My Email System (MES)

Table of Contents

Introduction

My email system is a combination of two programs (MES-C & MES-S) that is developed to implement a own personal email system where, client program is waiting for commands from the user and server program waiting to receive commands from client program, to provide service to the user.

Files

- A brief introduction for each file included in the software solution.

acc.h file :

- This file act as a header file and includes all the header files that both client and server programs needs program needs to run without any errors. Wrapper functions that are related to threads also have defined in the “acc.h” file. Doing this helps to avoid including every header file every time for each file that is going to be compiled.

```
17  #include <stdlib.h>
18  #include <unistd.h>
19  #include <time.h>
20  #include <string.h>
21  #include <errno.h>
22  #include <netdb.h>
23  #include <sys/uio.h>
24  #include <arpa/inet.h>
25  #include <netinet/in.h>
26  #include <sys/types.h>
27  #include <sys/socket.h>
28  #include <pthread.h>
29  #include <signal.h>
30  #include <dirent.h>
31
32
33  #ifdef  HAVE_PTHREAD_H
34  #include    <pthread.h>
35  #endif
```

Figure 01

```
69  void    Pthread_create(pthread_t *, const pthread_attr_t *, void * (*)(void *), void *);
70      /*void * (*)(void *), void *);*/
71  void    Pthread_join(pthread_t, void **);
72  void    Pthread_detach(pthread_t);
73  void    Pthread_kill(pthread_t, int);
74
75  void    Pthread_mutexattr_init(pthread_mutexattr_t *);
76  void    Pthread_mutexattr_setpshared(pthread_mutexattr_t *, int);
77  void    Pthread_mutex_init(pthread_mutex_t *, pthread_mutexattr_t *);
78  void    Pthread_mutex_lock(pthread_mutex_t *);
79  void    Pthread_mutex_unlock(pthread_mutex_t *);
80
81  void    Pthread_cond_broadcast(pthread_cond_t *);
82  void    Pthread_cond_signal(pthread_cond_t *);
83  void    Pthread_cond_wait(pthread_cond_t *, pthread_mutex_t *);
84  void    Pthread_cond_timedwait(pthread_cond_t *, pthread_mutex_t *,
85                                 const struct timespec *);
86
87  void    Pthread_key_create(pthread_key_t *, void (*)(void *));
88  void    Pthread_setspecific(pthread_key_t, const void *);
89  void    Pthread_once(pthread_once_t *, void (*)(void));
90
91  #endif  /* __unp_pthread_h */
92
```

Figure 02

mssc.c file :

- In this a socket is created for client to communicate with the server. Have not defined any IP address to socket so that kernel will assign a IP address and port to the socket. And socket address structure is created to store port and family of the server, so it will enable to communicate with the server.
- Communication with the server is handled by “str_cli()” function.

client_thread.c file :

- This file contains “str_cli()” function that will create a separate new thread to look for new data at the socket ,whereas main thread is looking for new data at standard input.
- When user enters any command main thread will catch it and write it to the socket.
- When data is available at the socket new thread will pass it to the standard output.

msss.c file :

- Includes code for hoe the thread pool is created and each function that run by each thread at the thread pool.
- Contains code to for main thread to create new threads to serve clients when all the threads at the thread pool are busy serving clients.
- Includes how server socket have been created to communicate with the client.
- Signal handler is placed to catch signal when the server programs get the EOF character to terminate the server program.
- Before creating threads and all this server program makes folder called “mailboxes” and all the mails and lists will be stored in that particular folder.
- Upon terminating server program, “mailboxes” folder will be deleted including files inside the “mailboxes” folder.

```
125     for ( ;; )
126     {
127         //Creating new threads to handle new clients when whole thread pool is busy
128         if(count == noOfCons)
129         {
130             nlen = naddrlen;
131 
132             //Mutex lock makes sure only one thread will call accept at the time and update
133             Pthread_mutex_lock(&mlock);
134 
135             if(( connfd = Accept(listenfd, ncliaddr, &nlen)) > 3)
136             {
137                 client_id++;
138             }
139 
140             cli_info.connfd = connfd;
141             cli_info.cli_id = client_id;
142             cli_info.listen_fd = listenfd;
143 
144             Pthread_mutex_unlock(&mlock);
145 
146             Pthread_create(&tid, NULL, &doit, (void *) &cli_info);
147         }
148     }
149 
150 }
```

