Capgemini

[redacted]

High Level Design & Low Level Design

Document Control :

Project Revision History

Date	Version	Author	Brief Description of Changes	Approver Signature
19.10.2022	1.0	Group 4		

Index

1. Introduction

The introduction of the High Level Design (HLD) and Low Level Design (LLD) provides an overview of the entire document with purpose, intended audience, scope and key objectives. The purpose of this High Level Design (HLD) Document is to add the necessary detail to the **Remote Backup Utility Application's** description to represent a suitable model for coding. This document is also intended to help detect contradictions prior to coding, and can be used as a reference manual for how the modules interact at a high level. This application will manage backing up files from client to the server side.

Project Purpose

The purpose of this project is to make backing up of files simple and elegant. The client side code is very simple and based on options it performs various tasks. It can perform full, incremental, versioned and scheduled backups

Intended Audience

This document is intended to be read by clients, designers , program and solution teams.
The intended audience for this application are anyone who want to backup their files remotely.

Project Scope

This project helps user who want to backup their files to a particular server. The user and perform any of the 4 tasks for both files and directories i.e full backup incremental backup versioned backup or scheduled backup.

Key Project Objectives

- Select the option for type of the backup
- Connect with the server
- Read files from the client
- Send the data to the server
- Write the received data from client in the backup folder

2. Design Overview

Model City Vaccination Drive comprises of following modules:

Name of the Module	Sending files from client
Handled by	Sriram Repaka
Description	A function to read files and send the data of the file to the server

Name of the Module	Reading the arguments and formatting the filenames at client
Handled by	Sriram Repaka
Description	Reading the arguments and making the file path

Name of the Module	Full backup client
Handled by	Shagir Husain
Description	This module is responsible for full backup of files and directories

Name of the Module	Incremental backup client
Handled by	Prinshu Raj
Description	This module checks if the particular files have been modified since last backup and backups only it has been updated. For both files and directories

Name of the Module	Versioned backup client
Handled by	Sarthak Srivasthava
Description	This module reads files and sends them to server.

Name of the Module	Sheduled Backup client
Handled by	Sriram Repaka
Description	This module takes user input creates a cronjob

Name of the Module	Full backup server
Handled by	Shagir Husain
Description	Writes the received files to the backup folder

Name of the Module	Incremental backup server
Handled by	Aryan Tiwari
Description	This simply updates the received files in the backup folder

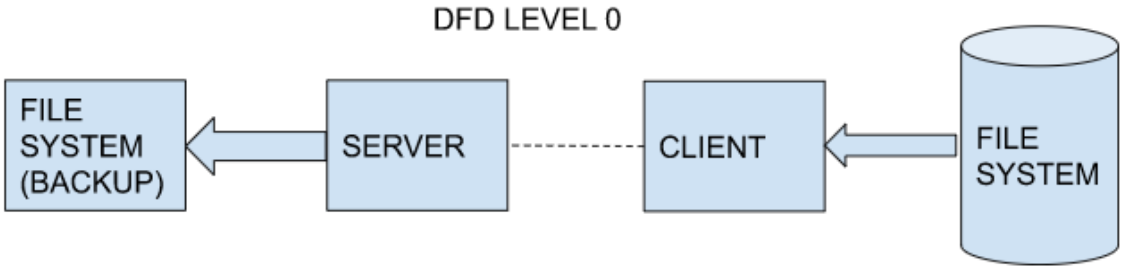
Name of the Module	Versioned Backup server
Handled by	Tushar Gautam
Description	Makes a copy of the files in the backup folder with time appended to the filename

Name of the Module	Scheduled Backup server
Handled by	Sriram Repaka
Description	Acknowledges the user that a scheduled backup has been added.

