MC833 - Relatório do Projeto 1

Ieremies Romero, Lucas de Oliveira Silva

May 10, 2022

Introdução

Este projeto tem como o objetivo desenvolver uma comunicação cliente-servidor utilizando uma conexão TCP. Para isso, utilizamos um servidor concorrente capaz de fornecer, guardar e atualizar informações sobre filmes. O cliente, é capaz de recuperar ou alterar essas informações via protocolo HTTP.

Esse relatório tem por objetivo descrever o processo e as decisões. Para a documentação mais técnica, referencia-se aos arquivos fonte, ou de forma online no site `https://ieremies.dev/mc833-projeto/html/` ou via pdf.

Descrição

Para esse projeto, desenhamos nosso sistema de forma modular e escalonável. Assim, o servidor possui operações simples e diretas, enquanto que o refinamento de tais fica a cargo do cliente.

Nosso servidor serve para fornecer uma lista de filme e suas informações. Ele pode ser utilizado como um armazém de informações que diversos clientes podem acessa-lo de forma assíncrona e fazer mudanças.

Por exemplo, digamos que um usuário está procurando por um filme de ação. Caso o administrador queira adicionar mais um filme de ação enquanto isso ocorre, o servidor será capaz de fazer as modificações e já servir, quando requisitado, o novo conjunto de filmes.

Vale ressaltar que não há autenticação dos usuários, partimos do pressuposto que todos eles são administradores.

Casos de uso

Neste projeto, podemos utilizar `cmake` para construir o `Makefile`.

```
mkdir build
cd build
cmake ../CMakeLists.txt
make
```

Após compilado, dois executáveis estarão disponíveis: `server` e `client`.

O servidor, uma vez executado, sentará em silêncio aguardando conexões. Em contrapartida, o cliente irá propor operações e solicitar as informações.

Armazenamento e estruturas de dados

As informações armazenadas estão contidas no arquivo `data/movie.h` que contém a `struct Movie` composta dos seguintes campos:

`id` Um identificador numérico único.

`title` Uma string contendo o título do filme.

`num_genres` A quantidade de gêneros cadastrados ao filme.

`genre_list` Os gêneros cadastrados ao filme.

`director_name` Uma string contendo o nome do diretor.

`year` O ano de publicação do filme.

Para armazenar uma lista de filmes, possuímos a `struct Catalog` definida no arquivo `data/Catalog.h` que se resume a um vetor de `struct Movie` e o indicador de quantos filmes a lista contém (`size`).

Implementação

Servidor

No lado do servidor, começamos populando a `struct addrinf` com algumas informações e passando-a para a função `getaddrinfo()` que nos retorna uma lista ligada de IP's disponíveis. Como indicado pelo tutorial fornecido, iteramos pelas possibilidades recuperando o socket descriptor, conferimos se está totalmente livre para ser utilizado com `setsockopt` e nos conectamos ao primeiro disponível com `bind()`.

Caso tudo isso tenha sido feito com êxito, garantimos que processos zombies serão tratados e avisar que o socket será limpo, antes de, finalmente,

começar a escutar por conexões. No laço `while`, aceitamos novas conexões com `accept()` e executamos `handle_client()`, função responsável por agir em vista das requisições do cliente.

É importante perceber que restringimos o tamanho do catálogo e das informações dos filmes para que caibam no datagrama do TCP.

Operações do servidor

O servidor é capaz de servir as seguintes operações:

GET retorna todo o catálogo.

POST coloca um novo filme no catálogo, determinando um ID único para ele. Caso não haja espaço, nada é feito.

PUT baseado no ID de uma struct movie, modificamos as informações do tal filme baseado nas informações preenchidas na struct.

DEL baseado no ID de uma struct movie, removemos o filme com a mesma ID do catálogo.

Cliente

No lado do cliente fazemos algo muito parecido. Populamos a `struct addrinf` com as informações e, junto com a função `getaddrinfo()`, iteramos pelos possíveis sockets até conseguirmos nos conectar a algum com o `bind()`. Caso tudo isso tenha sido bem sucedido, fazemos a limpeza final antes de passar o controle para a função `handle_user()`.

O cliente comunica-se com o servidor mediante as operações acima listadas, mas para o nosso usuário mais opções são apresentadas. Para as diferentes listagem de filmes e suas informações, fazemos esse filtro do lado do cliente, permitindo um controle maior de quais informações são apresentadas baseado no cliente. Seria possível, então, termos diversos clientes, com interfaces diferentes, conectando-se ao mesmo servidor.

Operações do cliente

Do ponto de vista do cliente, ele pode realizar as seguintes operações:

- cadastrar um novo filme, fornecendo suas informações.
- acrescentar um gênero a um filme.

- deletar um file a partir de seu identificador.
- listar filmes:
  - com seus títulos e indicadores.
  - com todas as informações.
  - de um gênero específico.
  - todas as informações de um filme em específico.

Conclusão

Assim, temos um servidor capaz de realizar operações e um cliente capaz de comunicar-se com o servidor e solicitar informações ao usuário.