Figure 03

serve_client.c file :

- Includes way of serving client for each command.
- When a client sends a command,

1. Checks number of whitespaces in the command.

```
66          //Accessing line[] array charachter by charachter
67          for(j=0; j<size; j++)
68          {
69              //checking if there is any whitespace as charachter
70              if(isspace(line[j]))
71              {
72                  count++;
73                  r = 1;
74              }
75          }
```

Figure 04

2. Then check the number of whitespaces in the command by several "if() " conditions.

```
159          //one Whitespace found (Two word comand)
160          else if (r == 1 && count == 1)
161          {
```

Figure 05

```
77          //no whitespace
78          if (r == 0 && count == 0)
79          {
```

Figure 06

```
501          //Two whitespaces found
502          else if (r == 1 && count == 2)
503          {
```

Figure 07

```
749          //More than two whitespaces(Incorrect command)
750          else if( count > 2 )
751          {
```

Figure 08

3. Then assign each word to new variable by dividing the command by whitespace delimiter.

```
527         char *ptr = strtok(line, delim);
528         first = ptr;
529
530         ptr = strtok(NULL, delim);
531         second = ptr;
532
533         ptr = strtok(NULL, delim);
534         third = ptr;
535
536         int idxToDel = strlen(third)-1;
```

Figure 09

4. Then gets first word divided from the command and compare it with command list and executes the related function.

```
541         if((strcasecmp("delete",first)) == 0)
542         {
```

Figure 10

```
633         else if((strcasecmp("read",first)) == 0)
634         {
```

Figure 11

```
416       else if((strcasecmp("get_mailbox",first)) == 0)
417       {
```

Figure 12

```
238       else if((strcasecmp("send",first)) == 0)
239       {
```

Figure 13

makeclient file :

- Contains all the compilation commands for “MES-C” program to run without errors.

```
mssc.out: error.o wrapunix.o wrapsock.o wraplib.o wrapstdio.o mssc.o readline.o writen.o wrappthread.
	cc -o mssc.out wrappthread.o client_thread.o error.o wrapunix.o wrapsock.o wraplib.o wrapstdio.o

error.o: error.c acc.h
	cc -c error.c

wrapunix.o: wrapunix.c acc.h
	cc -c wrapunix.c

wrapsock.o: wrapsock.c acc.h
	cc -c wrapsock.c

wraplib.o: wraplib.c acc.h
	cc -c wraplib.c

wrapstdio.o: wrapstdio.c acc.h
	cc -c wrapstdio.c

mssc.o: mssc.c acc.h
	cc -c mssc.c

client_thread.o: client_thread.c acc.h
	cc -c client_thread.c

readline.o: readline.c acc.h
	cc -c readline.c

writen.o: writen.c acc.h
	cc -c writen.c
wrappthread.o: wrappthread.c acc.h
	cc -c wrappthread.c
```

Figure 14

makeserver file :

- Contains all the compilation commands for “MES-C” program to run without errors.

```
msss.out: error.o wrapunix.o wrapsock.o wraplib.o wrapstdio.o msss.o readline.o writen.o wrappthread.
    cc -o msss.out wrappthread.o serve_client.o error.o wrapunix.o wrapsock.o wraplib.o wrapstdio.o m

error.o: error.c acc.h
    cc -c error.c

wrapunix.o: wrapunix.c acc.h
    cc -c wrapunix.c

wrapsock.o: wrapsock.c acc.h
    cc -c wrapsock.c

wraplib.o: wraplib.c acc.h
    cc -c wraplib.c

wrapstdio.o: wrapstdio.c acc.h
    cc -c wrapstdio.c

msss.o: msss.c acc.h
    cc -c msss.c

serve_client.o: serve_client.c acc.h
    cc -c serve_client.c

readline.o: readline.c acc.h
    cc -c readline.c

signal.o: signal.c acc.h
    cc -c signal.c

writen.o: writen.c acc.h
    cc -c writen.c
```