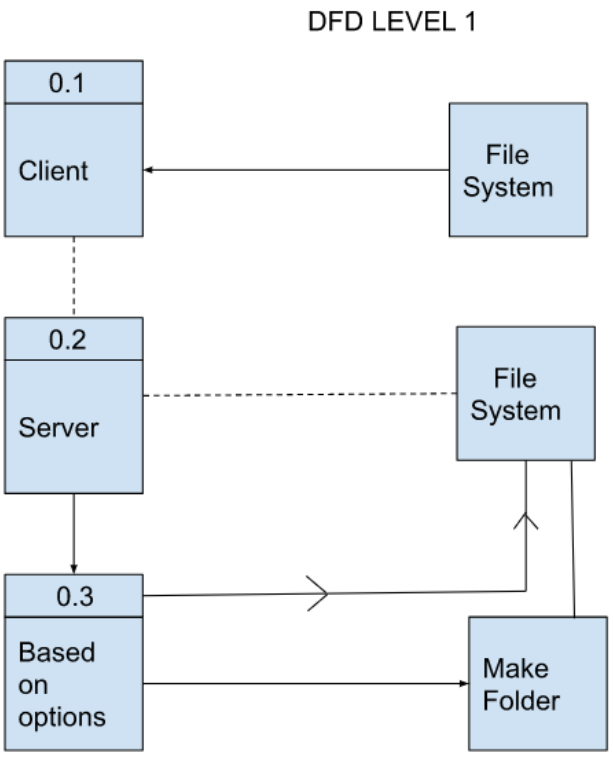
3. Detailed System Design

Data Flow Diagram

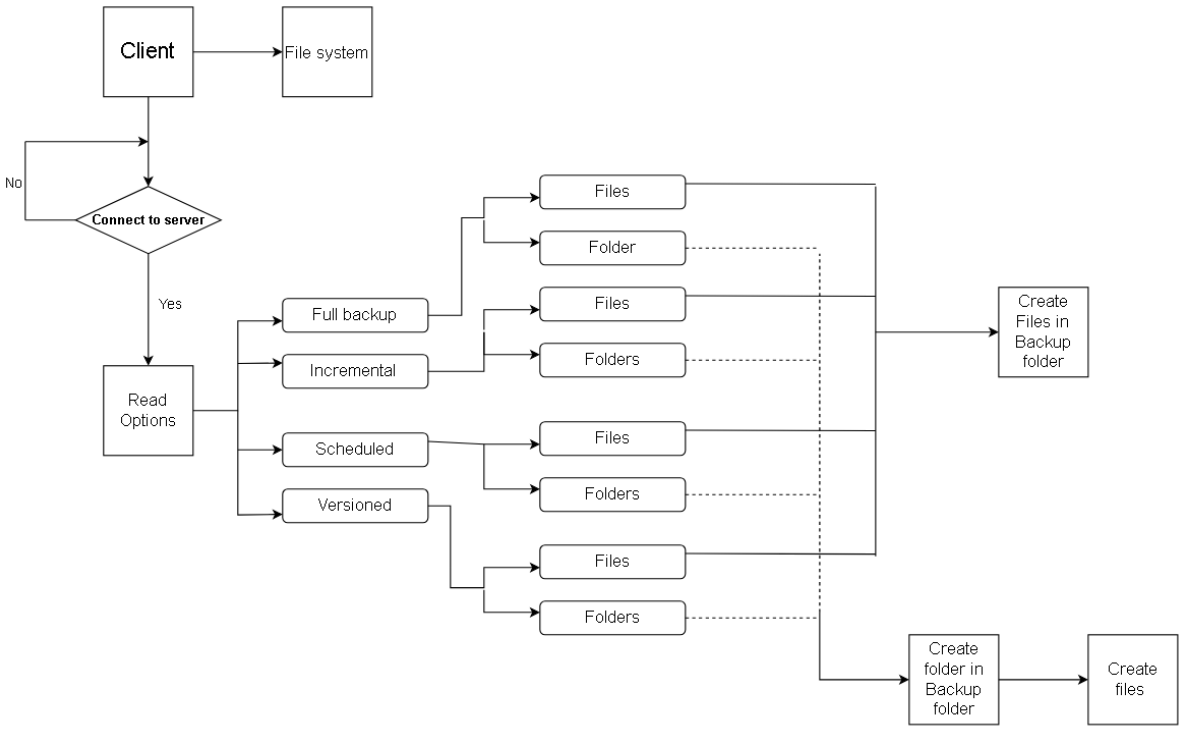
3.1.1. Level 0



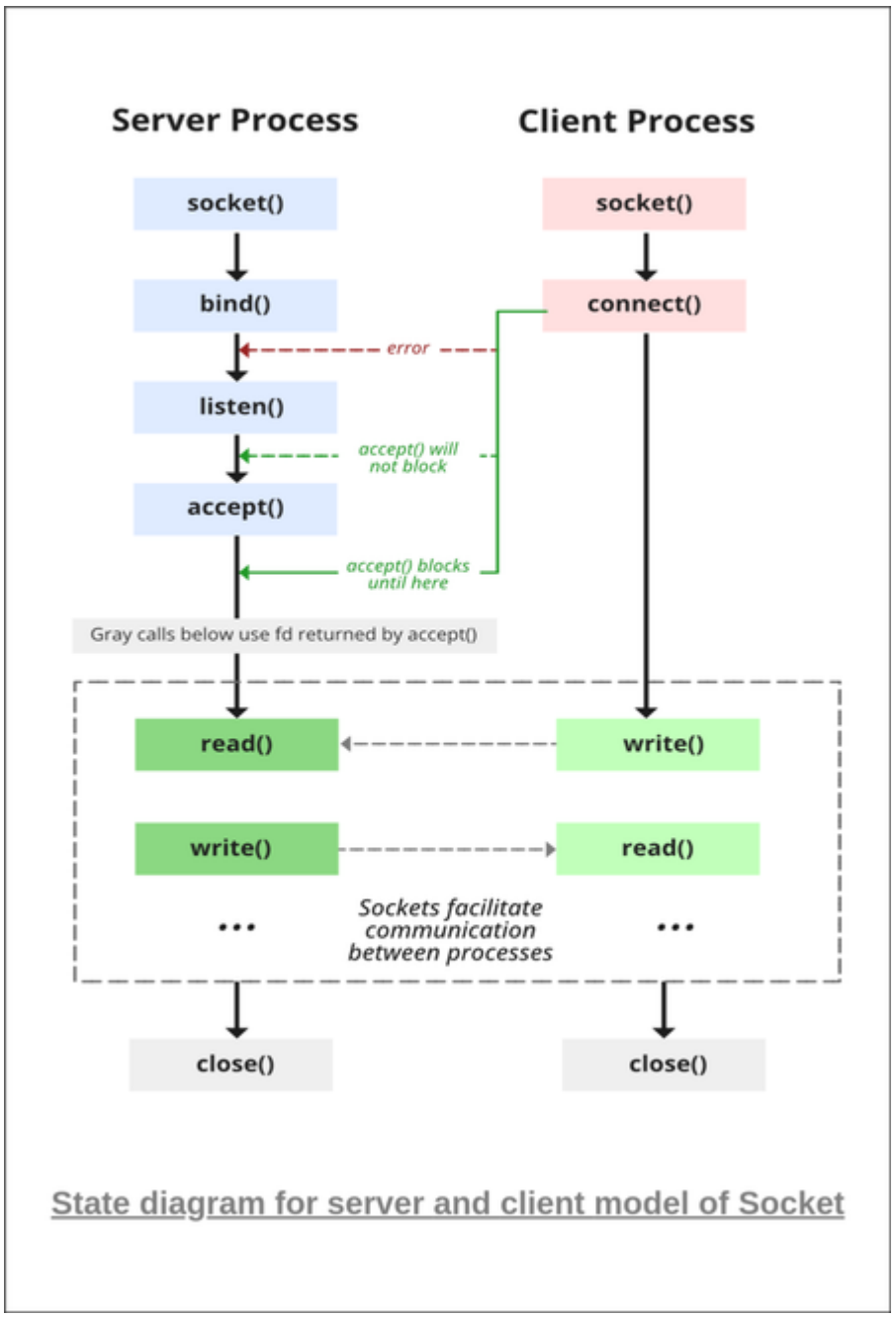
3.1.2. Level 1



Flowcharts



Usecase Diagram



State diagram for server and client model of Socket

Pseudocode

Client

Pesudocode for Remote Backup Utility

First function: MAIN FUNCTION

-----Creating Sockets to send and recieve information -----

----main function begins-----
BEGIN

Reading current time storing the MON & DOM in a string

CREATING A SOCKET
Creatinga socket and acknowledging the user
CONNECTING TO THE SERVER
checking weather the socket is connected or not
IF successful
acknowledge the user
ELSE
END

READ argc, argv
//validating arguments
//assigning argv[1] argv[2] to option and directory
//sending argc argv[1] argv[2] to the server

STARTING A IF ELSE LADDER FOR ALL THE OPTIONS
IF OPTION == -ff
IF ARGC != number THEN
PRINT "error!! enter correct number of arguments"
END
ELSE
reading argv[3] argv[4] .... argv[argc-1]
sending the argv[]'s (filenames) to server
making the file pathname by conatinating argv[]'s and Directory
calling the SEND_FILE() with socket name and filename

ELSE IF OPTION == -fd

```
        IF ARGC != number THEN
                PRINT "error!! enter correct number of arguments"
        END
      ELSE
                reading argv[2] argv[3] .... argv[argc-1]
                SENDing the directory paths to the server
                OPENING the argv[]'s(directories)
                READING all the filenames and concatinating to directories to
create filepath
                calling the SEND_FILE() with socket name and filepath

ELSE IF OPTION == -if
        IF ARGC != number THEN
                PRINT "error!! enter correct number of arguments"
        END
      ELSE
                reading argv[3] argv[4] .... argv[argc-1]
                making the file pathname by conatinating argv[]'s and
Directory
                check wheather the file has been modified today
                IF yes
                        calling the SEND_FILE() with socket name and
filename


ELSE IF OPTION == -id
        IF ARGC != number THEN
                PRINT "error!! enter correct number of arguments"
        END
      ELSE
                reading argv[3] argv[4] .... argv[argc-1]
                SENDing the directory paths to the server
                OPENING the directory and reading all the filenames
                making the file pathname by conatinating Directories and
filenames
                check weather the file has been modified or not
                IF yes
                        calling the SEND_FILE() with socket name and
