Relatório da atividade 2.1

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

Abstract

O objetivo do trabalho se baseou em modificar a implementação da aplicação cliente-servidor anterior, substituindo `fork()` por `select()`.

Neste projeto, foi implementado uma estrutura de comunicação entre cliente e servidor baseado em uma conexão TCP utilizando sockets na linguagem C. Nela, vários clientes podem se conectar a um mesmo IP, cada um por uma porta diferente. Foi utilizada a estrutura de `select()` para permitir que vários clientes se conectem simultaneamente.

Além disso, um *ack* é enviado para cada um dos clientes, em formato de *echo*, para certificar que a mensagem chegou com sucesso.

1. client.c

O código para o cliente não sofreu nenhuma modificação funcional em relação aos projetos anteriores, uma vez que o seu comportamento continua o mesmo, basta conectar-se ao IP do servidor e a uma porta disponível. Então, imprime a porta e o ip do servidor conectado.

O fluxo de execução de `client.c` pode ser resumido, portanto, em:

1. Criar o socket de conexão;
2. Estabelecer conexão com o servidor e imprimir porta e ip para o usuário;
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 1. Recuperação da porta e IP do servidor conectado

```
/* Get socket IP and Port and show to client */
struct sockaddr_in server;
socklen_t server_size = sizeof(server);

int res = getsockname(s,
    (struct sockaddr *)&server, &server_size);

char ip[INET_ADDRSTRLEN+1];
inet_ntop(AF_INET, &(client.sin_addr),
    ip, INET_ADDRSTRLEN);
ip[INET_ADDRSTRLEN] = '\0';

/* print text on screen */
fprintf(stdout, "<- IP: s PORT: d Connected!\n",
    ip, ntohs(client.sin_port));
```

2. server.c

Semelhante ao comportamento do programa do cliente, o servidor foi divido em:

1. Criar o socket ativo e associá-lo a um descritor;
2. Enquanto está ativo:
   (a) Criar um processo filho.
   (b) Checar novas conexões de clientes.
   (c) Conectar com o cliente e imprimir o IP e a porta da conexão.
   (d) Checar mensagens dos clientes ativos. Caso receba uma mensagem, imprimi-la e retornar o *echo*, especificando o IP e o port do cliente.
   (e) Caso um cliente se desconecte, fechar a conexão e desalocar o espaço do cliente.

Foi estabelecida uma interface bem simples, que permite que o usuário veja as mensagens que serão ecoadas na tela - as quais correspondem às mensagens que o cliente envia através da conexão com o servidor.

2.1. Criar o socket ativo e associá-lo a um descritor;

Do mesmo modo que foi feito no cliente, foi criado um socket com a família da qual o endereço pertence (no nosso

