Trabalho Prático 1
Controle de Envio de Arquivos

Fernanda Carolina da Silva Pereira
Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte – MG – Brasil
fernanda.pereira@dcc.ufmg.br

1. Introdução

O objetivo do trabalho foi praticar os conceitos aprendidos na disciplina de Redes de Computadores. O problema proposto foi a implementação de um sistema de transferência de arquivo simples entre um servidor e um cliente usando *sockets* em C. O sistema permite que um cliente se conecte a um servidor e envie um arquivo, que será recebido e armazenado pelo servidor em seu sistema de arquivos. O cliente receberá uma confirmação do servidor indicando que o arquivo foi recebido com sucesso.

O sistema é composto por duas partes principais: o servidor e o cliente. A implementação dessas partes envolve o uso de funções básicas de manipulação de *sockets* e arquivos em C.

2. Execução do Programa

Para execução do programa em ambiente Linux, atendendo ao requisito de se ter o gcc instalado, siga os seguintes passos:

- **Passo 1:** abrir a pasta …/TP1_2019431380 em dois terminais.
- **Passo 2:** digitar o comando *make clean* em qualquer um dos terminais para remover possíveis arquivos antigos que possam gerar saídas inesperadas.
- **Passo 3:** digitar o comando *make* em qualquer um dos terminais para compilar os arquivos e gerar os executáveis do programa.
- **Passo 4:** no terminal 1, digite:
  ./server v6 <porta> (em caso de uso de protocolo IPv6)
  ./server v4 <porta> (em caso de uso de protocolo IPV6)
- **Passo 5:** no terminal 2, digite:
  ./client <IPv6> <porta> (em caso de uso de protocolo IPv6)
  ./client <IPv4> <porta> (em caso de uso de protocolo IPV4)

3. Implementação

O programa foi implementado em C e a biblioteca *socket* foi utilizada para a comunicação entre cliente e servidor. Ao que tange a comunicação cliente e servidor a implementação foi baseada nos vídeos do professor Ítalo [1] e não será abordado neste documento. Os demais detalhes de implementação são detalhadas nas subseções a seguir.

client.c

Este arquivo contém o código-fonte de um programa cliente que se comunica com um servidor usando soquetes TCP/IP. O programa cliente permite ao usuário selecionar e enviar arquivos para o servidor.

Dependências
common.h: arquivo de cabeçalho contendo definições comuns e protótipos de função;
stdlib.h: biblioteca padrão para funções de uso geral;
stdio.h: biblioteca de entrada/saída padrão para operações de entrada e saída;
string.h: biblioteca de manipulação de strings para operações relacionadas a strings;
unistd.h: biblioteca API do sistema operacional POSIX para várias chamadas e constantes do sistema;
sys/types.h: arquivo de cabeçalho que fornece os tipos de dados básicos usados pelas chamadas do sistema;
sys/socket.h: arquivo de cabeçalho que fornece as definições para funções relacionadas a soquete e estruturas de dados;
arpa/inet.h: arquivo de cabeçalho que fornece as definições para operações na Internet.

Constantes
BUFSZ: Número inteiro constante que define o tamanho do buffer usado para transmissão de dados.

Função: *is_valid_extension(const char *filename)*
Verifica se a extensão de arquivo fornecida é válida com base em uma lista predefinida de extensões válidas. Retorna 1 se a extensão do arquivo for válida e 0 caso contrário.

Fluxo da função principal:
Após criar e configurar a conexão de um *socket* para o cliente, entra em um loop que solicita comandos ao usuário e os processa até que o programa seja finalizado.

O usuário insere um comando, que é analisado e verificado quanto à validade.

- Se o comando for "*select file*", o caminho do arquivo selecionado é armazenado e verificado quanto à validade e existência.
- Se o comando for "*send file*" e um arquivo estiver selecionado, o arquivo será aberto e o comando e o conteúdo do arquivo serão enviados ao servidor.
- Se o comando for "*enviar arquivo*", mas nenhum arquivo estiver selecionado, uma mensagem de erro será exibida.
- Se o comando for "*exit*", o programa o enviará para o servidor, fechará o *socket* e será encerrado.
- Se o comando for inválido, uma mensagem de erro será exibida.

server.c

Este arquivo contém o código-fonte de um programa de servidor que escuta conexões de clientes e recebe arquivos enviados por clientes.

Dependências
common.h: arquivo de cabeçalho contendo definições comuns e protótipos de função;
unistd.h: biblioteca API do sistema operacional POSIX para várias chamadas e constantes do sistema;
stdlib.h: biblioteca padrão para funções de uso geral;
stdio.h: biblioteca de entrada/saída padrão para operações de entrada e saída;
string.h: biblioteca de manipulação de strings para operações relacionadas a strings;
sys/types.h: arquivo de cabeçalho que fornece os tipos de dados básicos usados pelas chamadas do sistema;
sys/socket.h: arquivo de cabeçalho que fornece as definições para funções relacionadas a soquete e estruturas de dados;

Constantes

BUFSZ: Número inteiro constante que define o tamanho do buffer usado para transmissão de dados.

Fluxo da função principal:
Após criar e configurar a conexão de um *socket* para o servidor, cria um novo soquete *csock* para comunicação com o cliente e entra em um loop para receber comandos e arquivos do cliente até que o programa seja encerrado.

Assim, recebe um comando do cliente usando a função *recv()*.

- Se o comando for "*send file*": extrai o caminho do arquivo da mensagem recebida; imprime uma mensagem indicando o caminho do arquivo recebido; verifica se o arquivo já existe no servidor e imprime uma mensagem correspondente; cria um novo arquivo no servidor e recebe o conteúdo do arquivo do cliente, gravando-o no arquivo.
- Se o comando for "*exit*": imprime uma mensagem indicando que o cliente encerrou a conexão; fecha o *socket* do cliente e o *socket* do servidor. Ao fim, retorna 0, indicando execução bem-sucedida do programa.

4. Conclusão

Apesar da playlist do professor ter ajudado bastante, tive dificuldades para implementar a parte da leitura e escrita do arquivo efetivamente. Realizei diferentes tentativas e a que melhor funcionou até a finalização foi a atual, mas acredito que há melhores formas de implementação. Ademais, o programa está apresentando saídas conforme o esperado pelos casos de testes disponibilizados.

5. Bibliografia

[1] Introdução a Programação em Redes. Professor ítalo Cunha. Disponível em <Introdução a Programação em Redes - Youtube>.