

FTP APPLICATION

HLD & LLD DOCUMENT

SPRINT-2 IMPLEMENTATION

CASE STUDY-5



Index

1. Introduction	4
1.1 Intended audience	4
1.2 Project purpose	4
1.3 Key project objective	4
1.4 Project scope	4
2. Design overview	5
2.1 FTP Application	5
2.1.1 Client_server_Connection	5
2.1.2 User_Authentication	5
2.1.3 Start_At_Boot	6
2.1.4 Anonymous_User	6
2.1.5 Server_Concurrency	6
2.1.6 Authenticated_Users	7
2.1.7 PWD_Command	7
2.1.8 LS_Command	7
2.1.9 GET_Command	8
2.1.10 PUT_Command	8
2.1.11 BYE_Command	8
2.2 Design Objectives	9

3	Detailed System Design	9
3.1	Flowchart of application	10
3.2	Data Flow Diagrams(DFD)	11
4.	Tools Report	12
4.1	Valgrind	12
4.2	Strace	13
4.3	Gprof	14
5.	Unit Testing	19
5.1	Cunit	19
6.	Integration Testing	21
6.1	Server Starting at boot	21
6.2	Connection Established	22
6.3	Server Side concurrency	22
6.4	Client Authentication	23
6.5	Authenticated Users Commands	23
6.6	Anonymous	24
6.7	Invalid User	24

1. Introduction

The introduction of the software requirement specification provides an overview of the entire software. The entire SRS with overview description purpose, scope, tools used and basic description. The aim of this document is to gather, analyze and give an in-depth insight into the complete FTP application by defining the problem statement in detail. The detailed requirements of the FTP application is provided in this document.

1.1 Intended Audience: -

The intended audience for this application can be any client who wants to perform operations such as upload, download and view files in their remote working directory. The guest or mundane users can also use this application to upload any documents to their respective remote directory.

1.2 Project Purpose: -

It is a FTP application that refers to a process that involves the transfer of files between devices over a network. The process works when one party allows another to send or receive files over the Internet. Originally used as a way for users to communicate and exchange information between two physical devices.

In this application the user will be authenticated with the help of password to receive privileged access and permissions to view, upload and download files from the server. It also features an anonymous mode which any user can access to upload files to a specific file to the server while other functionalities are restricted.

1.3 Key Project Objectives: -

To transfer files from client to server and vice-versa through sockets using various commands (ls-list, put-upload, get-download, pwd - present working directory, bye-exit) with different login modes i.e authenticated user mode and anonymous/guest mode.

1.4 Project scope : -

This application can be very helpful to people FTP is an acronym for File Transfer Protocol. As the name suggests, FTP is used to transfer files between computers on a network/locally as well. You can use FTP to exchange files between computer accounts, transfer files between an account and a desktop computer, or access online software archives.

This application allows user authentication and download and upload privileges for authenticated users. It also features an anonymous mode which any user can access to upload files to a specific directory in the server while other functionalities are restricted.

2. Design Overview: -

FTP APPLICATION:

2.1.1

Name of the Module	Client_server_Connection
Handled by	
Description	This is the first requirement of the FTP application , where a connection between client and server is established using the port number and IP address.

2.1.2

Name of the Module	User_Authentication
Handled by	
Description	In this function the server will authenticate the user from a poll of registered user data kept in the “auth” file where all the usernames along with their passwords are kept. The user is placed into the appropriate directory after authentication.

2.1.3

Name of the Module	Start_At_Boot
Handled by	
Description	The FTP server is always set to enable & start from the time of boot , this makes the program run without needing to start the server every run.

2.1.4

Name of the Module	Anonymous_User
Handled by	
Description	This allows guest/unknown users to access the public directory where they can use commands such as pwd and put. The upload is done to the upload directory in the public domain.

2.1.5

Name of the Module	Server_Concurrency
Handled by	
Description	This application supports multiple clients to connect to the server and perform their operations smoothly.

2.1.6

Name of the Module	Authenticated_Users
Handled by	
Description	This is the core of the application that allows the authenticated users to browse, upload and download files from their respective directories in the server side using commands such as ls, pwd, get, put.

2.1.7

Name of the Module	PWD_Command
Handled by	
Description	This command lets both the anonymous user and the authenticated user to see their present working directory.

2.1.8

Name of the Module	LS_Command
Handled by	
Description	This command works only with authenticated users and allows them to see all the files in their working directory.

2.1.9

Name of the Module	GET_Command
Handled by	
Description	This command works only with authenticated and allows them to download files from server to client side.

2.1.10

Name of the Module	PUT_Command
Handled by	
Description	This command works with both the anonymous and authenticated users allowing them to upload files from client side to server side in their respective directories.

