**DUBLIN INSTITUTE OF TECHNOLOGY**

###### KEVIN STREET, DUBLIN 8.

**client server programming (CMPU3006)**

Mr. D. Bourke

Stage 3

**CONTINUOUS ASSESSMENT 2018/19**

Tuesday, 5th November 2018, 12:25 – 13:55

This assessment accounts for 50% of the overall CA mark for this module.

.



Figure 1: The Client Code for a Basic User Authentication System (double-tap icon to open).

Figure 1 reveals the client code for a basic ***User Authentication*** system. You are required to write the **server** component of this systemin *C* using the Berkeley Socket API.

Name the source file “authServer.c” and **include your name and student number in the first line of this file as a comment**.

There are two files to upload to Webcourses: “*authServer.c”* and “*test.txt”* (see how to create this file later).

Please refer to the “Lab Test Instruction Sheets” previously issued for instructions on how to do this.

Study the client code and determine the data exchange requirements for this system i.e. the Syntax and Semantics. The following is the minimum functionality required of the server:

* The basic operation of this application is that the client makes a connection request to the server.
* In normal operation the client will request text input from the local user
* The client constructs a string containing a text string containing the “username” and “password” and sends the string to the server.
* The **server** application should read the incoming data and be able to separate-out (parse) the “username” and “password” components into separate buffers.
* The **server** application should verify these components against the strings containing the username “admin” and password “pass”
* It should return **one** of the following messages for each submission by the client i.e. each attempt:
  + - PROCEED
    - You have x attempt(s) left
    - DENIED

The DENIED response should only be sent after **three** unsuccessful attempts.

If PROCEED or DENIED is returned to the client, close the connection and wait for the next connection. Otherwise return to reading the next attempt from the client.

Include the following lines of code into your application before closing the connection. This is to facilitate creating the *test.txt* file:

**printf(“Written: %s\n”, sendbuff);**

**fflush(stdout);**

Change the second argument to the correct identifier for the outgoing buffer you are using in your application.

**(100 Marks)**

Marks will be awarded for completeness, robustness and readability of the code so be sure to verify each step especially where data processing is expected, indent your code for readability and add comments.

**Notes on running your code for testing purposes:**

The client source code file from Figure 1 can be downloaded into the CSP folder. It should be compiled and used for testing against your server code. To download the file, use the following cp command (Note the spaces after cp and after \*.c):

cp /Student\_Distrib/DT228/DT228-3/CSP/LabAssign/\*.c ./

To test the *server* application it must be compiled using the following command (this assumes you have previously written the compile script and that both the script and the server source code file are in the *csp* directory):

**./compile authServer**

To run the server code from within the *csp* directory type:

**./authServer 1234**

To run the client code from within the *csp* directory type:

**./authClient 127.0.0.1 1234**

Replace the text **1234** in the above command strings with the port number you have chosen for your server.

When you are satisfied that the server application works correctly re-run the server code as follows to capture output from the server application:

**./authServer 1234 > test.txt**

Re-run the client as before and connect it to the server, testing for a variety of user names and passwords. This last step is important as it shows that the code works.

Ensure to upload the “authServer.c” **and the** “test.txt” files to Webcourses in accordance with the “Lab Test Instruction Sheets” previously issued.