Figure 15

Other files :

- “error.c”- Methods to handle errors generated by functions, errors are handled according to fatality of error produced by the function.
- “readline.c”- Contains function to read data from file descriptors.
- “signal.c”- Inclueds signal handlers.
- “wraplib.c”- Contains wrapper functions for “inet_ntop() & inet_pton()” functions.
- “wrapthred.c”- Contains wrapper functions for handling threads.
- “wrapsock.c”- Contains wrapper functions for socket related functions.
- “wrapstdio.c”- Contains wrapper functions for file handling related functions.
- “wrapunix.c”- Contains wrapper functions for common C programming related functions.
- “written.c”- Contains wrapper functions related to “written()” function.

Mutual Exclusion

- In server program, to avoid several threads updating connection file descriptor's value simultaneously have used a mutex lock, so that only one thread will be able to update connection file descriptor at a time.

```
132             //Mutex lock makes sure only one thread will call accept at the time and update
133             Pthread_mutex_lock(&mlock);
134
135             if(( connfd = Accept(listenfd, ncliaddr, &nlen)) > 3)
136             {
137                 client_id++;
138             }
139
140             cli_info.connfd = connfd;
141             cli_info.cli_id = client_id;
142             cli_info.listen_fd = listenfd;
143
144             Pthread_mutex_unlock(&mlock);
145
146             Pthread_create(&tid, NULL, &doit, (void *) &cli_info);
147         }
```

Figure 16

- Created new structure to store each thread id and allocated dynamic memory to store thread id at each structure

```
22  typedef struct{
23      //To store the thread ID
24      pthread_t thread_tid;
25  }Thread;
26
27  //Creating array of thread structures that was previously defined
28  Thread *tptr;
29
```

Figure 17

```
100         //Allocating dynamic memory for the thread strcuture
101         tptr = calloc( noOfCons,sizeof(Thread) );
```

Figure 18

- There are main three shared resources in this software solution,
    1) “clientlist.txt” file
    2) “mailbox_info.txt” file
    3) “mail.txt” file

- Before perform update operations on each of the file, mutex lock is acquired to avoid several threads accessing the file simultaneously giving only one thread to access and perform an update on the file. When finished updating the file the lock is released for other threads to access the file.

```
774     Pthread_mutex_lock(&nlock);
775
```

Figure 19

```
774     Pthread_mutex_lock(&nlock);
775
```

Figure 20

Problems of the program

- When all threads in thread pool are busy serving clients, then main thread creates new threads to handle new clients. And when one thread from thread pool becomes free after serving client, whenever a new client comes, the new client won't be given to thread that is free in the thread pool instead the client will be given a new thread created by the main thread but client coming will be given to the thread that is free in the thread pool.

- The client will have to enter a number for each email the user is writing, but if the client enters number that is already saved as a mail file server will tell the client that a email is already existing under the same number and will prompt to enter number until client enters a unique number.

Assumptions

- Assumed that user is independent from client. Client is made by user using "make client_name" command by the user. Simply a user make many clients just using one instance of client program.
- Client that are not listed under "client_list" can send mails but only to clients listed at the "client_list".

