CSCI 631 **Computer Assignment 3** **Spring 2015**

(30 points)

Book Search (Concurrent Server)

Modify your server program from your previous assignment so that now the server becomes 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.

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 program as prog3_client.c, the source file of your server program as prog3_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 system) and *lambda* (2nd system) machines on separate windows. Then, do the followings:

1. On the 2nd system, make a link to the script P3_2: ln –s ~631/bin/P3_2 and start the script utility: script prog3_2.out. When you are in the script utility, execute the command: P3_2 prog3_client.exe prog3_server.exe. This will execute the server in the background and the client in the foreground. Enter the following four book titles from the keyboard in the following order: "1985", "1984", "Hobbit", and "Frankenstein". Each book title should be entered on a separate line.

2. On the 1st system, make a link to the script P3_1: ln –s ~631/bin/P3_1 and start the script utility: script prog3_1.out. When you are in the script utility, execute the command: P3_1 prog3_client.exe. This will execute the 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 P3_1 will list all your active processes. Terminate the script prog3_1.out by the control character <ctrl>-D, and logout from the 1st system.

3. On the 2nd system, terminate the data entry by the control character <ctrl>-D. At the completion of data entry, the script P3_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 by the control character <ctrl>-D.

4. Just before you logout from either the 1st or the 2nd system, execute the command: /bin/ps –f –u $USER. This will display a 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.

The correct outputs for this assignment can be found in files prog3_1.out (1st system) and prog3_2.out (2nd system) in directory: ~631/progs/p3.

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

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

Proper documentation is required for all of your source and header files that you submit for grading.