filename


ELSE IF OPTION == -vf
        IF ARGC != number THEN
```

```
            PRINT "error!! enter correct number of arguments"
        END
      ELSE
            reading argv[2] argv[3] .... argv[argc-1]
            making the file pathname by conatinating argv[]'s and
Directory
            calling the SEND_FILE() with socket name and filename

ELSE IF OPTION == -vd
        IF ARGC != number THEN
            PRINT "error!! enter correct number of arguments"
        END
      ELSE
            reading argv[3] argv[4] .... argv[argc-1]
            SENDing the directory paths to the server
            OPENING the directory and reading all the filenames
            making the file pathname by conatinating Directories and
filenames
            calling the SEND_FILE() with name and socketname

ELSE IF OPTION == -sf
        IF ARGC != number THEN
            PRINT "error!! enter correct number of arguments"
        END
      ELSE
            reading argv[2] argv[3] .... argv[argc-1]
            creating a cronjob based on argv's
            concayinating the cronjob to the config file
            copying the cronjob schedule "* * * * *" to string
            SENDing the schedule string to the server

ELSE IF OPTION == -sd
        IF ARGC != number THEN
            PRINT "error!! enter correct number of arguments"
        END
      ELSE
            reading argv[3] argv[4] .... argv[argc-1]
            creating a cronjob based on argv's
            concayinating the cronjob to the config file
            copying the cronjob schedule "* * * * *" to string
            SENDing the schedule string to the server

----main function ends----
```

----function definitions----

FUNCTION SEND_FILE
PASS IN: FILENAME, SOCKETNAME
PASS OUT: NOTHING
    OPEN FILE USEING the FILENAME
    read the contents of the file
    SEND the contents of the file to the server using teh SOCKETNAME
    FLIGHT_LIST -> TAIL is NULL
ENDFUNCTION

Validations

- Users must enter the IP address and PORT numbers in the client and server codes
- Users must enter the backup directory in the server code
- The users must enter valid filenames and directories.

4. Detailed Features and Requirements

Functional Requirements

4.1.1. Full backup for files

full():

(client)

Firstly we check whether the option is -ff or not and whether more than 4 arguments have been passed or not. The directory of the files specified is passed as the 3rd argument and files are specified from the 4th argument onwards. Once we check the said conditions we start by sending the filenames to the server and then we concatenate the directories and the filename then we call the send_file() function.

