread pedido diferente, através dos quais o servidor envia mensagens de resposta relativas à tarefa pedida pela thread correspondente;

2.0.4 Registo de operações

Todas as operações realizadas ao longo da execução do programa são registadas, como mencionado anteriormente. Estas operações são registadas na saída padrão (stdout), sob a seguinte forma:

$$inst; i; t; pid; tid; res; oper,$$

onde:

- **inst** é o instante no qual a operação foi realizada;
- **i** é número único de identificação do pedido;
- **t** é a carga da tarefa;
- **pid** é o identificador de sistema do processo (cliente no caso do pedido, do servidor no caso da resposta);
- **tid** é o identificador de sistema da thread (cliente no caso do pedido, do servidor no caso da resposta);
- **res** é a resposta do servidor (ou -1 no caso da mensagem pertencer ao cliente);

- **oper** é um identificador relativo à fase da operação que acabou de ser realizada.

2.0.5 Tempo

A contagem do tempo é realizada recorrendo à system call time(). Dentro do ficheiro de código fonte timer.c estão declaradas diversas funções que são chamadas ao longo do programa, retornando o resultado apropriado à situação. Para efeitos de prolongar ligeiramente a execução do servidor, foi criada uma constante a ser adicionada ao tempo restante na execução do programa em alguns saltos condicionais.

Chapter 3

Avaliação

As contribuições para este projeto foram igualmente distribuídas entre os membros do grupo.

- 25% - Mário Travassos
- 25% - José Costa
- 25% - Tomás Fidalgo
- 25% - Flávio Vaz