MC833 - Projeto 1
Cliente e Servidor TCP

Caio Henrique Pardal - RA: 195216

Introdução

O projeto tem por objetivo estabelecer uma comunicação TCP, por meio de sockets, entre um cliente e um servidor. A aplicação desenvolvida consiste em um sistema de catálogo de filmes de um cinema.

O servidor dessa aplicação armazena as seguintes informações dos filmes em cartaz: Título, Sinopse, Gênero, Salas de exibição do filme e um identificador único para cada filme. Este servidor deve ser capaz de receber as requisições do cliente, localizado em uma máquina diferente da que o servidor estará rodando, e transmitir todas as informações disponíveis para cada uma das possíveis requisições que o cliente pode realizar.

Essas possíveis requisições são: Cadastro de novos filmes, recebendo como resposta positiva o respectivo identificador, Remoção de filmes a partir de seus identificadores, Listar o título e salas de exibição de todos os filmes, Listar todos os títulos de filmes de um determinado gênero, Retornar o título do filme a partir de seu identificador, Retornar todas as informações de um filme a partir de seu identificador e Listar todas as informações de todos os filmes.

Sistema

1. Descrição Geral

Explicando um pouco sobre a arquitetura dos arquivos e do sistema, ambas as partes (cliente e servidor) estão separadas em diferentes pastas e arquivos diferentes. O código principal (contendo as implementações das funções principais e regras de negócio) de são arquivos “.c” e existem arquivos “.h” para ambos, os quais definem as assinaturas de funções, incluem as bibliotecas necessárias e definem as constantes essenciais para o cliente e o servidor.

Em relação às constantes, deve-se ressaltar a importância de duas delas: “BUFFLEN” e “PORT”. A primeira é responsável por padronizar o tamanho da mensagem, fazendo com que haja um mapeamento 1:1 entre um envio do servidor com sua recepção no cliente, e vice-versa. A segunda define a porta em que deve-se estabelecer a conexão entre os dois serviços. Ambas devem assumir os mesmo valores no servidor e no cliente. Outros componentes importantes nos headers são os wrappers para as funções send e recv. Os wrappers, além de capturar erros das syscalls, garantem que o tamanho da mensagem enviada/recebida seja sempre igual ao valor de “BUFFLEN”, garantindo o mapeamento 1:1 citado anteriormente.

Os arquivos “.c” do servidor e do cliente são responsáveis por coordenar a troca de mensagens. O cliente executa essencialmente três operações: enviar mensagem, receber mensagem e receber arquivo. O servidor, analogamente, executa três operações: enviar

mensagem, receber mensagem e enviar arquivo. Troca de mensagens refere-se a enviar/receber uma única string contendo um comando ou dado, já troca de arquivos trata-se de uma série de trocas de mensagens contadas: antes de começar o envio do arquivo, é enviada uma mensagem com o tamanho do arquivo para que o cliente saiba quantos bytes ele deve receber.

Todo o fluxo de execução no servidor e no cliente são controlados por duas estruturas switch-case que aguardam uma a outra para realizar a comunicação coordenamente. Nota-se que a conexão é mantida em aberto até que seja encerrado o processo cliente ou o processo filho, correspondente do servidor. E nota-se também que para que seja estabelecida a conexão entre cliente e servidor, deve-se rodar o servidor em uma máquina e o cliente em outra (pode ser feito em terminais na mesma máquina também) por meio dos comandos (após terem sido compilados os programas): **Servidor: ./<nome do programa> e Cliente: ./<nome_do_programa> <ip_do_cliente> ou localhost, caso esteja na mesma máquina.**

Para a realização do armazenamento das informações, os dados são estruturados como arquivos. No servidor, a pasta “data/” armazena arquivos textos (.txt) com os dados dos filmes. Esses arquivos são nomeados de acordo com o identificador dos filmes (que geralmente buscam seguir o título do filme, ao menos que este possua um espaço no seu nome), e este identificador é utilizado como chave na busca de arquivos. Existe ainda dois outros arquivos dentro dessa pasta, um deles é o arquivo “index.txt”, utilizado para armazenar todos os identificadores dos filmes, presentes naquele momento no servidor, e acessar os arquivos quando a opção selecionada não especifica um identificador. E o outro, é um arquivo chamado “help.txt” que contém todas as opções de requisições disponíveis que um cliente pode fazer e este arquivo é enviado para o cliente quando o mesmo digita “h”, no terminal.

