**2803ICT – Assignment 1**

Ellis Rourke

Contents

1. Problem Statement 1

2. User Requirements 2

3. Software Requirements 3

4. Software Design 4

5. Requirement Acceptance Tests 6

6. Detailed Software Testing 7

7. User Instructions 8

## Problem Statement

# *(In this section you write a few sentences about what needed to be done for the assigment)*

The goal of this project is to create a client server program using C. The program accepts requests from one or more clients and servers their requests in a predicatable, robust and intuitive fashion.

## User Requirements

* The user should enter a valid command specified in the documentation (or error will be displayed)
* The user should be able to communicate with the server while other clients are also connected to the server.
* The user should be able to run all commands specified in this document without error.
* The user should be able to see how long each operation took.
* The user should be able to specify the connection ip address from the client side.
* The user should be able to enter an unlimited number of commands until they specify the program should quit

## Software Requirements

1. Using a socket connection, the client program will query the remote server program that is listening on port 80.
2. The IP address of the server to be queried by the client shall be given to the client as a command line argument.
3. The client shall wait for the user to enter queries (via stdin), which it then forwards to the server in a loop until the user types ‘quit’. Any responses from the server are immediately displayed to the user.
4. The client will report the time taken for the server to respond to each query together with the server’s response.
5. The client is non-blocking. An infinite number of server queries may be outstanding.
6. The server will spawn a new process to execute each new request and must be able to accept multiple clients.
7. The server will be able to accept one or more source files and a ‘progname’ and place the files in a directory called ‘progname’. It will be able to compile the source files (if not previously compiled), run the executable with command line arguments provided from the client and return the result to the clients.

9. The long list (-l) option of the list command will also return the file size, creation date and access permissions. If no progname is given, then the list of all available progname directories will be returned.

10. The get command will dump the file contents to the screen 40 lines at a time and pause, waiting for a key to be pressed before displaying the next 40 lines etc.

11. The put command will create a new directory on the server called ‘progname’ If the remote progname exists the server will return an error, unless -f has been specified, in which case the directory will be completely overwritten (old content is deleted). This command allows you to upload one or more files from the client to the server

12. If a localfile option is given to the run command a new file on the client will be created. If the localfile name exists the client will return an error, unless -f has been specified in which case the file will be overwritten. If a file with that name already exists the client will return an error before sending the get request to the server. 2

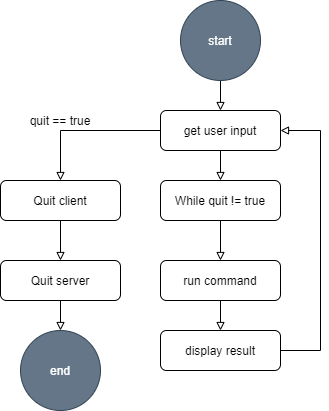
13. The run command will check to see if a ‘progname’ has been compiled, and if not will compile the relevant files as require. run will initiate a compile if there is no executable in the folder, or its creation date is older than the last modified date of a source file. It will then run the executable, passing to it any specified command line arguments, and the server will redirect output from the executed program to the client. If the program can’t be run (or compiled) an appropriate error will be returned to the client. You must not use the system() call to compile or run the ‘progname’.

14. If the server receives an incorrectly specified command it will return an error. If the server is unable to execute a valid command the server will return the error string generated by the operating system to the client.

15. All Zombie processes are terminated as required. There is to be no unwanted Zombie processes on either the client, or the server.

## Software Design

### High Level Design – Logical Block Diagram



### High Level Design – Structure Chart

A picture containing clock

Description automatically generated

### List of all functions in the software.

For each function in the list the following information is provided:

1. a brief description of what it does (1 or 2 sentences);
2. a list of the input parameters, and their data types, and what they are used for;
3. a list of any side effects caused by the function (ie change global or member variables, changes data passed by reference from calling function etc)
4. a description of the function’s return value

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Description | Parameters | Return |
| List | Lists all files and or details about files in current directory. | Optional parameter (-l) Lists additional details on every file | A list of files and or details of the files in the current directory in the console. |
| Sys | Displays to the user the current operating system | None | A string in the console containing the current os type |
| Get | Returns to the user the first 40 lines of a specified file | File name in current directory | A string in console containing the first 40 lines of the file |
| Run | Takes a c file name as input, compiles it and runs it | File name of c program for example (test.c) would have the input test | Result from running c file |
| quit | Ends the current client session and instructs the server to quit too | None | Program is terminated. |

## Requirement Acceptance Tests

| **Software  Requirement No** | **Test** | **Implemented (Full /Partial/ None)** | **Test Results (Pass/ Fail)** | **Comments (for partial implementation or failed test results)** |
| --- | --- | --- | --- | --- |
| 1 | Connect client and server on port 80 | Full | Pass |  |
| 2 | Specify IP address from the client as argument | Full | Pass |  |
| 3 | Client waits for user to input queries | Full | Pass |  |
| 4 | Client reports time taken for each request | Full | Pass |  |
| 5 | The client is non blocking | Full | Pass |  |
| 6 | Server accepts multiple clients | Full | Pass |  |
| 7 | Server can compile and execute c source files | Full | Pass |  |
| 9 | -l option of list displays more detailed information | Full | Pass |  |
| 10 | Get command dumps 40 lines of a file at a time and waits for enter and dumps the next 40 | Partial |  | Program only allows the first 40 lines to be printed. |
| 11 | Put command | Full | Pass |  |
| 12 | Run command checks to see if file has already been compiled, and if so will not recompile the file | Partial |  | The server will recompile the file in any condition to ensure it is upto date |
| 13 | Server can recognise invalid command and will return “error” to the client | Full | Pass |  |
| 14 | Zombie processes should be terminated | Partial | Fail | Unsure of requirement |

## Detailed Software Testing

| **No** | **Test** | **Expected Results** | **Actual Results** |
| --- | --- | --- | --- |
| 1 | Incorrectly spelled command | Server should return an error string to the user | As expected |
| 2 | Input invalid filename | Server responds with string “unable to open file” and returns to the input loop | As expected |
| 3 | More than one client attempts to connect to the server | The server should handle the new client and assign its socket without interupting the other clients | As expected |
| 4 | Robust error handling | Server should handle unexpected errors seamlessly | Server can sometimes fall into a loop where new input is not accepted as it was not recorded that the socket received information back from the client. |
| 5 | List with or without -l | Server should return the correct version of this list version based upon the -l flag | As expected |
| 6 | Put command run | Server recieves file data from client and either created a new file or updates the current copy | Not Implemented |
| 7 | Handle extra prefix added to a command | Server will not accept command (ie. Sys -l) is not value | As expected |
| 8 | Run command, execution error with c file to be run | Server returns command line error from the operation and returns to the client | As expected |

## User Instructions

# *(Here you state what a user needs to do to run and use your program(s), this also includes and special instructures on how to compile the program(s) as necessary)*

Compilation instructions:

Server:

Gcc server.c -o server

./server

Client:

Gcc client.c -o client

./client

Available Commands:

* Get [filename]
  + Prints 40 lines of specified file in server directory
* Put [filename]
  + Uploads specified file in client directory to server directory
* Sys
  + Displays the OS type of the server
* Time
  + Displays the local time of the server
* Hostname
  + Displays the hostname of the server system
* List [optional -l]
  + Displays all files in the server directory
  + [-l] flag displays additional information on these files
* Run filename (without .c)
  + Compiles and runs c files on the server system