Querying the Status of Remote Servers

2803 Systems and Distributed Computing

Damon Murdoch (s2970548)

1.0 Problem Statement

The objective of this project is to develop a client-server system for querying remote servers. A number of predefined commands will be accepted by the client program, forwarded to the server program and then the result will be returned to the client.

2.0 User Requirements

1. The user must be able to provide the IP to connect to.
2. The user must be able to send a series of predefined commands and arguments to the server.
3. The user must be able to provide filenames for the system to read and write from.
4. The user must be able to use the '|' operator to redirect output to another process.
5. The user must be able to use ctrl-c to safely close the server application.
6. The user must be able to enter 'quit' to safely shutdown the client application.

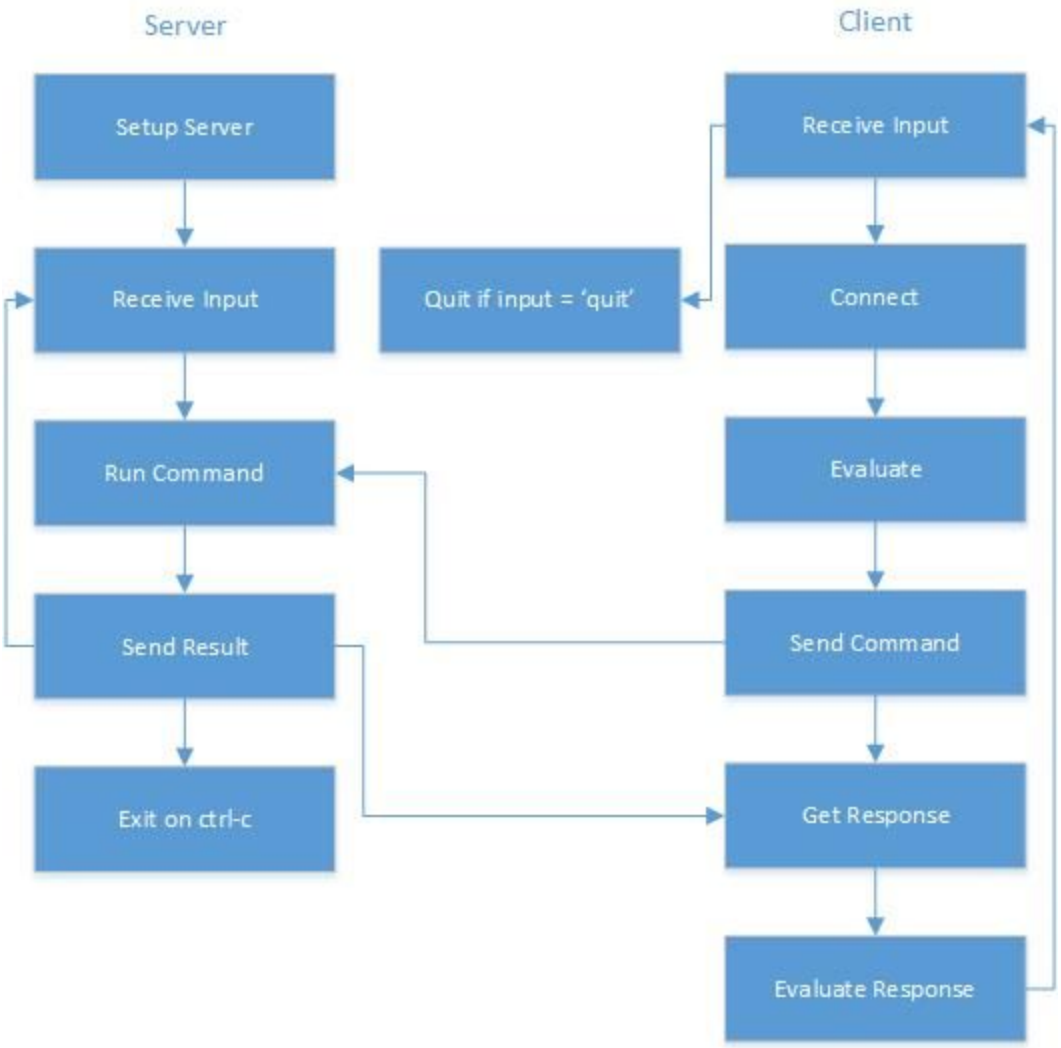
3.0 Software Requirements

1. The server application must be hosted on port 80
2. The client application will accept the address of the server as a command line argument.
3. The client application must wait for the user to enter queries, and when input is accepted it must forward it to the server and display any response as output.
4. The client application must report the time between sending the message and recieving the response.
5. The client application must be non-blocking.
6. The server application must spawn a new process to handle each request, and must be able to accept multiple clients.
