

Network Programming in C – Simple Server Example

Stage 1

Create two simple applications `client.c` and `server.c`, where the server will start and await a connexion from the client. The server should give a message that it is listening after it has started, and when the client tries to connect it should give a success or failure message before terminating.

Although the two programs should communicate over the network, for simplicity of development you can run them on the same physical computer.

You should refer to the information on network programming in Workbook 3 (Network Programming).

Remember that to get the two programs exchanging data, they need to BLAB:

BLAB – **B**ind, **L**isten, **A**cept, **B**egin processing.

As necessary, you can refer to the baseline solution in the folder `simple_server` to see a basic working example.

Note that if you stop and restart the server in quick succession, sometimes it may not restart correctly. This is because the operating system has not yet released the port bound to the socket created. (The problem can be avoided simply by waiting a few seconds). The sample code has this problem – find out how the issue can be circumvented.

Stage 2


Update your solution so that a server will send some fixed message (such as “hello there client!”) to the client when it connects to the server; the client should print out the message it receives. You will need to use the `send` and `recv` functions.

Stage 3

Create a simple Spanish translation server that will accept a word in English, and reply with a Spanish translation if the word is known, or an error message if the word is unknown. Note that the English word should be input and the Spanish word presented in the client application, but the translation should happen in the server application.

For the first version we will keep the translation very simple; you can use the two arrays of matching words in the file `words.h` as the basis of the translation.

The screenshot below shows a session of the translation server at work:



The screenshot shows a PuTTY terminal window titled "soc-web-liv-11.napier.ac.uk - PuTTY". The terminal displays a session with a translation server. The prompt is "40011071@soc-web-liv-11:~/netcode/example\$./client". The server prompts the user to "enter English word:". The user enters "work", and the server responds with "work --> trabajo". The user enters "fact", and the server responds with "fact --> hecho". The user enters "snordlegipp", and the server responds with "snordlegipp --> unknown / desconocida". The user enters "game", and the server responds with "game --> juego". The user enters another word, and the server prompts "enter English word:" with a green cursor.

```
40011071@soc-web-liv-11:~/netcode/example$ ./client
enter English word:    work
    work --> trabajo
enter English word:    fact
    fact --> hecho
enter English word:    snordlegipp
    snordlegipp --> unknown / desconocida
enter English word:    game
    game --> juego
enter English word:    █
```

Optional Exercise One : Improve the functionality of the server by allowing it to read the dictionary from a text file which can be separately authored to increase its vocabulary.

Optional Exercise Two : Update the server so that it can handle multiple clients connecting concurrently and serve them with separate translations.