Purpose: This assignment reviews the select and emphasized the topics covered in lecture 13b. You will build a (multi-user) chat client. Your client will connect to the concurrent connection-oriented single process server you developed in the previous assignment.

THE CHAT CLIENT

When started, the client will open a TCP connection to the server. Whatever the client sees the user type will be sent to the server (i.e., the client gets a line from the keyboard, then sends that line to the server). Whatever the client receives from the server will be printed on the screen (i.e., the client receives from the socket, then prints what it has received to the screen). Typing control-D (EOF from keyboard) should cause the client to shutdown the socket for further writes, and reads from the socket until it gets an end of file from the server, then exit. A good idea on any EOF (or error) is to close the descriptor and clear the bit from the descriptor set. In this case it is the keyboard that is being closed.

If at any time you see an EOF or error on the socket, exit; you are a client, if there is no server to talk to, you are done.

Since input comes from both the keyboard and the socket your client must use select.

Timeout requirement: You must add a timeout to your select. If no input has been received for 5 seconds, the select should timeout. In the case of a timeout the client should print to the screen the message "Anybody there?". It should then loop back to the select to wait some more.

Submit: The a fully commented copy (print-out) of the source code for the client. In addition, the source code for the client must be placed in your home directory in a file named `chatc.c`.

Do not submit a copy of your the server, you did that for the previous assignment.

Note 1: We don't really test the use of the shutdown; I'll have to check the source code by hand to make sure you used it.

Note 2: The emphasis not on the user interface. Use the same user interface (basically "none") found in my client that you used to test your code in the previous assigment.

Note 3: Remember that `fgets` returns `NULL` on end of input from the keyboard.

Testing: I have put up a compiled version of the server to assist you in testing the socket shutdown code. After you have done the shutdown the server will send a final "Good Bye" message to the client. If you don't see that message, you haven't handled used the shutdown correctly. This version of the chat server is available as `shells/chatdX`.