7. The system must be able to run on Windows and Unix operating systems.
8. The following commands must be supported by the server: (arguments in [] are optional)
   List [-l] [-f] [pathname] [localfile] - Prints a list of all the files in the current directory or at 'pathname' to stdout or 'localfile'.
   Get filepath [-f] [localfile] - Print content of 'filepath' to stdout or 'localfile'
   Put localfile [-f] [newname] - Create remote copy of local file with same or other name
   Sys - Display name / version of OS and CPU Type
   Delay Integer - Returns 'integer' after a delay of 'integer' seconds.
   Quit - Closes the client program.
9. The "-l" argument for list, if provided must also return the file size, owner, creation date and access permissions of the file.

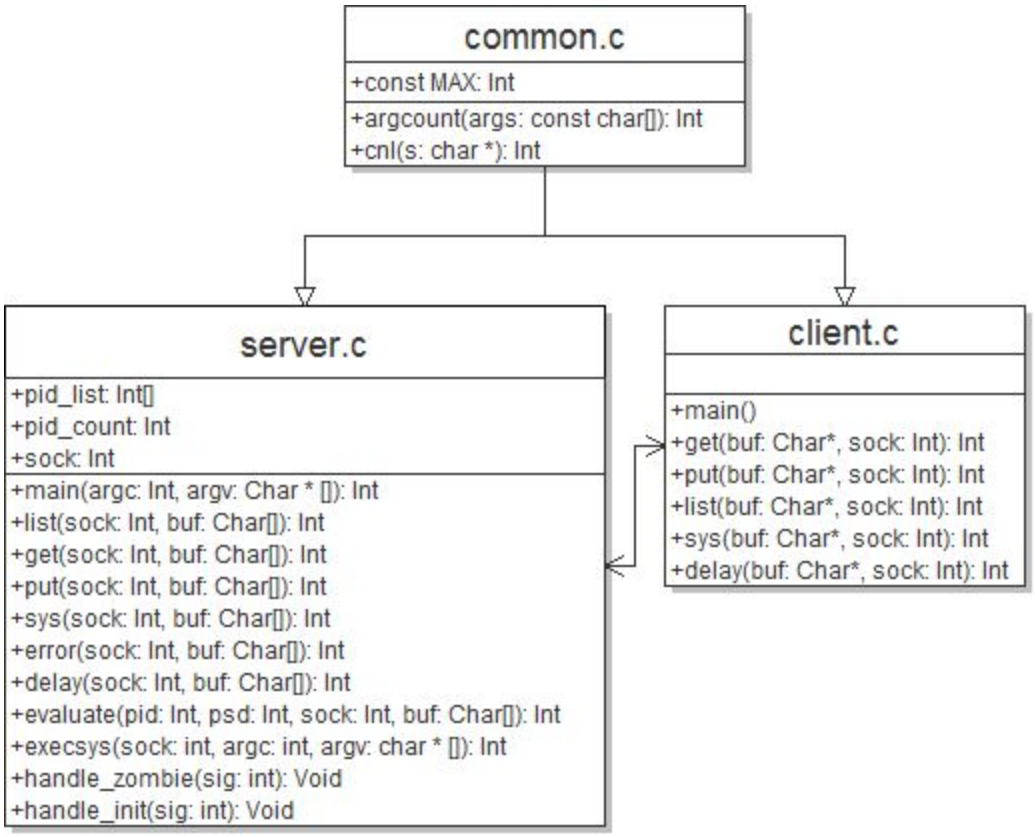
10. The “-f” argument if provided, any files which are created by the program must ovewrite existing files. If this argument is not provided and an existing file is needed to be overwritten, the write will fail.
11. If no filename is provided for list, the directory listing must be printed to the screen forty lines at a time.
12. Relative and local path names must be accepted for all functions.
13. The delay command must not interrupt any other processes running in the application.
14. The program must be able to redirect output to other processes using the ‘|’ argument.
15. All zombie processes must be exited when the application is closed.
16. If a command fails, the server must send the system error message to the client.