*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`

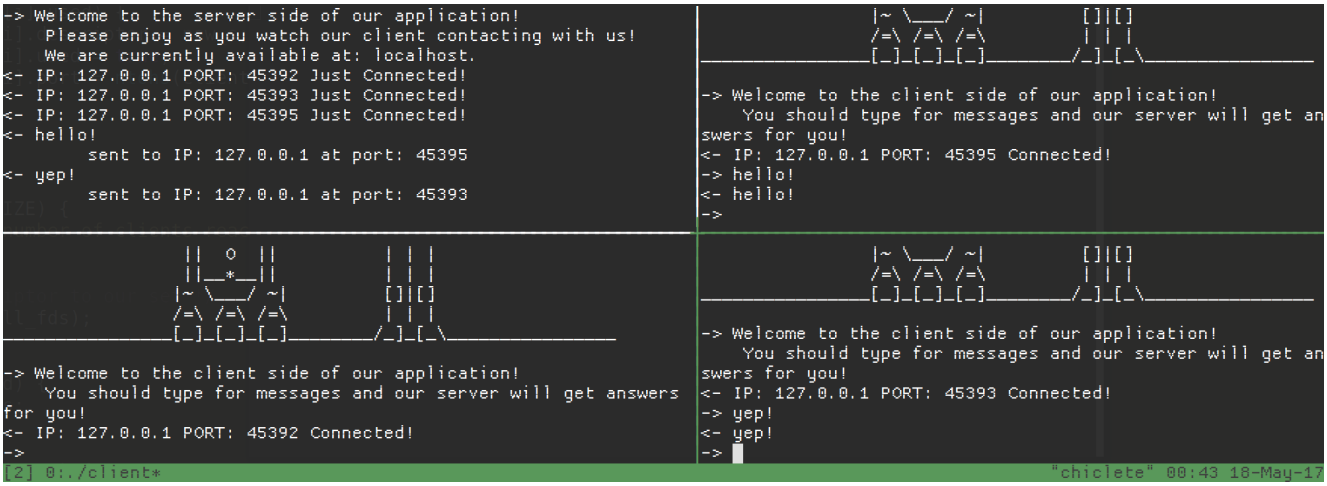

Figure 1. Exemplo de funcionamento do projeto

caso, IPv4), o seu respectivo port de *listening* e o IP, *localhost*. A associação do socket ao descritor se dá a partir da função *bind()*.

Caso ocorra algum erro ou o socket não consiga aceitar nenhuma conexão, um erro é emitido e o programa é finalizado.

2.2. Enquanto está ativo...

2.3. Criar um processo filho

Utilizou-se a função `fork()` para criar um processo filho que verifica se uma nova conexão é estabelecida.

2.3.1 Checar novas conexões de clientes;

Code 2. Estabelecimento de conexão a um novo cliente

```
/* Connect to client and
show IP and Port of the connection */
int32_t conn = accept(s,
    (struct sockaddr*) NULL, NULL);
valid(conn, "Failed to establish a connection
from socket.\n");
```

Do mesmo modo que nos projetos anteriores, checamos se o socket por uma nova conexão e conectamos.

2.4. Conectar com o cliente e imprimir o IP e a porta da conexão.

Code 3. Recuperação de IP e Porta do cliente

```
/* Connect to client and show
IP and Port of the connection */
struct sockaddr_in client;
socklen_t client_size = sizeof(client);

int res = getpeername(conn,
(struct sockaddr *)&client, &client_size);

char ip[INET_ADDRSTRLEN+1];
inet_ntop(AF_INET, &(client.sin_addr),
ip, INET_ADDRSTRLEN);
ip[INET_ADDRSTRLEN] = '\0';

/* print text on screen */
fprintf(stdout, "<- IP: s PORT: d Just Connected!\n",
ip, ntohs(client.sin_port));
```

Utilizando as funções acima, recuperamos o `peername` da conexão e imprimimos o IP e a Porta da conexão.

2.4.1 Checar mensagens dos clientes ativos. Caso receba uma mensagem, imprimi-la e retornar o *echo*, especificando o IP e o port do cliente;

Como temos vários processos filhos rodando concorrentemente o mesmo código, não houveram modificações nessa parte com relação ao projeto 1. Apenas imprimiu-se o IP de quem mandou a mensagem ao servidor no terminal:

Code 4. Impressão da porta de quem manda a mensagem

```
/* print message on screen and destination IP */
fprintf(stdout, "<- %s\tSent To: %s\n", buff, ip);
```

2.5. Comportamentos inesperados

Caso um cliente se desconecte, fechar a conexão e desalocar o espaço do cliente, ou qualquer erro ocorra durante os procedimentos descritos acima, é realizado o tratamento de erro e, caso seja necessário, a aplicação é encerrada imediatamente.

A mesma API utilizada nos projetos anteriores, `Api.h`, foi utilizada para encapsular as chamadas de erros.

3. Testes

Para testar o funcionamento da infraestrutura, foi necessário abrir o processo do servidor e diversos clientes para se conectar ao servidor por diferentes portas. Foram

testados tanto os corner quanto edge cases (como nenhum cliente ou muitos clientes se conectando simultaneamente) e foi checado se os erros foram tratados como o apropriado.

Apesar de algumas oscilações entre as conexões dos clientes (por algum motivo, eventualmente alguns programas do cliente não imprimiriam os *ack* do servidor até que a conexão fosse encerrada, por conflitos do próprio OS), o comportamento se manteve dentro o esperado.