UNIVERSIDADE DE COIMBRA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES

Computer Networks

Practical Laboratory Work

Sockets Programming (TCP and UDP)

Name: ____________________________________ Date: ___/___/___

Partner: _____________________________ Class: ____

The objective of these experiments is to broaden your knowledge about sockets programming. In the first part you will handle a server program that will accept several requests from clients, using TCP. In the second part, the transport protocol UDP is used instead of TCP, and finally the application protocol has to be changed.

Materials used: the PCs of the Computer Networks Laboratory (with the Linux operating system) and the files made available at *inforestudante.uc.pt*.

Perform the following actions, indicating the result where required. Experiment 1 (TCP):

1- Open a console window (terminal, for command line interface).

2- Create the working directory "aulaRC" (inside your login directory):
```
$ mkdir  aulaRC
```

3- Copy the files *client_tcp.c* and *server2_tcp.c* to your working directory (aulaRC).
```
$ cp   client_tcp.c   aulaRC
$ cp   server2_tcp.c   aulaRC
```

4- Identify your host name:
```
$ hostname                 _________________________________
```

5- Identify the IP address of your host:
```
$ ifconfig                       ______.______.______.______
```

6- Change to your working directory (aulaRC):
```
$ cd   aulaRC
```

7- Compile both programs (*client_tcp.c* and *server2_tcp.c*), maintaining their names:
```
$ gcc  client_tcp.c    -o  client_tcp
$ gcc  server2_tcp.c  -o  server2_tcp
```

8- By inspection of the source code, indicate what the client program does (from the point of view of the user and in the absence of errors):

1.____

2.____

3.____

9- By inspection of the source code, indicate what the server program does with each client (from the point of view of the user and in the absence of errors):

1.____

2.____

10- Let's start by executing the two programs in the same host, in different console windows. Use port 50000 or another that is free. Start by executing the server program. Check that it remains waiting (for a cliente request):

```
$ ./server2_tcp  50000
```

11- Open a new console window in order to enable having both programs in execution simultaneously. Execute the client program in that new console:

```
$ ./client_tcp  localhost  50000
```

12- Message introduced in the client console window: ______________________

13- Message written by the server in the respective console window: ______________________

14- Message written by the client in the respective console window: ______________________

15- Has the server program finished? Yes ☐ No ☐

16- Without launching a new server program, execute the client program again (may be in the same console window used before), indicating the name of your host instead of "localhost". If it does not operate with the name of your host, use its IP address.

```
$ ./client_tcp  hostname  50000
```

17- Does the server program continue operating? Yes ☐ No ☐ Obs.: ____________________

18- Open a new console window in order to enable having two client programs operating simultaneously. Launch the two programs in both console windows (for the same port).

```
$ ./client_tcp  localhost  50000
```

19- Did the two client programs operate simultaneously?
Yes ☐ No ☐ Obs.: ____

20- Message introduced in the first client console window: ______________________________

21- Message written by the server in the respective console window: ______________________

22- Message introduced in the second client console window: ___________________________

23- Message written by the server in the respective console window: ______________________

Perform the following actions, indicating the result where required. Experiment 2 (UDP):

24- Copy the files *client_udp.c* and *server_udp.c* to your working directory (aulaRC):

25- Compile both programs (*client_udp.c* and *server_udp.c*), maintaining their names:
```
$ gcc   client_udp.c   -o  client_udp
$ gcc   server_udp.c   -o  server_udp
```

26- By inspection of the source code, tell what the client program does (from the point of view of the user and in the absence of errors):

1.____

2.____

3.____

27- By inspection of the source code, tell what the server program does with each client (from the point of view of the user and in the absence of errors):

1.____

2.____

28- Let’s start by executing the two programs in the same host, in different console windows. Use port 50000 or another that is free. Start by executing the server program. Check that it remains waiting (for a cliente request).
```
$ ./server_udp 50000
```

29- Open a new console window in order to enable having both programs in execution simultaneously. Execute the client program in that new console.
```
$ ./client_udp  hostname  50000
```

30- Message introduced in the client console window: __________________________

31- Message written by the server in the respective console window: _________________

32- Message written by the client in the respective console window: __________________

33- Has the server program finished? Yes ☐ No ☐

34- Open a new console window in order to enable having two client programs operating simultaneously. Launch the two programs in both console windows (for the same port).
```
$ ./client_udp  hostname  50000
```

35- Did the two client programs operate simultaneously?
Yes ☐ No ☐ Obs.: ____

36- Message introduced in the first client console window: _______________________

37- Message written by the server in the respective console window: _________________

38- Message introduced in the second client console window: _________________

39- Message written by the server in the respective console window: ______________________

40- Execute now the client program of the previous experiment (with TCP), maintaining the same server (with UDP).

```
$ ./client_tcp  hostname  50000
```

Did it operate well? Yes ☐ No ☐ Error message: _________________________________

Perform the following actions, indicating the result where required. Experiment 3 (UDP, changing the application protocol):

41- Continuing with UDP as the transport protocol, perform the necessary changes to make the server ask its user, initially, for a string (e.g. a place) and an integer, and send them, in two separate messages, to the clients, which show them in the monitor and terminate. Each client starts by asking its user for a string (e.g. user name) and then sends it to the server to enable synchronization and inform the server about the addresses (IP and port) of the client. The server should continue operating indefinitely, showing the clients' strings. Implement these changes using the functions *hton* and *ntoh* to send the integer to any type of host (*Little Endian* or *Big Endian*). Copy the original programs, changing their names to *server_udp_endian.c* and *client_udp_endian.c*

42- Draw a time-space diagram of the protocol:

```
server          client
  |               |
  |               |
  |               |
  |               |
  |               |
  |               |
```

43- Message introduced in the server program: ______________________

44- Message written by the client program in the respective console window: ________________

45- Integer introduced in the server program (e.g. birthday): ______________________

46- Integer written by the client program in the respective console window: _________________

47- Let's now change the <u>client program</u>, without using the function *ntoh* for reception of the integer.

48- Integer introduced in the server program (birthday): ______________________

49- Integer written by the client program in the respective console window: _________________

Did it perform well? Yes ☐ No ☐ Why? _________________________________

50- Before shutting down your computer, show this form and the code produced to the lecturer, to check for their correctness.