How to compile

- Server Program

```
[Foxtrot@localhost Assignment]$ make -f makeserver
```

Figure 21

- Client program

```
[Foxtrot@localhost Assignment]$ make -f makeclient
```

Figure 22

How to run the programs

- Server program

```
[Foxtrot@localhost Assignment]$ ./msss.out 2 50002
MES-S > Starting up thread 0
MES-S > Starting up thread 1
```

Figure 23

- Client program

```
[Foxtrot@localhost Assignment]$ ./mssc.out 127.0.0.1 50002
MES-C >
```

Figure 24

Sample inputs and outputs

- Create a mailbox for a client

```
MES-C > make sam
MES-C > Mailbox for the user created successfully
MES-C > make mike
MES-C > Mailbox for the user created successfully
MES-C > make ross
MES-C > Mailbox for the user created successfully
MES-C > make rachel
MES-C > Mailbox for the user created successfully
MES-C > make louis
MES-C > Mailbox for the user created successfully
MES-C > make diana
MES-C > Mailbox for the user created successfully
MES-C > make martha
MES-C > Mailbox for the user created successfully
MES-C > make bruce
MES-C > Mailbox for the user created successfully
MES-C >
```

Figure 25

- Send a mail to a user

```
MES-C > send oshan
MES-C > The recieving user verified
MES-C > Please enter the senders name :
MES-C > dulaj
MES-C > Please enter a unique id for the email. :
MES-C > 1
MES-C > Their is email with the id you enter. Please re-enter :
MES-C > 2
MES-C > Please enter the message and press enter. :
MES-C >
MES-C > Hello how are you
MES-C > Done! Mail sent.
MES-C >
```

Figure 26

- Get client list

```
MES-C > get_client_list
MES-C > 0 oshan 127.0.0.1 51048 2 8:26:33
MES-C > 1 sam 127.0.0.1 51050 0 8:31:16
MES-C > 1 mike 127.0.0.1 51050 0 8:31:56
MES-C > 1 ross 127.0.0.1 51050 0 8:32:0
MES-C > 1 rachel 127.0.0.1 51050 0 8:32:18
MES-C > 1 louis 127.0.0.1 51050 0 8:32:38
MES-C > 1 diana 127.0.0.1 51050 0 8:32:45
MES-C > 1 martha 127.0.0.1 51050 0 8:32:55
MES-C > 1 bruce 127.0.0.1 51050 0 8:33:3
MES-C >
```

Figure 27

- Get mailbox of a particular client (Before reading any mails)

```
MES-C > get_mailbox oshan
MES-C > 1 dulaj 8:28:32 unread
MES-C > 2 dulaj 8:29:16 unread
MES-C > 3 martha 8:36:51 unread
MES-C > 4 bruce 8:37:52 unread
MES-C > 6 sam 8:38:15 unread
MES-C >
```

Figure 28

- Read mail of a particular client

```
MES-C > read oshan 2
MES-C > User verified!
MES-C > From: dulaj
MES-C > To: oshan
MES-C > Date: Mon, 22 Oct 118 8:29:16
MES-C >
MES-C > Hi oshan,
MES-C > Hello how are you
MES-C >
MES-C > Regards
MES-C >
MES-C > dulaj
MES-C > Done! MailBox info updated.
MES-C >
```

Figure 29

- Get mailbox of a particular client (After reading mails)

Figure 30

- Get client list (Before deleting a mail)