4.0 Software Design

4.1 Logical Block Diagram



4.2 UML Diagram



4.3 Function Definitions

4.3.1 Common

Argcount

Description: Returns the number of arguments in the provided string separated by \0, \t, \n, or \r.
Parameters:char array args
Side-Effects: None
Return: Integer containing the number of arguments

cnl

Description: Strips the newline character from the character pointed to by 's'
Parameters: Character pointer ‘s’
Side-Effects: Removes ‘\n’ at the last character of the string at ‘s’
Return: 0 on successful termination

4.3.2 Client

Get

Description: Gets a file from the server specified by buf and displays or stores it on the local machine.
Parameters:
Char * buf: Character pointer to the arguments to the array
Int sock: Socket to read/write data.
Side-Effects: Strtok modifying buf array
Return: 0 on successful termination

Put

Description: Sends a filestream to be stored on the server, with an optional new filename.
Parameters:
Char * buf: Character pointer to the arguments to the array
Int sock: Socket to read/write data.
Side-Effects: Strtok modifying buf array
Return: 0 on successful termination

List

Description: Parses buf for arguments, submits them to the server and prints the response to stdout or a file.
Parameters:
Char * buf: Character pointer to the arguments to the array
Int sock: Socket to read/write data.
Side-Effects: Strtok modifying buf array
Return: 0 on successful termination

Main

Description: The main program loop which accepts commands, connects to the server program, submits the commands to the server, runs relevant commands depending on the input and displays the output, time taken for server to respond.
Parameters:
int argc: Number of arguments.
Char * argv[] : array of character pointers containing arguments.
Side-Effects: None
Return: 0 on successful termination

4.3.3 Server

List

Description: Parses buf for arguments and passes them to execsys.
Parameters:
Int sock: Socket to send/recieve data
Char * buf: String containing arguments
Side-Effects: Strtok modifying buf array
Return: 0 on successful termination

Get

Description: Opens a local file specified by buf and sends it to the client.
Parameters:
Int sock: Socket to send/recieve data
Char * buf: String containing arguments
Side-Effects: Strtok modifying buf array
Return: 0 on successful termination

Put

Description: Accepts a filestream from the client and stores it in a local file, based upon the parameters in buf.
Parameters:
Int sock: Socket to send/recieve data
Char * buf: String containing arguments
Side-Effects: Strtok modifying buf array
Return: 0 on successful termination

Sys

Description: Sends the system information of the server to the client.
Parameters:
Int sock: Socket to send/recieve data
Char * buf: String containing arguments
Side-Effects: None
Return: 0 on successful termination

Error

Description: Sends an error message to the client.
Parameters:
Int sock: Socket to send/recieve data
Char * buf: String containing arguments
Side-Effects: None
Return: 0 on successful termination

Delay

Description: Delays the client for the number of seconds provided by 'buf'.
Parameters:
Int sock: Socket to send/recieve data
Char * buf: String containing arguments
Side-Effects: Strtok modifying buf array
Return: 0 on successful termination

Evaluate

Description: Runs a function based on the value of buf, and passes it int and buf.
Parameters:
Int pid: Program ID that called the function
Int psd: Connection to client
Int sock: Socket to read / write data
Char * buf: String containing arguments
Side-Effects: None
Return: 0 on successful termination

Execsys

Description: executes a system command using the arguments in argv and sends the result to sock.
Parameters:
Int sock: Socket to read/write data
Int argc: Number of arguments provided in argv
Char * [] argv: Array containing arguments for execvp
Side-Effects: Creates a fork child process, Routes the output from execvp to a buffer string
Return: 0 on successful termination, 1 on failure

Main

Description: Hosts a server on port 80, accepts connections from multiple clients, handles incoming queries and passes input to relevant functions. Cleans up all child processes upon manual termination.
Parameters:
Int argc: Number of arguments provided in argv
Char * [] argv: Array containing arguments
Side-Effects:
Insert values to pid_list, increase pid_count, generate child processes
Return: 0 on successful program completion

4.4 Data Structures

4.4.1 Client

Buf
Type: Char[1024]
Description: Stores the command line input from the user.
Members: up to 1023 characters entered by the user.
Functions: Main, get, put, list

