Relatório do projeto 1

Isadora Sophia Garcia Rodopoulos *
Matheus Mortatti Diamantinos †
Luiz Fernando Bittencourt‡

Abstract

O objetivo do trabalho se baseou em implementar uma estrutura de cliente e servidor que interagissem entre si.

Neste projeto, foi implementado uma estrutura de comunicação entre cliente e servidor baseado em uma conexão TCP utilizando sockets na linguagem C. Nela, o cliente pode mandar mensagens de texto para o servidor que, ao confirmar o recebimento, retorna a mesma mensagem como um Acknowledge devolta ao cliente.

1. client.c

O cliente precisa executar os seguintes passos:

1. Criar o socket de conexão
2. Estabelecer conexão com o servidor
3. Receber mensagem do usuário, mandar ao servidor e esperar resposta
4. Se algum erro ocorrer ou o cliente fechar a aplicação, fechar a conexão

Para isso, utilizou-se funções da library `<netdb.h>`, que nos fornece as implementações necessárias para criarmos a conexão com o servidor. A seguir, serão detalhadas as funções utilizadas e seu contexto na implementação do projeto.

1. Criar o socket de conexão

Code 1. Função utilizada para criação do socket

```
int socket(int domain, int type, int protocol);
```

Code 2. Aplicação da função na implementação do projeto

```
/* create active socket */
s = socket(AF_INET, SOCK_STREAM, 0);
```

A função acima recebe como parâmetro a família da qual o endereço pertence (no nosso caso, IPv4), o tipo de socket (`SOCK_STREAM`, que fornece streams de byte sequenciados, confiáveis e bidirecionais), e o protocolo a ser utilizado (0 significa que o socket vai utilizar um protocolo padrão apropriado para o tipo do socket requerido).

2. Estabelecer conexão com o servidor

Para estabelecer a conexão com o servidor, primeiramente buscamos o host baseado no endereço fornecido pelo usuário:

Code 3. Função utilizada para acessar o endereço do host

```
/* get ip address */
host_address = gethostbyname(host);
if (host_address == NULL) {
    error("Invalid host name!\n");
}
```

Esta função nos retorna o endereço do servidor na forma de `struct_hostent`, que pode retornar `NULL` em caso de não achar o servidor.

A variável `host address` é do tipo `hostent` especificado abaixo:

Code 4. Struct_hostent

```
struct hostent {
    char*   h_name;
    char**  h_aliases;
    int     h_addrtype;
    int     h_length;
    char**  h_addr_list;
    char*   h_addr;
}
```

Contudo, apenas utilizamos `char *h_ddr` para acessarmos o endereço do host ao realizar a conexão TCP.

Assim, podemos utilizar a estrutura `sockaddr_in` para especificarmos a porta e o endereço do host para a conexão:

*RA 158018, Instituto de Computação, Universidade de Campinas, Unicamp. **Contact**: `isadorasophiagr@gmail.com`
†RA 156740, Instituto de Computação, Universidade de Campinas, Unicamp. **Contact**: `matdiamantino@gmail.com`
‡MC833, Instituto de Computação, Universidade de Campinas, Unicamp. **Contact**: `bit@ic.unicamp.br`

Code 5. Struct sockaddr_in

```
struct sockaddr_in {
        // AF_INET, no nosso caso
    short            sin_family;

    // Porta de conexao
    unsigned short   sin_port;

    // Endereco do host
    struct in_addr   sin_addr;
    char             sin_zero[8];
};
```

Code 6. Implementação no Projeto

```
        /* initialize data address */
    bzero((char*) &socket_addr, sizeof(socket_addr));
    socket_addr.sin_family
= AF_INET;              /* ipv4 addresses */
    socket_addr.sin_port
= htons(SERVER_PORT);
    socket_addr.sin_addr = *(struct in_addr*)host_address->h_addr;
```

3. Receber mensagem do usuário, mandar ao servidor e esperar resposta

4. Se algum erro ocorrer ou o cliente fechar a aplicação, fechar a conexão

Code 7. Conexão do cliente com o endereço do servidor

```
python code here to show something
```

Explica mais e sei lá o que e `i can talk code too`.

Code 8. descreva aqui

```
moar code it never ends
```

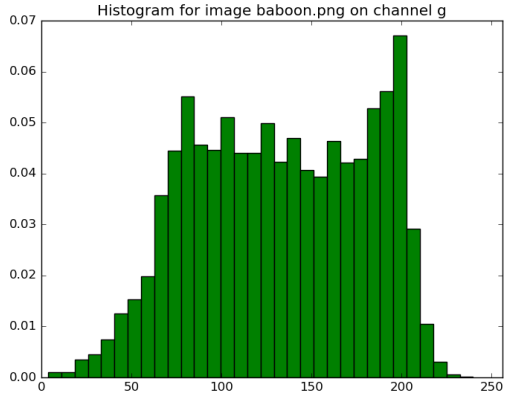

Figure 1. Exemplo de imagem

2. server.c

ablablabla aqui explica o servidor

Code 9. aaaaaa

```
vou parar de code
```

e pretty much isso.

3. Conclusões finais

precisa mesmo?? coloquei mais por whatever.