

Exercício 3

Raissa Cavalcante Correia – 150619

Allana M. Idalgo - 145166

1. Explique o funcionamento das funções `inet_pton`, `htons` e `htonl`. Existe alguma diferença entre elas? Explique.

`inet_pton`: converte um endereço web no formato texto padrão para a forma binária.

As funções `htons` e `htonl` convertem e retornam (o valor na ordem de bytes da rede) um o endereço passado como parâmetro para um ordenamento de byte significativo.

Sim, a `htons` converte um *hostshort* e a `htonl` converte um *hostlong*.

2. Compile e execute os programas [cliente.c](#) e [servidor.c](#) em uma mesma máquina. A saída do programa deverá ser a data atual, por exemplo, **Sun Sep 3 20:54:06 2018. Caso não consiga gerar a saída indicada, corrija e informe em detalhes quais foram as correções e porque você as fez.**

Modificações realizadas:

No cliente: linha 33 passou a ser “`servaddr.sin_port = htons(8000);`”

linha 36 passou a ser: “`connect(sockfd, (struct sockaddr *)&servaddr, sizeof(servaddr));`”

Na execução a data é obtida com o comando “`./cliente 127.0.0.1`”

3. Através de ferramentas existentes no sistema operacional, como você comprova, durante a execução em máquinas diferentes, que os algoritmos estão realizando uma comunicação via rede? Comprove sua resposta.

Podemos comprovar através da ferramenta e comando `netstat`. Como você pode observar abaixo:

```

Activities Terminal
File Edit View Search Terminal Help
Proto Destination Source State Channel
bash-4.4$ netstat
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address Foreign Address State
tcp 0 0 0 rabugento.lab.ic.:32934 grul0s10-in-f14.1e:htp ESTABLISHED
tcp 0 0 0 rabugento.lab.ic.un:885 cebolinha.lab.ic.un:nfs ESTABLISHED
tcp 0 0 0 rabugento.lab.ic.:46010 grul0s10-in-f14.1:https TIME WAIT
tcp 0 0 0 rabugento.lab.ic.:48740 mingau.lab.ic.unic:ldap ESTABLISHED
tcp 0 0 0 rabugento.lab.ic.:46012 grul0s10-in-f14.1:https TIME WAIT
tcp 0 0 0 rabugento.lab.ic.:35124 stackoverflow.com:https ESTABLISHED
tcp 0 0 0 rabugento.lab.ic.:42820 grul0s03-in-f202.:https ESTABLISHED
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags Type State I-Node Path
unix 3 [ ] DGRAM 14337 /run/systemd/notify
unix 7 [ ] DGRAM 14356 /run/systemd/journal/socket
unix 28 [ ] DGRAM 14365 /run/systemd/journal/dev-log
unix 2 [ ] DGRAM 22930 /var/run/chrony/chronyd.sock
unix 2 [ ] DGRAM 28653 /run/user/19666/systemd/notify
unix 2 [ ] STREAM CONNECTED 56877
unix 3 [ ] STREAM CONNECTED 44220
unix 3 [ ] STREAM CONNECTED 35866 /run/systemd/journal/stdout
unix 3 [ ] STREAM CONNECTED 34913 /run/user/19666/bus
unix 3 [ ] STREAM CONNECTED 33207 /run/user/19666/bus
unix 2 [ ] STREAM CONNECTED 30548
unix 3 [ ] STREAM CONNECTED 37433 /run/dbus/system_bus_socket
unix 3 [ ] STREAM CONNECTED 96504 @/dbus-vfs-daemon/socket-P204gg9a
unix 3 [ ] STREAM CONNECTED 41679 /run/user/19666/bus
unix 3 [ ] STREAM CONNECTED 33331
unix 2 [ ] DGRAM 32869
unix 3 [ ] STREAM CONNECTED 49792
unix 3 [ ] STREAM CONNECTED 44213
unix 3 [ ] STREAM CONNECTED 33930
unix 3 [ ] STREAM CONNECTED 33170 /run/systemd/journal/stdout
unix 3 [ ] STREAM CONNECTED 27171
unix 3 [ ] STREAM CONNECTED 20699 /run/dbus/system_bus_socket
unix 3 [ ] STREAM CONNECTED 39155 /run/user/19666/bus
unix 3 [ ] STREAM CONNECTED 36884
unix 3 [ ] STREAM CONNECTED 31637
unix 3 [ ] STREAM CONNECTED 32414 /run/user/19666/bus
unix 3 [ ] STREAM CONNECTED 29578 /run/user/19666/bus
unix 3 [ ] STREAM CONNECTED 32981 /run/dbus/system_bus_socket
unix 2 [ ] STREAM CONNECTED 103053
unix 2 [ ] DGRAM 52862
unix 3 [ ] STREAM CONNECTED 42186
unix 3 [ ] STREAM CONNECTED 32877
unix 3 [ ] STREAM CONNECTED 28190 /run/systemd/journal/stdout
unix 3 [ ] STREAM CONNECTED 39156 /run/user/19666/bus
unix 3 [ ] STREAM CONNECTED 32637
unix 3 [ ] STREAM CONNECTED 37902 /run/user/19666/bus
unix 3 [ ] STREAM CONNECTED 35135 /run/systemd/journal/stdout
unix 3 [ ] STREAM CONNECTED 32329 /run/systemd/journal/stdout
unix 3 [ ] STREAM CONNECTED 23457
unix 3 [ ] STREAM CONNECTED 42183
unix 3 [ ] SEQPACKET CONNECTED 31803 /run/user/19666/bus
unix 3 [ ] STREAM CONNECTED 48574 /run/systemd/journal/stdout
unix 3 [ ] STREAM CONNECTED 39158 /run/dbus/system_bus_socket
unix 3 [ ] STREAM CONNECTED 33929
unix 3 [ ] STREAM CONNECTED 32199 /run/user/19666/bus