Rcv
Type: Char[1024]
Description: Stores messages recieved from the server. This variable is created locally for each function and is not shared between functions.
Members: up to 1023 characters entered by the user.
Functions: Main, get, put, list

4.4.2 Server

Pid_list
Type: Int[1024]
Description: An array containing every program id being used by a subprocess for the application.
Members: Program ids to close when the main process is ended.
Functions: handle_int, main

Buf
Type: char *
Description: Char array which contains input recieved from a client application.
Members: up to 1023 characters recieved from the client.
Functions: main, evaluate, list, get, put, sys, delay, error

Argv
Type: Char *
Description: Char pointer array which are used inside each list, get, and put functions in order to split the buffer array.
Members: Each index contains a single argument.
Functions: list,get,put

Tmp
Type: Char *
Description: Temporarily stores information which is parsed using strtok, before it is inserted into an argv array.
Members:
Functions: list,get,put,delay

4.5 Detailed Design

4.3.1 Client

4.3.1.1 Main

```
main(char * address)
	while(TRUE)
		Char array buf[1024], rcv[1024]
		Timeval t1,t2
		Flush stdin
		Accept user input and store in buf
		Strip newline from buf
		If buf[0] = quit, close the program
		T1 = current time
		Create socket
			quit if socket creation fails
		Get host data from address
		Connect to host
			Quit if connection fails
		If buf[0] is list, send buf and run list command
		If buf[0] is get, send buf and run get command
		if buf[0] is put, send buf and run put command
		Else
			Send buf, recieve response, print response
		T2 = current time
		Get time difference
		Print time difference
		Close socket
	Return 0
```

4.3.1.2 list

```
list(char * buf, int sock)
	Create variables
	If buffer has more than one argument
		Read buffer arguments into array
		Identify buffer arguments
		Send command to server
		Recieve response from server
		If destination file is provided in buf
			If force is not set
				Attempt to open file
					If file already exists,file is not accessible
				Close file
			If file is accessible
				Open file
				If file exists
					Put recieved file data into local file
				Close local file
			Else
				Print error
		Else
			while(recieved buffer is not empty)
				If fortieth line has been printed
					Wait for user input
				Reset count
				Print line
	Else
		Print input error
	Return 0
```

4.3.1.3 Put

```
put(char * buf, int sock)
	Create variables
	If buffer has more than one argument
		Read buffer arguments into array
		Identify buffer arguments
		Open file to send
		If file can be opened
			Recieve response from server
			If file cannot be moved
				print error and return 0
			Else
				Send file data to server
		Else
			print file error
	Else
		print input error
Return 0
```

4.3.1.4 Get

```
get(char * buf, int sock)
	Create variables
	Read buffer arguments into array
	Evaluate arguments
	If destination file is provided in buf
		If force is set and file already exists
			cannot access
		If file can be accessed
			open file
		Else
			Print cannot open file
	while(TRUE)
		Recieve text from server
		If file is opened
			Write to file
		Else
			Write to standard output
		If recieved text = END_OF_STREAM or FILE_NOT_FOUND
			break
	If f is open, close it
	Return 0
```

4.3.2 Server

4.3.2.1 Main

```
main()
	Initialise signals
	Create socket
	Create and assign address structure
	Bind address to socket
	Listen for connections
	while(TRUE)
		Accept connection
		Create new process
		If fork process fails, continue to next iteration
		If this process = child process
			Recieve input from client
			Call evaluate on input
			Close process
		If this process = parent process
			Add child program ID to program ID list and increment count
			Cleanup zombie processes
	Return 0
```

4.3.2.2 Get

```
Get(int sock, char * buf)
	Create variables
	If buf has at least 2 arguments
		Split buf into array argv and print each element to server terminal
		Open filename in argv[1]
		If file opened successfully
			Send contents to client
			Send END_OF_STREAM
		Else
			Print file error, Send file error to client, return 1
	Else
		Send argument error
	Return 0
```

4.3.2.3 Put

```
Put(int sock, char * buf)
        Create variables
        If argument count is at least 2
                Read buf args into array
                If there are more than 2 arguments
                        Evaluate additional ar guments
                If force is not set
                        If file to write to already exists
                                Print error, send error, close file
                If file is accessible
                        Open write file
                                While stream has not ended
                                        Store recieved data in file
        Return 0
```