Como o cliente não armazena nenhum dado, mas apenas os exibe no terminal, este não possui uma pasta “data/” para armazenamento de informações.

2. Casos de Uso

Os casos de uso para o cliente e o servidor são:

Comando	Ação
“h” ou “help”	Mostra a lista de requisições e comandos disponíveis
1(espaço)<nome_do_filme>,<sinopse>,<gênero>,<salas de exibição separadas _por_traços(-)>	Cadastra um novo filme usando as informações passadas e recebe como resposta positiva o identificador do filme adicionado
2(espaço)<identificador>	Remove um filme a partir do identificador passado
3	Lista o título e salas de exibição de

	todos os filmes
4(espaço)<gênero>	Lista todos os títulos de filmes de um determinado gênero
5(espaço)<identificador>	Dado o identificador de um filme, retorna o título do filme
6(espaço)<identificador>	Dado o identificador de um filme, retorna todas as informações deste filme
7	Lista todas as informações de todos os filmes

3. Estrutura de dados e armazenamento

Os dados do servidor estão todos armazenados dentro do diretório “server/data”. A estrutura dos dados que são armazenados consiste em um arquivo “index.txt” para registrar os identificadores dos filmes cadastrados no sistema, arquivos “[identificador].txt” para armazenar as informações relativas ao filme em cartaz e um arquivo “help.txt” que armazena todas as requisições possíveis para o cliente fazer e o servidor processar e realizar.

Falando mais especificamente dos arquivos de cada um dos filmes, estes seguem uma estrutura própria: Na primeira linha do arquivo encontra-se o título do filme, na segunda a sua sinopse, na terceira o seu gênero e, por fim, na quarta, as salas de exibição daquele filme. Vide representação a seguir:

Linha 1	Título do filme
Linha 2	Sinopse
Linha 3	Gênero
Linha 4	Salas de exibição

4. Detalhes de implementação

Implementação do servidor TCP

O servidor foi implementado visando o armazenamento de dados de forma persistente e paralelismo entre conexões. A fim de garantir o paralelismo, o servidor TCP possui, inicialmente, um único processo pai que atua como dispatcher para novas conexões. O dispatcher é associado à porta “PORT” (especificada nos arquivos “.h”) com protocolo IPv4 (AF_INET) e, para que este socket seja associado à porta “PORT” de todas as interfaces de rede do computador, ele recebe o IP “INADDR_ANY”.

Enquanto o dispatcher aguarda novas conexões com o uso da syscall listen, os clientes já conectados são distribuídos aos processos filhos, que ficam responsáveis por atender às requests dos clientes associados a eles sem que o processo pai tenha que interromper a escuta por novas conexões.

Para fins de praticidade, supôs-se que todos os comandos fornecidos pelo cliente eram válidos dentro do escopo de operações do servidor (sem distinções de permissões de usuários) e também assumiu-se que as conexões são sempre estáveis. A finalidade das suposições são para evitar eventuais tratamento de erros, os quais trariam uma maior complexidade para o servidor e fugiria do objetivo proposto para o projeto.

Conclusão

O projeto criou uma comunicação TCP entre um cliente e um servidor, localizados em máquinas diferentes em uma rede local, para atender os requisitos das sete operações citadas e as condições de paralelismo e persistência.

Analisando-se de forma qualitativa as comunicações feitas entre o servidor e o cliente, pode-se perceber que as operações que são apenas referentes ao envio de informações específicas de um filme são mais rápidas do que àquelas que necessitam do envio de todos os dados de todos os filmes. O que faz sentido com o proposto para o projeto, pois em um caso estamos enviando mensagens menores e com tempo de busca menor no servidor e, no outro, estamos enviando todas as informações e buscando em vários arquivos diferentes.

Dessa maneira, pode-se afirmar que o projeto atendeu os requisitos para um servidor TCP concorrente e apresentou uma performance coerente com o esperado. A persistência e confiança na transmissão de arquivos foi garantida em ambos, servidor e cliente, pelo protocolo TCP e a interação adequada dos programas com o protocolo através de wrappers.

Referências

Beej's Guide to Network Programming (http://beej.us/guide/bgnet/html/single/bgnet.html)