```

4. Modifique o programa `cliente.c` para que ele obtenha as informações do socket local (# IP, # porta local) através da função `getsockname()`. Modifique o programa `servidor.c` para que este obtenha as informações do socket remoto do cliente (# IP remoto, # porta remota), utilizando a função `getpeername()`. Imprima esses valores na saída padrão.

```

Terminal
File Edit View Search Terminal Help
/isi cliente.c:50:32: note: each undeclared identifier is reported only once for each
function it appears in
bash-4.4$ gcc cliente.c -o cliente
bash-4.4$ ./cliente 127.0.0.1
Local ip address: 127.0.0.1
Local port : 36336
packet send donebash-4.4$ ./cliente 127.0.0.1
Local ip address: 127.0.0.1
Local port : 36342
packet send donebash-4.4$ ./cliente 127.0.0.1
Local ip address: 127.0.0.1
Local port : 36344
packet send doneMon Sep 17 21:32:18 2018
bash-4.4$ ./cliente 127.0.0.1
Local ip address: 127.0.0.1
Local port : 36440
packet send doneMon Sep 17 21:34:12 2018
IP bash-4.4$ gcc cliente.c -o cliente
bash-4.4$ ./cliente 127.0.0.1
Local ip address: 127.0.0.1
Local port : 36442
Mon Sep 17 21:35:11 2018
bash-4.4$

Terminal
File Edit View Search Terminal Help
printf("Peer IP address: %s\n", inet_ntoa(addr.sin_addr));
bash-4.4$ ./servidor
Segmentation fault (core dumped)
bash-4.4$ gcc servidor.c -o servidor
servidor.c: In function 'main':
servidor.c:50:37: warning: implicit declaration of function 'inet_ntoa' [-Wimpli
cit-function-declaration]
printf("Peer IP address: %s\n", inet_ntoa(addr.sin_addr));
bash-4.4$ ./servidor
Segmentation fault (core dumped)
bash-4.4$ gcc servidor.c -o servidor
bash-4.4$ ./servidor
Peer port : 0
^C
bash-4.4$ gcc servidor.c -o servidor
bash-4.4$ gcc servidor.c -o servidor
bash-4.4$ ./servidor
Peer IP address: 0.0.0.0
Peer port : 0
Peer IP address: 0.0.0.0
Peer port : 0

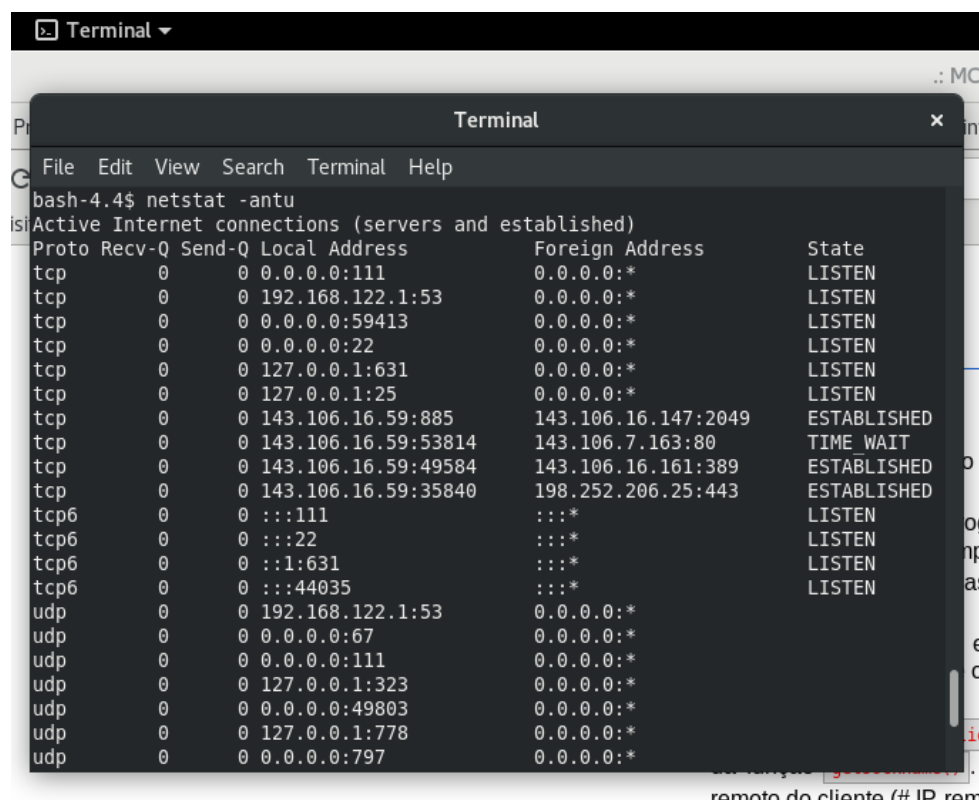
```