4.3.2.4 Execsys

```
Execsys(int sock, int argc, char * argv[])
        Create pipe
        If pipe creation fails
                Exit function with error
        Create child process
        If child creation fails
                Exit function with error
        If current process is the child process
                Run command line program specified in argv
                If run fails, send errorno to client and exit with error
        If current process is parent process
                send command line program output to client
        Wait for child to exit
        return 0
```

5.0 Requirement Acceptance Tests

User Requirement No	Test	Implemented (Full/Partial/None)	Test Results (Pass/Fail)	Comments
1	User can provide IP address	Full	Pass	
2	User can send predefined commands and arguments to the server	Full	Pass	
3	User can provide filenames to read and write from	Full	Pass	
4	User can use '|' to redirect output to other processes	None	Fail	This feature has not been implemented.
5	User can close server application safely with ctrl-c	Full	Pass	
6	User can enter 'quit' to safely close the client application	Full	Pass	

Software Requirement No	Test	Implemented (Full/Partial/None)	Test Results (Pass/Fail)	Comments
1	Server hosted on port 80	Full	Pass	
2	Client provides address to query as command line argument	Full	Pass	
3	Client waits for the user to input qieries and forwards them to the server, then displays response	Full	Pass	
4	Time report between sending command and recieving response	Full	Pass	
5	Client must be non-blocking	None	Fail	Commands cannot be sent to the server while another command is being processed for that client.
6	Server spawns new process for each request and handles multiple clients	Full	Pass	
7	System must be able to run on windows and unix operating systems.	Partial	Fail	Windows code is not supported.

8	List,Get,Put,Sys, Delay and quit function implementations	Full	Pass	
9	List -l argument	Full	Pass	
10	Query -f Argument	Full	Pass	
11	Directory listing 40 lines at a time if no filename provided	Full	Pass	
12	Relative and local path name support	Full	Pass	
13	Delay command does not interrupt other processes	Partial	Fail	Delay does not interrupt existing processes, however new processes cannot be spawned by the delaying client
14	Redirect output using | argument	None	Fail	This feature is not supported by the program.
15	Zombies killed on application close	Full	Pass	
16	Error message sent to user on command fail			

6.0 Detailed Software Testing

No	Test	Expected Results	Actual Results
List Function			
1.	list	List server directory files	Expected result
2.	List C:/	List C:/ directory files on server	Expected Result
3.	List C:/ c.txt	Created text file listing all files in C:/	Expected Result
4.	List C:/Cygwin c.txt	Fail to overwrite existing file	Expected Result
5.	List -f C:/Cygwin c.txt	Overwrite existing file	Expected Result
6.	List -l C:/	List all fiels in C:/ with long notation	Expected Result
Get Function			
1.	Get c.txt	Print the results of c.txt	Expected Result
2.	Get c.txt newc.txt	Create a new file containing the contents of c.txt	Expected Result
3.	Get newc.txt c.txt	Fail to overwrite existing file	Expected Result
4.	Get -f newc.txt c.txt	Overwrite existing file	Expected Result
5.	Get	Print argument error to client console	Expected Result
Put Function			
1.	Put c.txt c2.txt	Create local copy of file	Expected Result
2.	Put c2.txt c.txt	Fail to overwrite file	Expected Result

3.	Put -f c2.txt c.txt	Overwrite existing file	Expected Result
4.	Put c.txt	Create local copy of file	Expected Result
5.	Put c.txt -f	Overwrite existing file	Expected Result
6.	Put c.txt	Fail to overwrite file	Expected Result
7.	Put	Print argument error to client console	Expected Result
Sys Function			
1.	Sys	Print system data to client terminal	Expected Result
Delay Function			
1.	Delay 1	Delay 1 second and display 1 in client terminal	Expected Result
2.	Delay 100, list	Delay 100 seconds, fail to initiate new command in terminal	Delay 100 seconds, initiate new command
3.	Delay word	No delay, send input error to client	Expected Result

7.0 User Instructions

1. Execute the client and server applications.
2. On the client application, enter the IPv4 address for the server you wish to connect to.
3. You may now start sending commands to the server using the client application.
4. Once you are finished entering commands, you may close the client application using ‘quit’.
5. You may use the ctrl-c interrupt to close the server application.