(server)

Coming to the server side we firstly receive the filenames then we concatenate the backup directory and the filename essentially creating the path of the backup file we also add the backup_of tag to the filename received from the client. After creating the path we call the write_file() function to write the data received from the client using the send_file() function.

4.1.2. Full backup for folders

full_dir():

(client)

Firstly we check whether the option is -fd or not and whether more than 3 arguments have been passed or not. The directory is passed from the 3rd argument. Once we check the said conditions we start by sending the Directory paths to the server and then using opendir() and readdir() we go through all the filenames in the specified directories. Then we concatenate the directory and the filenames and we call the send_file() function.
(server)
Coming to the server side we firstly receive the Directory paths, then we concatenate the backup directory and the paths received. Now we use mkdir() to create the sub-directory in the backup directory essentially creating the directories for the files we are going to receive from the client. After creating the Directories simply repeat what we did for full backup of files.

4.1.3. Incremental backup for files

incremental():

(client)
Firstly we check whether the option is -if or not and whether more than 4 arguments have been passed or not. The directory of the files specified is passed as the 3rd argument and files are specified from the 4th argument onwards. Once we check the said conditions we start by concatenating the directories and the filename then using filestat we create a file status variable then using st_ctime we get the time when the file was modified and it it was modifies today if it was modifies today then we first send the filename to server and then we call the send_file() function to send only that particular file.

(server)
Coming to the server side we firstly receive the filenames then we concatenate the backup directory and the filename which will be creating the path of the backup file we also add the "backup_of" tag to the filename received from the client. After creating the path we call the write_file() function to write the data received from the client using the send_file() function.

4.1.4. Incremental backup for folders

incremental_dir():

(client)
Firstly we check whether the option is -id or not and whether more than 3 arguments have been passed or not. The directory is passed from the 3rd argument. Once we check the said conditions we start by sending the Directory paths to the server and then using opendir() and readdir() we go through all the filenames in the specified directories and check weather they have been updated today. If yes then we concatenate the directory and the filenames and we call the send_file() function.

(server)
Coming to the server side we firstly receive the Directory paths, then we concatenate the backup directory and the paths received. Now we use mkdir() to create the sub-directory in the backup directory which will be creating the directories for the files we are going to receive from the client. After creating the Directories simply repeat what we did for full/incremental backup of files

4.1.5. Versioned backup for files

versioned():

(client)

Firstly we check whether the option is -ff or not and whether more than 4 arguments have been passed or not. The directory of the files specified is passed as the 3rd argument and files are specified from the 4th argument onwards. Once we check the said conditions we start by sending the filenames to the server and then we concatenate the directories and the filename then we call the send_file() function.

(server)

Coming to the server side we firstly receive the filenames then we concatenate the present time till the minutes to the filename then we concatenate the backup directory and the filename essentially creating the path of the backup file we also add the "backup_at" and "_of" tag to the filename received from the client. After creating the path we call the write_file() function to write the data received from the client using the send_file() function.

4.1.6. Versioned backup for folders

versioned_dir():

(client)

Firstly we check whether the option is -vd or not and whether more than 3 arguments have been passed or not. Then directory is passed as the 3rd argument. Once we check the said conditions we start by sending the Directory paths to the server and then using opendir() and readdir() we go through all the filenames in the specified directories. Then we concatenate the directory and the filenames and we call the send_file() function.

(server)

Coming to the server side we firstly receive the Directory paths, then we concatenate the backup directory and the paths received. Now we use mkdir() to create the sub-directory in the backup directory essentially creating the directories for the files we are going to receive from the client. After creating the Directories we simply repeat what we did for versioned backup for files. We add the time to the filename and create the versioned file of the received files.

4.1.7. Scheduled backup for files

scheduled():

(client)

We scan the arguments first which include the timing of the backup i.e the schedule of the back up in the form of "mm hh dom mon dow" the we start creating a string for the cronjob we create the cronjob such that at the scheduled time the we cd to the directory where our client is and then we run the client executable file with -ff option with the files specified. We then append this cron job to the config file. I have to mention that the config file already has a cronjob that updates the contab every hour. The client also sends the cronjob time to the server

(Server)

Server receives the cornjob time and it appends its config file which tells when the server should run so as to be available for the client at that time.

4.1.8. Scheduled backup for folders

scheduled_dir():

(client)

We scan the arguments first which include the timing of the backup i.e the schedule of the back up in the form of "mm hh dom mon dow" the we start creating a string for the cronjob we create the cronjob such that at the scheduled time the we cd to the directory where our client is and then we run the client executable file with -fd option with the directories specified. We then append this cron job to the config file.

(Server)

Server receives the cornjob time and it appends its config file which telling when the server should run so as to be available for the client at that time.

5. Tools Report

Gcov:

Full backup:

```
└─$ gcov a-client.c
File 'client.c'
Lines executed:21.31% of 244
Creating 'client.c.gcov'

Lines executed:21.31% of 244
```

```
        -:   83:
        -:   84:        //Full backup for **files**
        -:   85:        //./client1.c -ff dir file1 file2 file3......
        1:   86:        if(strcmp(option,"-ff") == 0){
        -:   87:
        1:   88:                if(argc<4)
        -:   89:                {
    #####:   90:                        printf(" Invalid Usage : No file name
 entered!!!\n");
    #####:   91:                        printf(" Usage: ./client1.c -ff dir f
ile1 file2 file3......");
    #####:   92:                        exit(1);
        -:   93:                }
        -:   94:
        3:   95:                for(int i = 3;i < argc;i++){
        -:   96:
        -:   97:                        int k;
        -:   98:                        char temp[30];
        2:   99:                        strcpy(temp,argv[i]);
        2:  100:                        send(sockfd,temp,sizeof(temp),0);
        2:  101:                        printf("%s\n",temp);
        2:  102:                        strcpy(temp,filename);
        2:  103:                        strcat(temp,argv[i]);
        -:  104:                        //strcpy(filename,temp);
        2:  105:                        printf("%s\n",temp);
        2:  106:                        send_file(temp,sockfd);
        -:  107:
        -:  108:                }
        -:  109:
        -:  110:        }
```