5. Modifique o cliente e o servidor para que o cliente envie 1 (um) caractere ao servidor, e o servidor na sequência deverá retornar esse caractere ao cliente.

Vide código em anexo.

6. Mantenha o binário do servidor .c executando em uma máquina A e execute três vezes seguidas o binário do cliente .c em uma máquina B. Observando a saída do comando `netstat`, alguma máquina ficou no estado `TIME_WAIT`? Se, sim, qual? Explique os motivos para este estado ocorrer.

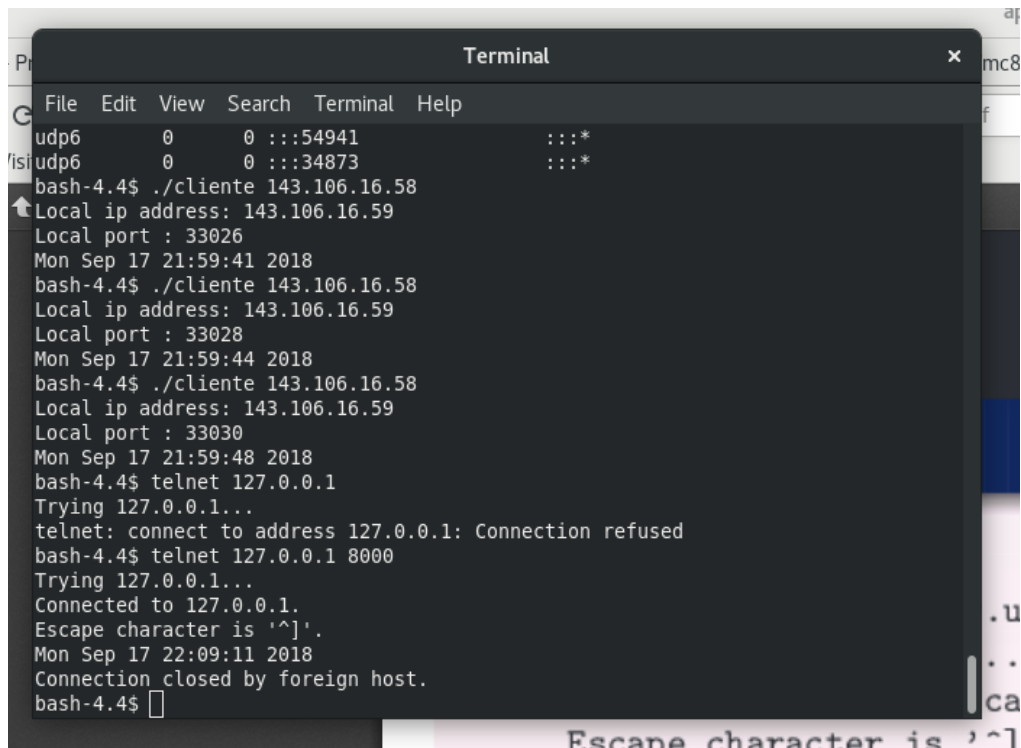
Sim. Pois, o servidor TCP implementado não aceita conexões concorrentes.



```
Terminal
File Edit View Search Terminal Help
bash-4.4$ netstat -antu
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:111             0.0.0.0:*               LISTEN
tcp        0      0 192.168.122.1:53        0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:59413           0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN
tcp        0      0 143.106.16.59:885       143.106.16.147:2049     ESTABLISHED
tcp        0      0 143.106.16.59:53814     143.106.7.163:80       TIME_WAIT
tcp        0      0 143.106.16.59:49584     143.106.16.161:389     ESTABLISHED
tcp        0      0 143.106.16.59:35840     198.252.206.25:443     ESTABLISHED
tcp6       0      0 :::111                  :::*                    LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
tcp6       0      0 :::1:631                :::*                    LISTEN
tcp6       0      0 :::44035                 :::*                    LISTEN
udp        0      0 192.168.122.1:53        0.0.0.0:*               *
udp        0      0 0.0.0.0:67              0.0.0.0:*               *
udp        0      0 0.0.0.0:111             0.0.0.0:*               *