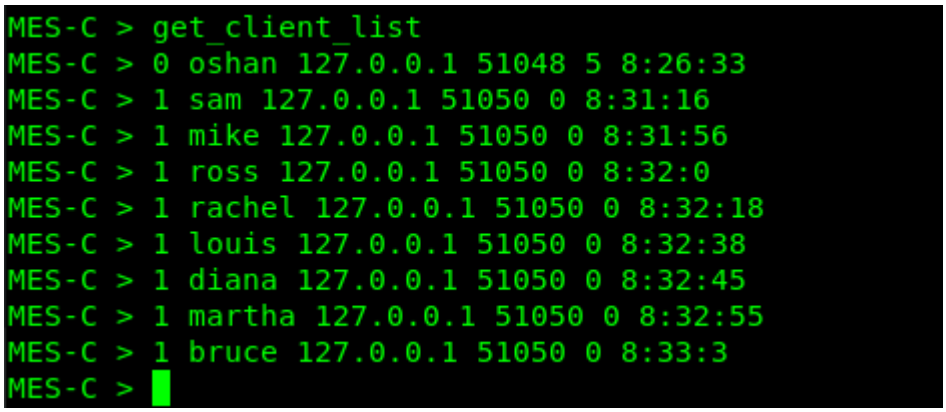

Figure 31

- Delete a mail of a particular client

Figure 32

- Get client list (After deleting a mail)

```
MES-C > get_client_list
MES-C > 0 oshan 127.0.0.1 51048 4 8:26:33
MES-C > 1 sam 127.0.0.1 51050 0 8:31:16
MES-C > 1 mike 127.0.0.1 51050 0 8:31:56
MES-C > 1 ross 127.0.0.1 51050 0 8:32:0
MES-C > 1 rachel 127.0.0.1 51050 0 8:32:18
MES-C > 1 louis 127.0.0.1 51050 0 8:32:38
MES-C > 1 diana 127.0.0.1 51050 0 8:32:45
MES-C > 1 martha 127.0.0.1 51050 0 8:32:55
MES-C > 1 bruce 127.0.0.1 51050 0 8:33:3
MES-C >
```

Figure 33

- Get mail box info (After deleting mail)

```
MES-C > get_mailbox oshan
MES-C > 2 dulaj 8:29:16 read
MES-C > 3 martha 8:36:51 unread
MES-C > 4 bruce 8:37:52 read
MES-C > 6 sam 8:38:15 unread
MES-C >
```

Figure 34

- Quit (from client program)

```
MES-C > quit
MES-C > Bye! Thank you for using MES.
MES-C > [Foxtrot@localhost Assignment]$
```

Figure 35

- Trying to make a mailbox for a client that is already existing

```
MES-C > make oshan
MES-C > The user already exists! Please try again with different username.
MES-C >
```

