Distributed Programming I

Solution sketch of laboratory 1

Exercise 1.2 (connection)

Use the functions Socket() and Connect() after having parsed the port number and the address through the Getaddrinfo() function or, as alternatives, inet_addr() or inet_aton() functions. These two alternatives, however, are not suggested as they are not portable to IPv6.

Exercise 1.3 (ASCII data)

Proceed as in the previous exercise. Write date to the socket by using Write() or Send() functions and read data through Readline(), as example. Use the return value of an sscanf() to guess if the server response is the expected value or an error message, then print it in a proper way.

Exercise 1.4 (basic client-server UDP)

Use functions Socket(), SendTo() and Recvfrom(). Alternatively, it is possible to use the Connect() function to set the destination address once and then use the Send() function to send data.
There are two possible approaches to handle timeouts in waiting a response from the Server: using Select() or a SIGALARM signal.

- Select() solution: call it before reading from the socket, by setting its timeout to the desired value. As a first parameter it is possible to use a file descriptor corresponding to the socket fd value + 1.
- SIGALARM solution: register a function able to handle this signal through the sigaction function. Before calling Recvfrom_timeout(), set a timeout through the alarm() function. If no packet has been received until then, Recvfrom_timeout() will be interrupted by the signal if the last parameter has been set to 1: then the program will be able to terminate by printing the correct message, basing on the return value of Recvfrom_timeout().

For what concerns the UDP server, use the Socket() and Bind() functions to register, as example, to the address INADDR_ANY, then insert in an infinite loop the couple Recvfrom() and Sendto(). Pay attention to just send the valid part of the buffer read through the Recvfrom().