CSCI 631 **Computer Assignment 2** **Spring 2015**

(30 points)

Book Search
(Iterative Server)

For this computer assignment, design two C programs for a client and server, using the IPv4 protocol, where the client reads a title of a book from stdin and sends it to a book server to get the name of the author(s) of the book. When a query comes to the server for the name of author(s), then the server searches its database to obtain this information. If the server finds the requested information in its database, then it sends the located information to the client and the client prints it out on stdout. If the search becomes unsuccessful, then the server sends the author's name as "unknown".

The usage of the client is: client server-address [port-number], where server-address and port-number are the IP address and port number of the server. The usage of the server is: server [port-number]. The port number is optional and if it's not supplied, then the default port number SERV_PORT should be chosen. A valid (ephemeral) port number should be between 9880 and 9890, where the client runs on the *tiger* machine and the server on the *lambda* machine.

The server keeps its database in file: /home/courses/631/common/books.d. The information for a book in the database is stored in a separate line in the following format: title of the book:author(s) of the book: You note that the two attributes of a book are not terminated by white spaces but by colons ':', since the title and author(s) of a book may contain spaces. You note that books in database are not kept in a particular order.

The client may have multiple queries. For each query, the server starts searching its database from the beginning. Use either the function fseek ( ) or rewind ( ) to set the file-position indicator to the beginning of the input file. <u>You are not allowed to copy the contents of the input file to a data structure or to modify its contents</u>.

Use a wrapper function for each of the system calls that you use in your program. In case of an error, execute either the function perror ( ) or fprintf ( ). If a socket function, such as connect ( ), bind ( ), etc., generates an error, do not forget to close the socket before executing the exit ( ) function.

Name the source file of your client as prog2_client.c, the source file of your server as prog2_server.c, and the source file of your wrapper functions as wrapper.c; however, you can give them any names you like, but make it sure that name of a source file should clearly identify the contents of the file. In your program, use the I/O library routines, such as readline ( ) and writen ( ), instead of the system calls read ( ) and write ( ). <u>Since these I/O routines are alternatives for read ( ) and write ( ), do not forget to include the wrappers for them</u>. To use these functions in your programs, simply link the object file in_out.o with the rest of your object files, and to get the prototypes of these functions, include the following statement in your header file for your wrapper functions: #include "/home/courses/631/common/in_out.h".

When your program is ready, login both the *tiger* (1st machine) and *lambda* (2nd machine) on separate windows. Then, do the followings:

1. On the 2nd machine, make a link to the script P2_2: ln –s ~courses/631/bin/P2_2 and start the script utility: script prog2_2.out. When you are in the script utility, execute the command: P2_2 prog2_server.exe. This will execute the server in the background. When the script asks for a port number, enter a port number from the available pool of port numbers or just enter for the default port number. That port number will be displayed on the screen to be used for the client on the 1st machine. Then, the script will ask you to enter from the keyboard to continue, but do not enter anything at this point.

2. On the 1st machine, make a link to the script P2_1: ln –s ~courses/631/bin/P2_1 and start the script utility: script prog2_1.out. When you are in the script utility, execute the command: P2_1 prog2_client.exe. This will execute your client in the foreground and automatically enters the following four book titles from the keyboard in the following order: "Introduction to C", "introduction to c++", "LOCAL COMPUTER NETWORKS", and "The Theory of Numbers". At the completion of data entry, the script P2_1 will list all your active processes. Terminate the script prog2_1.out by the control character <ctrl>-D.

3. To resume the operation on the 2nd machine, just hit the enter key from the keyboard. Then the script P2_2 will display all active processes on the system, and after terminating the server program, it will display all active processes one more time. Terminate the script prog2_2 by the control character <ctrl>–D.

4. Just before you logout from either the 1st or the 2nd machine, execute the command: /bin/ps –f –u $USER. This will display a list of all processes generated by your still active processes – some of those processes might even be created in one your previous terminal sessions. In that list, if you see any processes except your login shells, terminate all those processes by the kill command: kill pid … pidn, where pid … pidn is the list of process ids of all processes that you want to terminate.

The correct outputs for this assignment can be found in files prog2_1.out (on the 1st machine) and prog2_2.out (on the 2nd machine) in directory: ~courses/631/progs/p2.

For the client program, you need wrappers for the following functions: close ( ), connect ( ), fgets ( ), fputs ( ), inet_pton ( ), socket ( ), readline ( ), and writen ( ); and for the server program, you need wrappers for the following functions: accept ( ), bind ( ), close ( ), fclose ( ), fgets ( ), fopen ( ), fseek ( ), listen ( ), socket ( ), readline ( ), and writen ( ).

When your program is ready, on the 1st system: submit the source/header files for your client on the 1st machine, and the source/header files for your server on the 2nd machine.

You need to have proper documentation in all your files that you submit for grading.