Figure 36

- Incorrect command

Figure 37

Figure 38

- Trying to send a mail to a client that is not existing

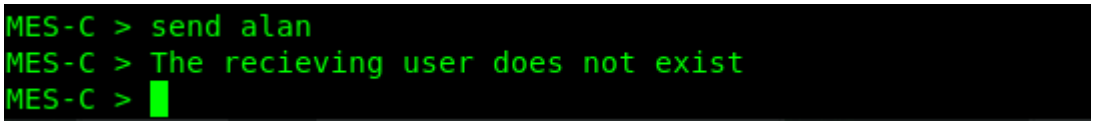

Figure 39

- Trying to get mailbox of a client that does not existing

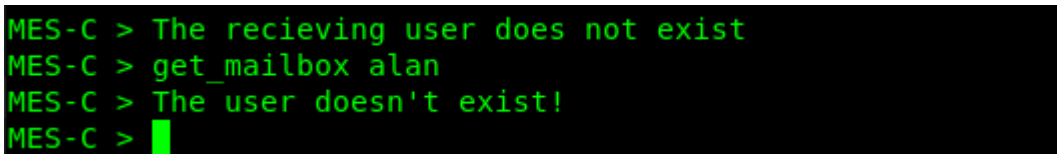

Figure 40

- Trying read a mail does not exist

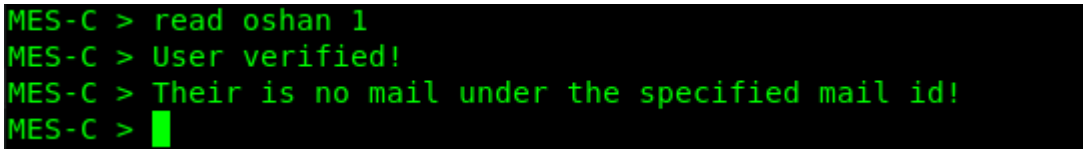

Figure 41

- Trying read a mail from a client that does not exist

Figure 42

- Trying delete a mail that does not exist

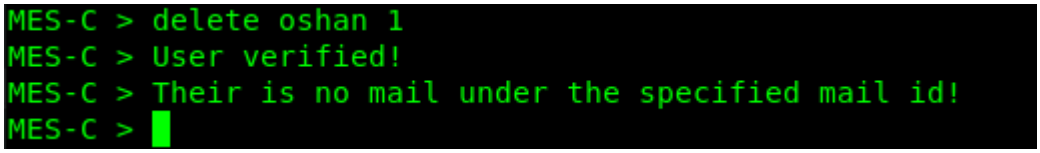

Figure 43

- Trying delete a mail from a client that does not exist

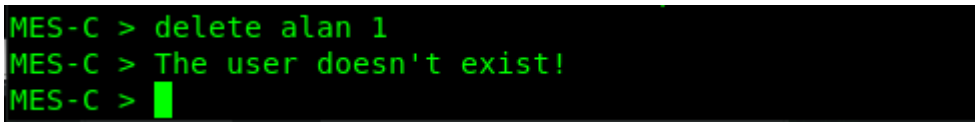

Figure 44

- Entering a mix case command (capital and simple mix letters)
- Server will ignore the case

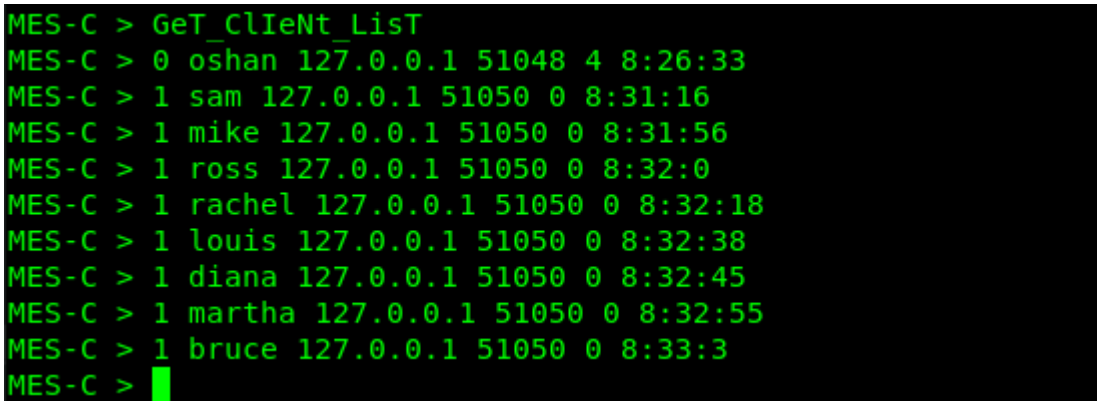

Figure 45

References

[1] Curtin University.(AU). Worksheet 05: Threads. In Advanced Computer Communications (Semester 2 2018 Sri Lanka Inst Info Tech - INT[1]). Retrieved from https://lms.curtin.edu.au

[2] Stevens, W. Richard, Bill Fenner, and Andrew M Rudoff. 1999. UNIX Network Programming. Chapter 3. Sockets Introduction (lib/writen.c). 1st ed. Boston: Addison-Wesley/Prentice-Hall.

[3] "Print Statement Won't Print Before An Infinite Loop". 2018. Stack Overflow. https://stackoverflow.com/questions/13667625/print-statement-wont-print-before-an-infinite-loop.

[4] Stevens, W. Richard, Bill Fenner, and Andrew M Rudoff. 1999. UNIX Network Programming. chapter 26 (threads/strclithread.c). 1st ed. Boston: Addison-Wesley/Prentice-Hall.

[5] Stevens, W. Richard, Bill Fenner, and Andrew M Rudoff. 1999. UNIX Network Programming. Appendix D. Miscellaneous Source Code (lib/error.c). 1st ed. Boston: Addison-Wesley/Prentice-Hall.

[6] Curtin University.(AU). Worksheet 03 (tcpcli01.c). In Advanced Computer Communications (Semester 2 2018 Sri Lanka Inst Info Tech - INT[1]). Retrieved from https://lms.curtin.edu.au

[7] Curtin University.(AU). Worksheet 05(tcpserv01.c): Threads. In Advanced Computer Communications (Semester 2 2018 Sri Lanka Inst Info Tech - INT[1]). Retrieved from https://lms.curtin.edu.au

[8] Curtin University.(AU). Module 05: Threads. In Advanced Computer Communications (Semester 2 2018 Sri Lanka Inst Info Tech - INT[1]). Retrieved from https://lms.curtin.edu.au

[9] Stevens, W. Richard, Bill Fenner, and Andrew M Rudoff. 2012. UNIX Network Programming(Chapter 30) : serv07. 1st ed. Boston [etc.]: Addison-Wesley.

