CSCI 631 **Computer Assignment 4** **Spring 2015**

(30 points)

Book Search
(For TCP & UDP Clients)

For this computer assignment, design a C program for a book search, using the IPv4 protocol, where a TCP client or a UDP client reads a title of a book from stdin and sends it to the TCP/UDP 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), the server searches its database to obtain this information. If the server finds the requested information in its database, it sends the located information to the client. If the search becomes unsuccessful, the server sends the author's name as "unknown".

You need to implement two different client programs for this assignment, one of them is for the TCP protocol and the other one is for the UDP protocol, and the server program needs to handle both types of clients simultaneously, using the I/O multiplexing model by calling the select ( ) system call. Use your client program from an earlier assignment for the TCP client, and modify the first client program from Chapter 8 of your textbook or the course website for the UDP client, and modify the first server program from Chapter 8 of your textbook or the course website for the TCP/UDP server. Both clients accept an optional port number with the default value SERV_PORT (as defined in the header file 631.h), and their usage is: client server–address [port–number], where IP–address and port–number are the address and the 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 is chosen. A valid (ephemeral) port number should be between 9880 and 9890.

Implement the TCP segment of the server as a *concurrent server* – one can handle multiple clients simultaneously. This can be done using the fork ( ) system call, creating a child process for each client, but implement the UDP segment of the server as an *iterative server*, which can handle one datagram at a time.

The book server keeps its database in file: /home/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 also note that books in the input file are not kept in a particular order.

A 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 TCP client program as prog4_tcpclient.c, the source file of your UDP client as prog4_udpclient.c, and the source file of your server program as prog4_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.

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

1. On the 2nd machine, make a link to the script P4_2: ln –s ~631/bin/P4_2 and start the script utility: script prog4_2.out. When you are in the script utility, execute the command: P4_2 prog4_udpclient.exe prog4_server.exe. This will execute your server in the background and your UDP client in the foreground and automatically enters the following four book titles in the following order: "1985", "1984", "Hobbit", and "Frankenstein". AT THIS POINT, DO NOT CONTINUE TO EXECUTE THE REST OF THE STATEMENTS IN THE SCRIPT.

2. On the 1st machine, make a link to the script P4_1: ln –s ~631/bin/P4_1 and start the script utility: script prog4_1.out. When you are in the script utility, execute the command: P4_1 prog4_tcpclient.exe. This will execute your TCP client in the foreground and automatically enters the following four book titles in the following order: "Introduction to C", "introduction to c++", "LOCAL COMPUTER NETWORKS", and "The Theory of Numbers". At the completion of the data entry, the script P4_1 will list all your active processes. Terminate the script utility prog4_1.out by the control character <ctrl>–D, and logout from the 1st machine.

3. Go back to the 2nd machine, and enter to continue. Then the script P4_2 will list all your active processes, and after terminating the server, it will list all your active processes one more time. Finally, terminate your script utility prog4_2.out 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 the list of all processes generated by your still active programs – 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.

Proper documentation is required for all your source and header files that you submit for grading, and to eliminate repetitions in your files, you should consider of using extra subroutines.

The correct outputs for this assignment can be found in files prog4_1.out (on the 1st machine) and prog4_2.out (on the 2nd machine) in directory: ~631/progs/p4.

When your program is ready, submit your source/header files for the TCP client to the 1st machine; and your source/header files for the UDP client and TCP/UDP server to the 2nd machine.