Incremental Backup

```
┌──(kali㉿kali)-[~/RBU]
└─$ gcov incremental-client.c
File 'client.c'
Lines executed:28.69% of 244
Creating 'client.c.gcov'

Lines executed:28.69% of 244
```

Scheduled Backup

```
┌──(kali㉿kali)-[~/RBU]
└─$ gcc -fprofile-arcs -ftest-coverage -o scheduled client.c

┌──(kali㉿kali)-[~/RBU]
└─$ ./scheduled -sf House/tcp "* * * * *" nile1.txt nile2.txt
Last backup at : Oct 19
[+]Server socket created successfully.
[+]Connected to Server.

* * * * * cd /home/kali/RBU && ./client -ff House/tcp nile1.txt nile2.txt

[+]Closing the connection.

┌──(kali㉿kali)-[~/RBU]
└─$ gcov scheduled-client.c
File 'client.c'
Lines executed:21.31% of 244
Creating 'client.c.gcov'

Lines executed:21.31% of 244
```

Versioned Backup

```
┌──(kali㉿kali)-[~/RBU]
└─$ gcov versioned-client.c
File 'client.c'
Lines executed:22.95% of 244
Creating 'client.c.gcov'

Lines executed:22.95% of 244
```

Gprof

```
┌──(kali㉿kali)-[/media/…/Airline Reservation/CUT/code/src]
└─$ gprof ./a.out
Flat profile:

Each sample counts as 0.01 seconds.
 no time accumulated

  %   cumulative   self              self     total
 time   seconds   seconds    calls  Ts/call  Ts/call  name
  0.00      0.00     0.00       19     0.00     0.00  allocate_tickets
  0.00      0.00     0.00       14     0.00     0.00  insert_at_end
  0.00      0.00     0.00       12     0.00     0.00  create
  0.00      0.00     0.00       12     0.00     0.00  insert_flight
  0.00      0.00     0.00        1     0.00     0.00  create_flight_LL
  0.00      0.00     0.00        1     0.00     0.00  read_flight

 %         the percentage of the total running time of the
time       program used by this function.
```

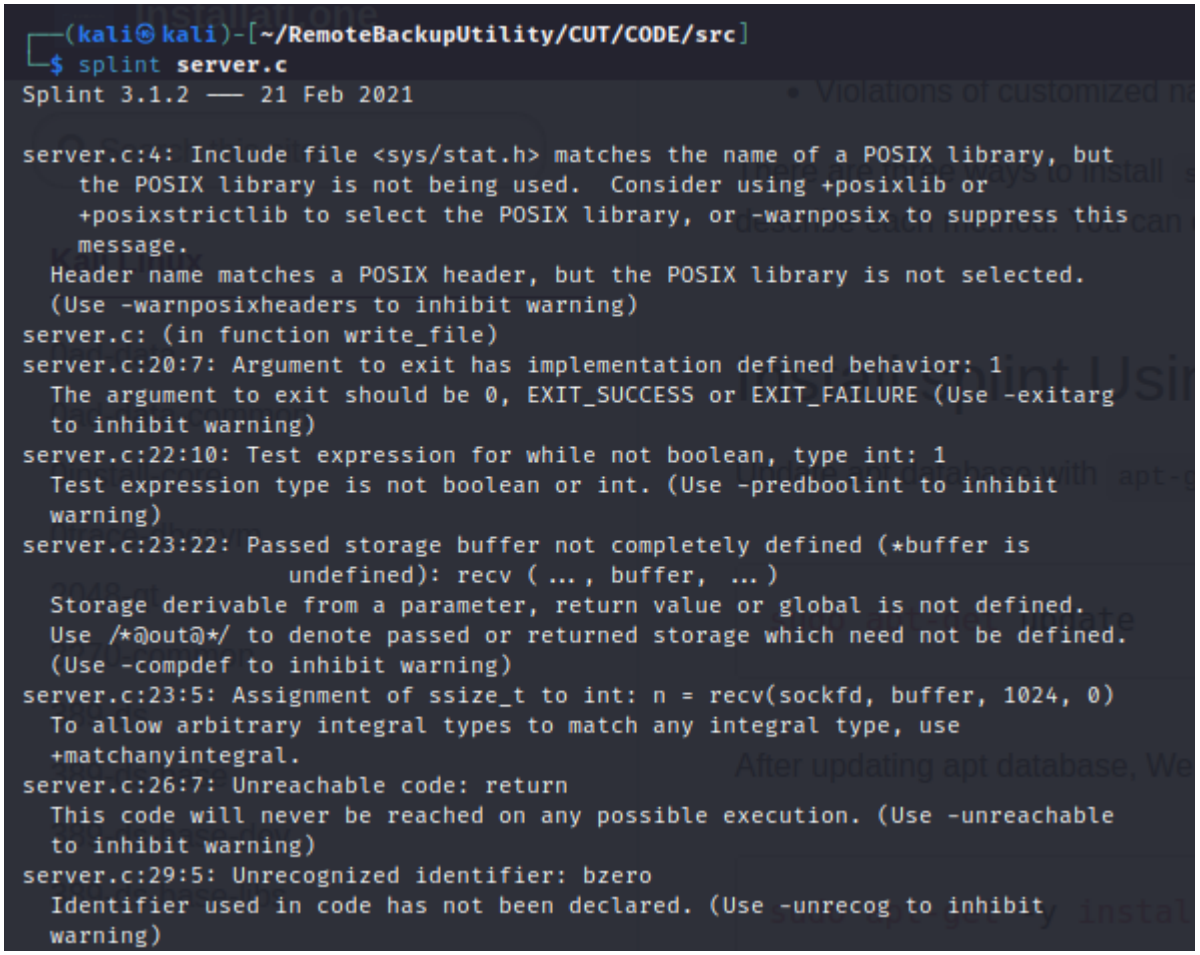
```
                     Call graph (explanation follows)


granularity: each sample hit covers 2 byte(s) no time propagated

index % time    self  children    called     name
                0.00    0.00      19/19          read_customer_data [20]
[1]      0.0    0.00    0.00      19         allocate_tickets [1]
                0.00    0.00      14/14          insert_at_end [2]
-----------------------------------------------
                0.00    0.00      14/14          allocate_tickets [1]
[2]      0.0    0.00    0.00      14         insert_at_end [2]
-----------------------------------------------
                0.00    0.00      12/12          insert_flight [4]
[3]      0.0    0.00    0.00      12         create [3]
-----------------------------------------------
                0.00    0.00      12/12          read_flight [6]
[4]      0.0    0.00    0.00      12         insert_flight [4]
                0.00    0.00      12/12          create [3]
-----------------------------------------------
                0.00    0.00       1/1           main [18]
[5]      0.0    0.00    0.00       1         create_flight_LL [5]
-----------------------------------------------
```