udp        0      0 127.0.0.1:323           0.0.0.0:*               *
udp        0      0 0.0.0.0:49803           0.0.0.0:*               *
udp        0      0 127.0.0.1:778           0.0.0.0:*               *
udp        0      0 0.0.0.0:797             0.0.0.0:*               *
```

7. É correto afirmar que o programa `telnet` pode ser usado no lugar do binário do `client.c`? Explique e comprove caso seja verdadeira. Cite uma modificação no `servidor.c` que impediria a utilização do `telnet`

Sim, pois ele conecta a um servidor TCP, tal qual o `cliente.c`.

A terminal window titled "Terminal" with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the following output:

```
udp6      0      0 :::54941      :::*
udp6      0      0 :::34873      :::*
bash-4.4$ ./cliente 143.106.16.58
Local ip address: 143.106.16.59
Local port : 33026
Mon Sep 17 21:59:41 2018
bash-4.4$ ./cliente 143.106.16.58
Local ip address: 143.106.16.59
Local port : 33028
Mon Sep 17 21:59:44 2018
bash-4.4$ ./cliente 143.106.16.58
Local ip address: 143.106.16.59
Local port : 33030
Mon Sep 17 21:59:48 2018
bash-4.4$ telnet 127.0.0.1
Trying 127.0.0.1...
telnet: connect to address 127.0.0.1: Connection refused
bash-4.4$ telnet 127.0.0.1 8000
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^['.
Mon Sep 17 22:09:11 2018
Connection closed by foreign host.
bash-4.4$
```

Uma modificação no servidor.c seria torná-lo um servidor UDP, uma vez que o telnet apenas atua sob o protocolo TCP.

8. Agora que todas as alterações no código foram realizadas, adicione mensagens e verificações de erros, e comentários que sejam relevantes para uma melhor interpretação dos algoritmos.

Vide arquivos .c em anexo.