CSCI 311 / 312 Project 3
Group Assignment
Due: October 20, 2021
Points: 35

Purpose: The objective of this project is to write a client-server program that uses connection-oriented sockets.

Requirements:

This project is similar to exercise 10-8 on page 415 of the textbook: *Interprocess Communications in LINUX* by Gray. These exercises center around creating a client-server program to play a game (see below) with clients. The following requirements give additional details about what your implementation should look like.

1. Client-Server:

    a. *Server*: After the server process executes a fork system call to create a child process to handle a particular client connection, have the child process exec another program, ServerG, to perform the server tasks for playing the game. That is, the server will consist for two programs: ServerC and ServerG. ServerC is responsible for accepting the initial connection and establishing each child process based on the ServerG program. ServerC must pass the socket descriptor to ServerG. In addition, ServerC must continue to accept connections and monitor the termination of all child processes. The textbook: *TCP/IP Sockets in C* by Donahoo and Calvert in Section 6.4.1 contains relevant ideas. Appropriate console messages should be issued by ServerC to document the occurrence of significant events (new connection accepted, child created, child died, etc.).

    b. *Client*: The client is responsible for all communication with the human user who is playing the game. This includes such things as prompting for user input, displaying the status of the game, displaying the server's "move", displaying error messages, etc.

    c. *Shared code*: Create a C/C++-module that contains any common subroutines that are used by both the client and server. Link the resulting code into both the server and client programs. Client and server must utilize these routines. In addition, if there are any shared structure definitions or symbol definitions, define these in a .h file that both client and server use.

d. *System call usage*: Use each system call in an appropriate manner; read the man pages for the system calls you use.

2. *The Game*: Rock-Paper-Scissors as described below:

   Play best n out of m rounds of the game: Rock-Paper-Scissors. *n* and *m* are randomly selected to be: 2 out of 3, 3 out of 5, 4 out of 7, or 5 out of 9. In each round of this children's game, the two players simultaneously display symbols for: rock, paper or scissor. The winner is determined as follows:

   Paper beats rock, rock beats scissors, scissors beats paper; all other combinations are a draw.

3. *Error checking*: Provide appropriate error checking for each system call and take appropriate steps if an error is encountered.

4. *Use of sockets*: You must use Internet type sockets that are connection oriented.

5. *Test Environment*: Use the virtual machine: 199.17.28.80. Though I only expect you to test your program on one of the virtual machines; it should be possible to have client and server located on different machines and still have the programs communicate correctly.

6. *Man page*: create a man page that describes the usage of your programs; include a description of the selected game.

7. *Readability:* Your program must be written using good C/C++ programming conventions:
   - Variable names and function names should be descriptive of what they represent.
   - Use indentation to show the structure of the program. Typically this means using indentation with *for*, *while*, *do-while*, *if*, and *switch* statements as well as indenting the body of functions. Indentation should show the nesting level of the statements involved.
   - Include some in-line documentation to give a high level view of what the program and groups of statements is/are doing.

Turn-in:

1. A description of your design for the program.
2. A listing of your code.
3. Your man page documentation.
4. Output that demonstrates that the code performs correctly. Your output must demonstrate that your program functions correctly. At least one test case must demonstrate that the server can handle two simultaneous games, each between a client and the server.
5. Leave a copy of your source and executable files under your Coursefile work folder for this class in a folder called Project3. Leave an empty file in the folder that has your name on it.

   Turn in your Project 3 results (design description, program code, output, and man page) in a PDF document with the results clearly identified to the D2L Assignment Folder for Project 3. There should be one report per group. Also, each person is to individually turn in their peer evaluation form to the Project 3 Evaluation Assignment folder.

Grading:

Grading of this project will be divided into two parts:

a. A group score (g) consisting of up to 80% of the available points (28 points).
b. An individual (i) score.

These two parts are added together to determine your total score for this project.

The group score is based on the following criteria:

a. Documentation (15 %): this includes a short accurate description of the design and the man page.
b. Readability (10%)
a. Correctness (50%): Does the program meet the requirements?
b. Testing (25%): Your tests are expected to demonstrate that the program(s) are operating correctly.

Each group member is to assign points to the other group members, but not to him/herself. Suppose there are n group members and n*(0.20 * 35) points to be assigned (pool_points). Then each person would assign ((n-1)/n) * pool_points to the other members of the group. This assignment is to be based on the contribution of each group member; in making this decision, consider both the effort expended by each group member and their effectiveness. A signed form with this point distribution is to be given to the instructor by each group member when the project is turned in. The instructor will compute an average potential score, p_k for each group member based on the input received. The actual individual score will then be determined by: $i = p_k * (g / 28)$.

Name: _____

Group : _____

Project: _____

a. Number of people in group: _____ (n)

b. Available Points to Assign: _____ (n – 1) * 0.20 * 35

For each person in the group, except yourself, assign points to that person based on their effort and effectiveness in the project. Note that the sum of the points must not exceed the value computed on line-b above.

Name	Assigned Points	Justification

Signature: _____