Splint

server



```
┌──(kali㉿kali)-[~/RemoteBackupUtility/CUT/CODE/src]
└─$ splint client.c
Splint 3.1.2 --- 21 Feb 2021

client.c:2: Include file <unistd.h> matches the name of a POSIX library, but
    the POSIX library is not being used.  Consider using +posixlib or
    +posixstrictlib to select the POSIX library, or -warnposix to suppress this
    message.
  Header name matches a POSIX header, but the POSIX library is not selected.
  (Use -warnposixheaders to inhibit warning)
client.c:6: Include file <sys/stat.h> matches the name of a POSIX library, but
    the POSIX library is not being used.  Consider using +posixlib or
    +posixstrictlib to select the POSIX library, or -warnposix to suppress this
    message.
client.c:7: Include file <dirent.h> matches the name of a POSIX library, but
    the POSIX library is not being used.  Consider using +posixlib or
    +posixstrictlib to select the POSIX library, or -warnposix to suppress this
    message.
client.c: (in function send_file)
client.c:14:21: Initializer block for data has 1 element, but declared as char
                  [1024]: 0
  Initializer does not define all elements of a declared array. (Use
  -initallelements to inhibit warning)
client.c:14:22: Initial value of data[0] is type int, expects char: 0
  Types are incompatible. (Use -type to inhibit warning)
client.c:15:30: Format argument 2 to printf (%ld) expects long int gets size_t:
                  strlen(filename)
  To allow arbitrary integral types to match long unsigned, use +longintegral.
   client.c:15:16: Corresponding format code
client.c:23:10: Argument to exit has implementation defined behavior: 1
  The argument to exit should be 0, EXIT_SUCCESS or EXIT_FAILURE (Use -exitarg
  to inhibit warning)
client.c:29:12: Argument to exit has implementation defined behavior: 1
client.c:31:5: Unrecognized identifier: bzero
  Identifier used in code has not been declared. (Use -unrecog to inhibit
  warning)
client.c:33:10: Possibly null storage data passed as non-null param:
                  strcpy (data, ...)
  A possibly null pointer is passed as a parameter corresponding to a formal
  parameter with no /*@null@*/ annotation.  If NULL may be used for this
  parameter, add a /*@null@*/ annotation to the function parameter declaration.
  (Use -nullpass to inhibit warning)
   client.c:14:21: Storage data may become null
client.c:34:3: Return value (type ssize_t) ignored: send(sockfd, dat...
  Result returned by function call is not used. If this is intended, can cast
  result to (void) to eliminate message. (Use -retvalother to inhibit warning)
client.c: (in function main)
client.c:46:3: Suspect modification of observer las_backup[]:
                 las_backup[strlen(las_backup) - 1] = '\0'
  Storage declared with observer is possibly modified. Observer storage may not
  be modified. (Use -modobserver to inhibit warning)
   client.c:45:16: Storage las_backup[] becomes observer
client.c:58:10: Argument to exit has implementation defined behavior: 1
client.c:63:3: Assignment of int to in_port_t: server_addr.sin_port = port
  To allow arbitrary integral types to match any integral type, use
  +matchanyintegral.
client.c:66:61: Function connect expects arg 3 to be socklen_t gets size_t:
                  sizeof((server_addr))
client.c:69:10: Argument to exit has implementation defined behavior: 1
client.c:75:2: Return value (type ssize_t) ignored: send(sockfd, &k, ...
```

