Universidade Estadual de Campinas

Instituto de Computação - MC833 - A 2s2020

Atividade 2.1: Cliente e Servidor TCP

Luana Felipe de Barros **RA: 201705**

1. As funções utilizadas na comunicação via socket nos arquivos cliente.c e servidor.c foram:
    a. socket

    ```
    socket(AF_INET, SOCK_STREAM, 0)
    ```

    Utilizada tanto para o cliente quanto servidor, esta função cria um socket. constante AF_INET indica que o socket utilizará a versão 4 do protocolo IP na camada de rede, e utilizando o protocolo TCP na camada de transporte. A função socket retorna um número que é o descritor de socket, no cliente em sockfd e no servidor como listenfd.

    b. connect

    ```
    connect(sockfd, (struct sockaddr *) &servaddr, sizeof(servaddr)
    ```

    Utilizada pelo cliente para iniciar uma conexão TCP fazendo a apresentação de 3 vias ao servidor remoto, que teve endereço IP e porta definidos na criação da estrutura do socket: sockaddr. Esta função recebe como parâmetros o descritor de arquivo do socket, a estrutura de dados do socket e seu tamanho.

    c. bzero

    ```
    bzero(&servaddr, sizeof(servaddr))
    ```

    Utilizada pelo cliente e servidor para inicializar a estrutura socket correspondente com valores nulos.

    d. bind

    ```
    bind(listenfd, (struct sockaddr *)&servaddr, sizeof(servaddr)
    ```

    Utilizada pelo servidor, faz a ligação do descritor de socket listenfd a um valor de socket local, associando assim o IP e porta definidos na estrutura. No entanto, a porta pode ser definida de forma automática. Recebe o descritor de socket, a estrutura e o tamanho.

e. listen

```
listen(int listenfd, int backlog);
```

Utilizada no servidor para transformar um socket no modo passivo, de forma que agora espera por conexões dos clientes. Recebe o descritor de arquivo do socket no servidor, e o inteiro backlog que representa o tamanho máximo da fila de conexões pendentes ou completadas que esse socket pode ter.

f. accept

```
accept(int listenfd, struct sockaddr *addr, socklen_t *addrlen)
```

Utilizada pelo servidor para aceitar uma conexão do cliente. Esta função um pedido de conexão da lista de conexões do socket de escuta (listenfd), extrai informações para criar um novo socket conectado e associado a um descritor de socket retornado por esta função, para que finalmente pode acontecer a transferência de dados entre cliente e servidor.

g. read e write

```
write(connfd, mssg, sizeof(mssg))
```

```
read(connfd, recvline, MAXLINE)
```

Utilizada pelo cliente e servidor, dada uma conexão ativa associada ao descritor connfd, para transferência de mensagens através de um socket de conexão.

h. close

```
close(connfd)
```

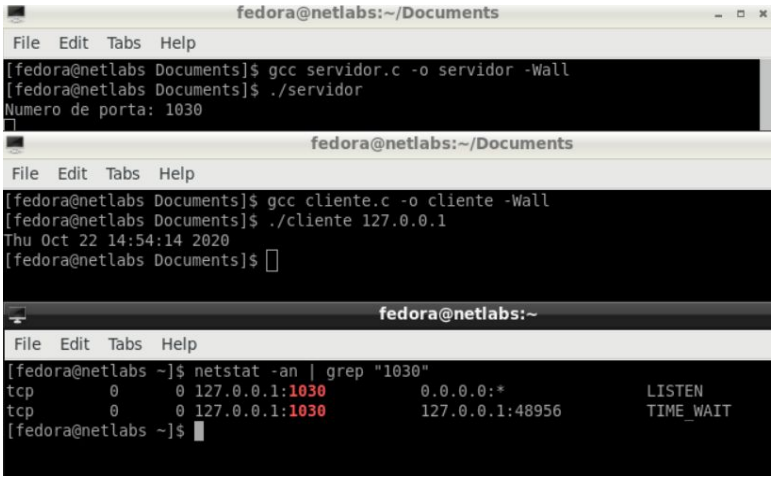
Utilizada para fechar a conexão associada ao descritor de socket connfd.

2. Na compilação dos arquivos, o servidor.c apresentou erros de conversão e de tipo de variável, na linha:

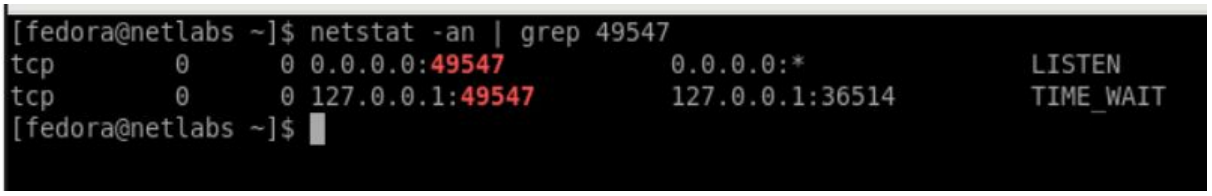
servaddr.sin_addr.s_addr = htonl("192.168.0.16");

Isso ocorreu porque a função htonl espera um unsigned integer longo e recebeu uma string. Logo, utilizei a função inet_addr no lugar, visto que esta função recebe um endereço IP no formato string e converte para endereço em bytes da Internet. Além disso, não consegui utilizar este IP, na função bind apareceu “cannot assign requested address”. Então, mudei por enquanto para 127.0.0.1 e funcionou.

