MC833 - Tarefa 3

Erik de Godoy Perillo - RA: 135582

7 de abril de 2016

1. A função `connect` estabelece uma conexão entre um *socket* especificado e um host, cujas características (como família de endereços a ser usada, endereço em si, porta) são especificadas também (na implementação em C para Linux, por exemplo é usado o apontador para uma `struct sockaddr`. O tipo de conexão (protocolo a ser usado) varia de acordo com o socket especificado.

2. –

   O teste foi executado primeiro rodando o servidor e depois o cliente.

   Lado cliente (somente há entrada):

   ```
   $ ./client localhost
   ey b0ss
   ey b0ss?
   follow the white rabbit.
   ```

   Lado servidor (somente há saída):

   ```
   $ ./server
   ey b0ss
   ey b0ss?
   follow the white rabbit.
   ```

3. Usando-se `netstat` enquanto está havendo a comunicação, pode-se comprovar (observe o nome do processo referente à porta que estamos usando, 31472):

```
$ netstat -ap
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address Foreign Address State
PID/Program name
tcp 0 0 *:31472 *:* LISTEN 11423/./server
tcp 0 0 *:ssh *:* LISTEN
tcp 0 0 localhost.localdom:6342 *:* LISTEN 623/megasync
```

Usando-se `tcpdump`, também podemos checar. O `tcpdump` foi iniciado com a conexão já iniciada e, após uma mensagem ser enviada pelo cliente, houve saída pelo comando.

```
$ sudo tcpdump -i any -vvv "port 31472"
tcpdump:  listening on any, link-type LINUX_SLL (Linux cooked),
capture size 262144 bytes
12:23:35.652932 IP (tos 0x0, ttl 64, id 51944, offset 0, flags
[DF], proto TCP (6), length 60)
localhost.localdomain.40918 > localhost.localdomain.31472:
Flags [S], cksum 0xfe30 (incorrect -> 0xc0ee), seq 3165645406,
win 43690, options [mss 65495,sackOK,TS val 28647606 ecr
0,nop,wscale 7], length 0
12:23:35.652974 IP (tos 0x0, ttl 64, id 0, offset 0, flags
[DF], proto TCP (6), length 60)
localhost.localdomain.31472 > localhost.localdomain.40918:
Flags [S.], cksum 0xfe30 (incorrect -> 0x5343), seq 1737876377,
ack 3165645407, win 43690, options [mss 65495,sackOK,TS val
28647606 ecr 28647606,nop,wscale 7], length 0
12:23:35.653010 IP (tos 0x0, ttl 64, id 51945, offset 0, flags
[DF], proto TCP (6), length 52)
localhost.localdomain.40918 > localhost.localdomain.31472:
Flags [.], cksum 0xfe28 (incorrect -> 0x2588), seq 1, ack 1,
win 342, options [nop,nop,TS val 28647606 ecr 28647606], length
0
```

4. Sim, é possível. Isso porque, no protocolo telnet, o cliente apenas pega a entrada do usuário e a manda para o servidor, por meio de uma conexão TCP. Assim, como nosso servidor está preparado para lidar com as mensagens enviadas, tudo funciona. Assim, basta especificar a porta.

   Lado do cliente (telnet):

```
$ telnet localhost 31472
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
oie
fascinante, legal mesmo, estou impressionado.
```

Lado do servidor:

```
$ ./server
oie
fascinante, legal mesmo, estou impressionado.
```

5. Este foi resolvido no Susy. :^). Um exemplo da brincadeira:

   Lado servidor:

```
$ ./server
oi
testeee
echooooooo
uau!!!!!!!!!!!!!!!!!
```

   Lado cliente:

```
$ ./client
oie
oie
testeee
testeee
echooooooo
echooooooo
uau!!!!!!!!!!!!!!!!!
uau!!!!!!!!!!!!!!!!!
```