Valgrind

```
┌──(kali㉿kali)-[~/RemoteBackupUtility/CUT/CODE/obj]
└─$ valgrind ./backup.exe
==81401== Memcheck, a memory error detector
==81401== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al.
==81401== Using Valgrind-3.19.0 and LibVEX; rerun with -h for copyright info
==81401== Command: ./backup.exe
==81401==
Last backup at : Oct 19
[+]Server socket created successfully.
[-]Error in socket: Connection refused
==81401==
==81401== HEAP SUMMARY:
==81401==     in use at exit: 0 bytes in 0 blocks
==81401==   total heap usage: 12 allocs, 12 frees, 9,418 bytes allocated
==81401==
==81401== All heap blocks were freed -- no leaks are possible
==81401==
==81401== For lists of detected and suppressed errors, rerun with: -s
==81401== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

6. Integration Testing

Test 1

Correct Input

```
┌──(kali㉿kali)-[~/RBU]
└─$ ./client -ff House/tcp nile1.txt nile2.txt nile3.txt
Last backup at : Oct 19
[+]Server socket created successfully.
[+]Connected to Server.

nile1.txt
../House/tcp/nile1.txt
nile2.txt
../House/tcp/nile2.txt
nile3.txt
../House/tcp/nile3.txt
[+]Closing the connection.
```

Incorrect Input

```
└─$ ./client -ff House/tcp nile1.txt nile2.txt nile4.txt
Last backup at : Oct 19
[+]Server socket created successfully.
[+]Connected to Server.

nile1.txt
../House/tcp/nile1.txt
nile2.txt
../House/tcp/nile2.txt
nile4.txt
../House/tcp/nile4.txt
[-]Error in reading file.: No such file or directory
```

Test 2

Incorrect input

```
└─$ ./client -fd /home/kali/temp0 /home/kali/temp4
Last backup at : Oct 19
[+]Server socket created successfully.
[+]Connected to Server.

..//home/kali/temp0
/home/kali/temp0
/home/kali/temp4
/home/kali/temp0/temp01.txt
/home/kali/temp0/temp02.txt
/home/kali/temp0/temp03.txt
```

Correct Input

```
└─$ ./client -fd /home/kali/temp0 /home/kali/temp1 /home/kali/temp2
Last backup at : Oct 19
[+]Server socket created successfully.
[+]Connected to Server.

..//home/kali/temp0
/home/kali/temp0
/home/kali/temp1
/home/kali/temp2
/home/kali/temp0/temp01.txt
/home/kali/temp0/temp02.txt
/home/kali/temp0/temp03.txt
/home/kali/temp1/temp01.txt
/home/kali/temp1/temp02.txt
/home/kali/temp1/temp03.txt
/home/kali/temp2/temp01.txt
/home/kali/temp2/temp02.txt
/home/kali/temp2/temp03.txt
[+]Closing the connection.
```

Test 3

Incorrect Input

```
└─$ ./client -if House/tcp nile1.txt nile2.txt nile10.txt
Last backup at : Oct 19
[+]Server socket created successfully.
[+]Connected to Server.

../House/tcp/nile1.txt
last_backup : Oct 19
File changed at : Oct 19
Backup Needed
10 50

../House/tcp/nile2.txt
last_backup : Oct 19
File changed at : Oct 14
```

```
./server
Oct_19_14:10
[+]Server socket created successfully.
[+]Binding successfull.
[+]Listening....
go
../backup/backup_of_nile1.txt
stopped
```

Correct input

```
└─$ ./client -if House/tcp nile1.txt nile2.txt nile3.txt
Last backup at : Oct 19
[+]Server socket created successfully.
[+]Connected to Server.

../House/tcp/nile1.txt
last_backup : Oct 19
File changed at : Oct 19
Backup Needed
10 50

../House/tcp/nile2.txt
last_backup : Oct 19
File changed at : Oct 14

../House/tcp/nile3.txt
last_backup : Oct 19
File changed at : Oct 14

[+]Closing the connection.
```

Test 4

Correct Input

```
└─$ ./client -id /home/kali/temp0 /home/kali/temp1
Last backup at : Oct 19
[+]Server socket created successfully.
[+]Connected to Server.

/home/kali/temp0
/home/kali/temp1

/home/kali/temp0
last backup : Oct 19
Directory changed : Oct 19
backup needed
/home/kali/temp0/temp01.txt
/home/kali/temp0/temp02.txt
/home/kali/temp0/temp03.txt

/home/kali/temp1
last backup : Oct 19
Directory changed : Oct 16
[+]Closing the connection.
```

Incorrect input

```
└─$ ./client -id /home/kali/temp0 /home/kali/temp10
Last backup at : Oct 19
[+]Server socket created successfully.
[+]Connected to Server.

/home/kali/temp0
/home/kali/temp10

/home/kali/temp0
last backup : Oct 19
Directory changed : Oct 19
backup needed
/home/kali/temp0/temp01.txt
/home/kali/temp0/temp02.txt
/home/kali/temp0/temp03.txt
```

Test 5

Incorrect input

```
└─$ ./client -vf home/kali/House/tcp nile1.txt nile2.txt nile3.txt
Last backup at : Oct 19
[+]Server socket created successfully.
[+]Connected to Server.

nile1.txt
../home/kali/House/tcp/nile1.txt
[-]Error in reading file.: No such file or directory
```

Correct Input

```
└─$ ./client -vf /House/tcp nile1.txt nile2.txt nile3.txt
Last backup at : Oct 19
[+]Server socket created successfully.
[+]Connected to Server.

nile1.txt
..//House/tcp/nile1.txt
nile2.txt
..//House/tcp/nile2.txt
nile3.txt
..//House/tcp/nile3.txt
[+]Closing the connection.
```

```
./server
Oct_19_14:17
[+]Server socket created successfully.
[+]Binding successfull.
[+]Listening....
nile1.txt
../backup/backup_at_Oct_19_14:17_of_nile1.txt
nile2.txt
../backup/backup_at_Oct_19_14:17_of_nile2.txt
nile3.txt
../backup/backup_at_Oct_19_14:17_of_nile3.txt
```

Test 6

Incorrect input

```
└─$ ./client -vd /home/kali/temp0 /home/kali/test1
Last backup at : Oct 19
[+]Server socket created successfully.
[+]Connected to Server.