Além disso, tanto servidor quanto cliente apresentarem erros na função bind, pois a porta era igual a 13. Então, mudei para 1030.

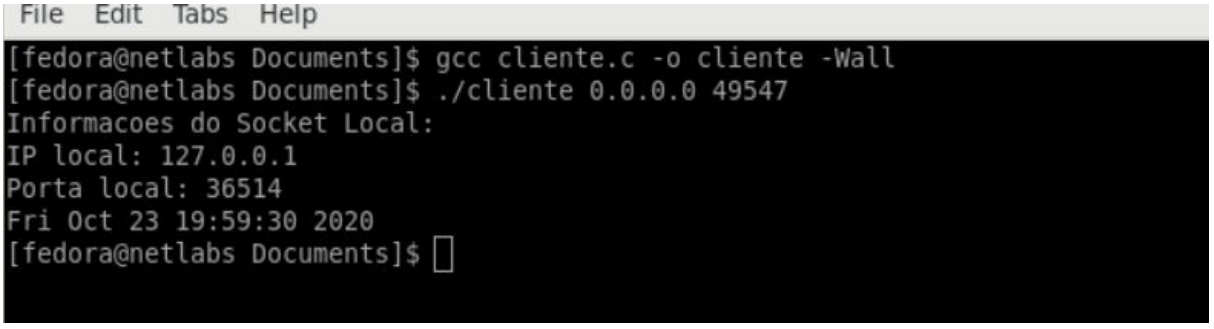


3. Troquei o IP para ser escolhido utilizando a constante INADDR_ANY e a porta como 0 para que seja escolhida de forma automática. Utilizei o getsockname no servidor, para descobrir a porta escolhida e então printar. Assim, o cliente pode enviar esta porta como parâmetro na criação do socket remoto. O IP sempre vem como 0.0.0.0 pois significa que todas as interfaces de rede estão escutando. Na figura, o número de porta que o servidor mostrou foi 49547.

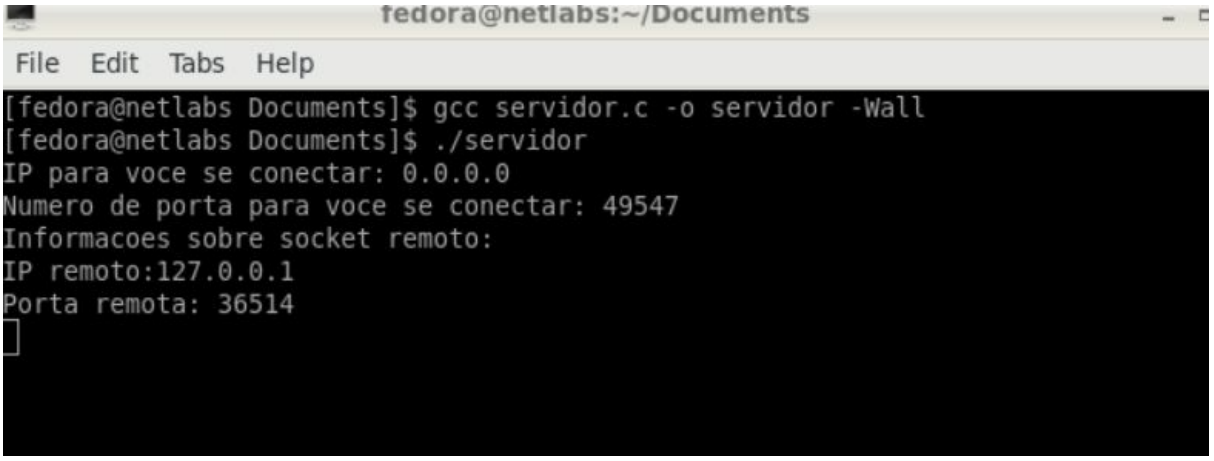


4. As chamadas de sistema necessárias para o servidor escutar conexões futuras são: *socket, bind, listen*. Feito isso, a lista de conexões já é criada associada a um descritor de socket.

6. Depois que a conexão foi aberta, utilizei getsockname para obter o IP e porta local em cliente.c.



7. Depois que o servidor aceitou a conexão, utilizei getpeername para obter informações de porta e ip do socket remoto: no caso o cliente mostrado na questão 6.



8. Sim, o executável de servidor.c nos retorna a porta que deve ser utilizada. Assim, com o comando: telnet <ip> <port> podemos substituir os valores pelo ip local, que é o mesmo do servidor e a porta fornecida. Note que obtivemos a resposta da mesma forma.

Como executar os arquivos:

1. No servidor, compile normalmente servidor.c
2. Depois rode o executável: ./servidor e veja qual porta o cliente pode se conectar
3. Depois, em outro prompt de comando, compile cliente.c
4. Rode o executável de cliente da seguinte forma:

   ./cliente 127.0.0.1 <numero de porta fornecido no passo 2>

Não informei o endereço de IP do servidor, pois mudamos para ser sempre o local.