[10] "Terminating A Thread In C". 2018. Stack Overflow. https://stackoverflow.com/questions/6836894/terminating-a-thread-in-c.

[11] Stevens, W. Richard, Bill Fenner, and Andrew M Rudoff. 1999. UNIX Network Programming. Chapter 3. Sockets Introduction (lib/readline.c). 1st ed. Boston: Addison-Wesley/Prentice-Hall.

[12] Curtin University.(AU). Worksheet 05(str_echo.c): Threads. In Advanced Computer Communications (Semester 2 2018 Sri Lanka Inst Info Tech - INT[1]). Retrieved from https://lms.curtin.edu.au

[13] Stevens, W. Richard, Bill Fenner, and Andrew M Rudoff. 2012. UNIX Network Programming(Chapter 30) : serv07. 1st ed. Boston [etc.]: Addison-Wesley.

[14] "Get The Current Time In C". 2018. Stack Overflow. https://stackoverflow.com/questions/5141960/get-the-current-time-in-c.

[15] "Socket Function Reference : Getpeername". 2018. Support.Sas.Com. https://support.sas.com/documentation/onlinedoc/sasc/doc750/html/lr2/zeername.htm.

[16] "How To Copy A Char Array In C?". 2018. Stack Overflow. https://stackoverflow.com/questions/16645583/how-to-copy-a-char-array-in-c.

[17] "String Array With Garbage Character At End". 2018. Stack Overflow. https://stackoverflow.com/questions/270708/string-array-with-garbage-character-at-end.

[18] "Concatenate Char Array In C". 2018. Stack Overflow. https://stackoverflow.com/questions/2218290/concatenate-char-array-in-c.

[19] "How Can I Check If A Directory Exists?". 2018. Stack Overflow. https://stackoverflow.com/questions/12510874/how-can-i-check-if-a-directory-exists.

[20] "How To Remove The Character At A Given Index From A String In C?". 2018. Stack Overflow. https://stackoverflow.com/questions/5457608/how-to-remove-the-character-at-a-given-index-from-a-string-in-c.

[21] "The Free Knowledge-Sharing Platform For Technology". 2018. Tech.Io. https://tech.io/playgrounds/14213/how-to-play-with-strings-in-c/string-split.

[22] "Create File Inside A Directory (C)". 2018. Stack Overflow. https://stackoverflow.com/questions/22949500/create-file-inside-a-directory-c.

[23] "What's The Best Way To Check If A File Exists In C? (Cross Platform)". 2018. Stack Overflow. https://stackoverflow.com/questions/230062/whats-the-best-way-to-check-if-a-file-exists-in-c-cross-platform?fbclid=IwAR1fhLAlTSsrylzKsUEIaumVX8knIwL5Pk4keMKAFbwXkqF0YMuLgSZ_IWo.

[24] "Return To The Beginning Of An If Statement". 2018. Stack Overflow. https://stackoverflow.com/questions/26368697/return-to-the-beginning-of-an-if-statement?fbclid=IwAR3GRZeoARFQ7v_utXs981MfYz1KUc4SzLB-cCFEfNmCFGD8jGB9HD23pRA.

[25] Curtin University.(AU). Worksheet 04: I/O Models. In Advanced Computer Communications (Semester 2 2018 Sri Lanka Inst Info Tech - INT[1]). Retrieved from https://lms.curtin.edu.au

[26] Stevens, W. Richard, Bill Fenner, and Andrew M Rudoff. 1999. UNIX Network Programming. Chapter 5. TCP Client/Server Example(lib/signal.c). 1st ed. Boston: Addison-Wesley/Prentice-Hall.