.. //home/kali/temp0
/home/kali/temp0
/home/kali/test1
/home/kali/temp0/temp01.txt
/home/kali/temp0/temp02.txt
/home/kali/temp0/temp03.txt
```

```
Couldnt not make file
make: *** [Makefile:10: run] Error 1
```

Correct Input

```
└─$ ./client -vd /home/kali/temp0 /home/kali/test
Last backup at : Oct 19
[+]Server socket created successfully.
[+]Connected to Server.

.. //home/kali/temp0
/home/kali/temp0
/home/kali/test
/home/kali/temp0/temp01.txt
/home/kali/temp0/temp02.txt
/home/kali/temp0/temp03.txt
/home/kali/test/cron_bkp
/home/kali/test/test.txt
/home/kali/test/test1
/home/kali/test/test
/home/kali/test/test1.c
/home/kali/test/test.c
[+]Closing the connection.
```

```
./server
Oct_19_14:19
[+]Server socket created successfully.
[+]Binding successfull.
[+]Listening....
../backup
/home/kali/backup/home/kali/temp0
/home/kali/backup/home/kali/test
../backup/home/kali/temp0/temp01.txt
../backup/home/kali/temp0/temp01_Oct_19_14:19.txt
../backup/home/kali/temp0/temp02.txt
../backup/home/kali/temp0/temp02_Oct_19_14:19.txt
../backup/home/kali/temp0/temp03.txt
../backup/home/kali/temp0/temp03_Oct_19_14:19.txt
../backup/home/kali/test/cron_bkp
../backup/home/kali/test/cron_bkp_Oct_19_14:19
../backup/home/kali/test/test.txt
../backup/home/kali/test/test_Oct_19_14:19.txt
../backup/home/kali/test/test1
../backup/home/kali/test/test1_Oct_19_14:19
../backup/home/kali/test/test
../backup/home/kali/test/test_Oct_19_14:19
../backup/home/kali/test/test1.c
../backup/home/kali/test/test1_Oct_19_14:19.c
../backup/home/kali/test/test.c
../backup/home/kali/test/test_Oct_19_14:19.c
stopped
```

Test 7

Inorrect Input

```
└─$ ./client -sf "1 * * * *" nile1.txt nile2.txt nile3.txt
Last backup at : Oct 19
[+]Server socket created successfully.
[+]Connected to Server.

nile1.txt cd /home/kali/RBU && ./client -ff 1 * * * * nile2.txt nile3.txt

[+]Closing the connection.
```

Correct Input

```
└─$ ./client -sf House/tcp "1 * * * *" nile1.txt nile2.txt nile3.txt
Last backup at : Oct 19
[+]Server socket created successfully.
[+]Connected to Server.

1 * * * * cd /home/kali/RBU && ./client -ff House/tcp nile1.txt nile2.txt nile3.txt
```

Test 8

Correct Input

```
└─$ ./client -sd "30 * * * *" /home/kali/temp0 /home/kali/temp1 /home/kali/Desktop
Last backup at : Oct 19
[+]Server socket created successfully.
[+]Connected to Server.

30 * * * * cd /home/kali/RBU && ./client -fd  /home/kali/temp0 /home/kali/temp1 /home/kali/Desktop

[+]Closing the connection.
```

Incorrect Input

```
└─$ ./client -sd "30 * * * *"
Last backup at : Oct 19
[+]Server socket created successfully.
[+]Connected to Server.

 Invalid Usage : No folder name entered!!!
 Usage: /client -sd 'mm hh dom mon dow' dir1 dir2 dir3 .....
```

```
┌──(kali㉿kali)-[~/RBU]
└─$ ./client -kk home/temp0
Last backup at : Oct 19
[+]Server socket created successfully.
[+]Connected to Server.

Invalid Option please choose from any of the options mentioned below
-ff for full backup for files
-fd for full backup for Directories
-if for incremental backup for files
-id for incremental backup for Directories
-vf for Versioned backup for files
-vd for Versioned backup for Directories
-sf for Scheduled backup for files
-sd for Scheduled backup for Directories
[+]Closing the connection.
```

7. Requirement Traceability Matrix

Req	Design Mapping	Code Mapping		IT Mapping
RBU_01	3.1.1	full( )	{for files}	IT CASE 1
RBU_02	3.1.2	full_dir( )	{for directories}	IT CASE 2
RBU_03	3.1.3	incremental( )	{for files}	IT CASE 3
RBU_04	3.1.4	incremental_dir( )	{for directories}	IT CASE 4
RBU_05	3.1.5	versioned( )	{for files}	IT CASE 5
RBU_06	3.1.6	versioned_dir( )	{for directories}	IT CASE 6
RBU_07	3.1.7	scheduled( )	{for files}	IT CASE 7
RBU_08	3.1.8	scheduled_dir( )	{for directories}	IT CASE 8

8. Reference

https://idiotdeveloper.com/file-transfer-using-tcp-socket-in-c/

https://stackoverflow.com/questions/55765083/running-gcov-on-a-program-with-arguments

https://www.geeksforgeeks.org/socket-programming-cc/