2.1.11

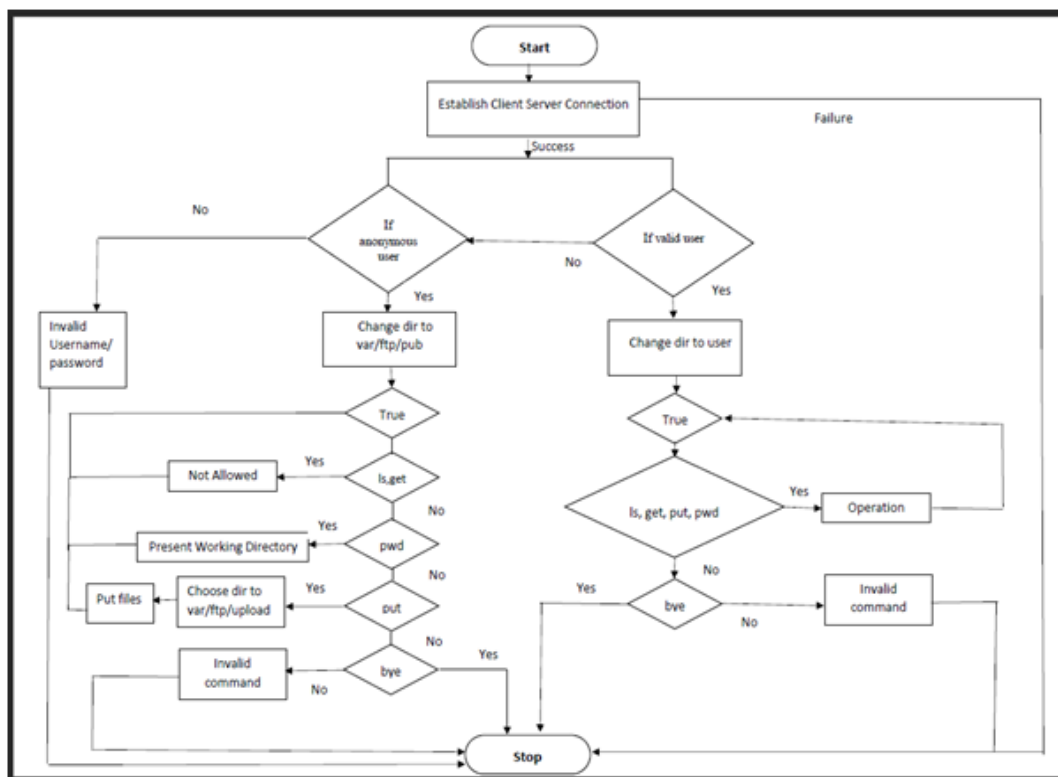
Name of the Module	BYE_Command
Handled by	
Description	This command works with both the anonymous and authenticated users and exits from the application with “GoodBye” message

2.2 Design Objectives:

1. Creating an easy to use FTP Application.
2. User authentication provides security to the user.
3. Only authorized users are provided all the functionalities in the application.
4. Anonymous users can login to the system but are restricted to only upload files to specific remote working directory in the server.
5. Quick and fast transfer of files between client and user.

3. DETAILED SYSTEM DESIGN

FLOWCHART DIAGRAM:



Flowchart of the FTP Application

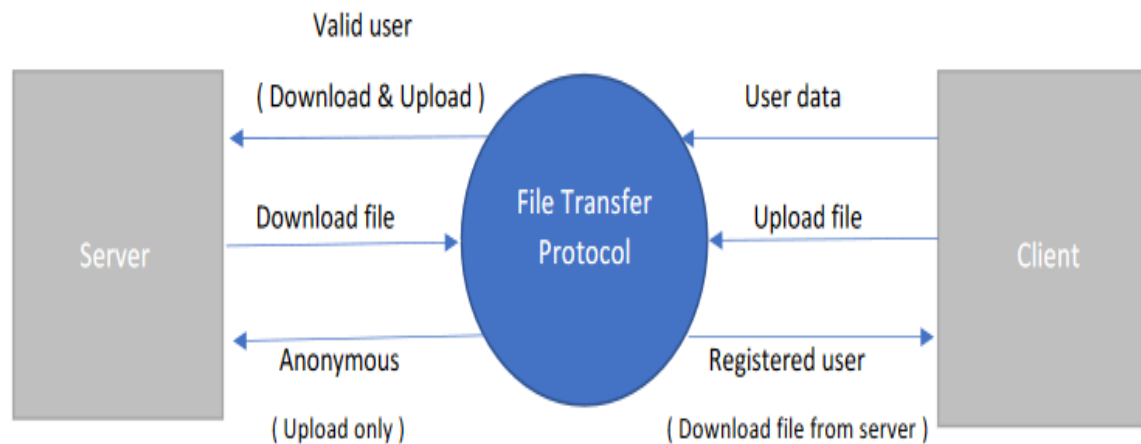
As shown in the design, the application starts with establishing connection between Client and Server. On Success, login credentials are asked by the user. On Failure, the application terminates.

If the user is valid, then the directory is changed to user, and he is allowed to browse the file list, download, upload the file and view the present working directory using the commands ls, get, put, pwd. Then the bye command is used to terminate the connection.

If the user is anonymous, then the directory is changed to var/ftp/pub, and he is allowed to just check his present working directory and to upload the files in var/ftp/upload directory. The anonymous user will not have access to browse or download the files. At Last, Bye command is used to terminate the connection and closes the application.

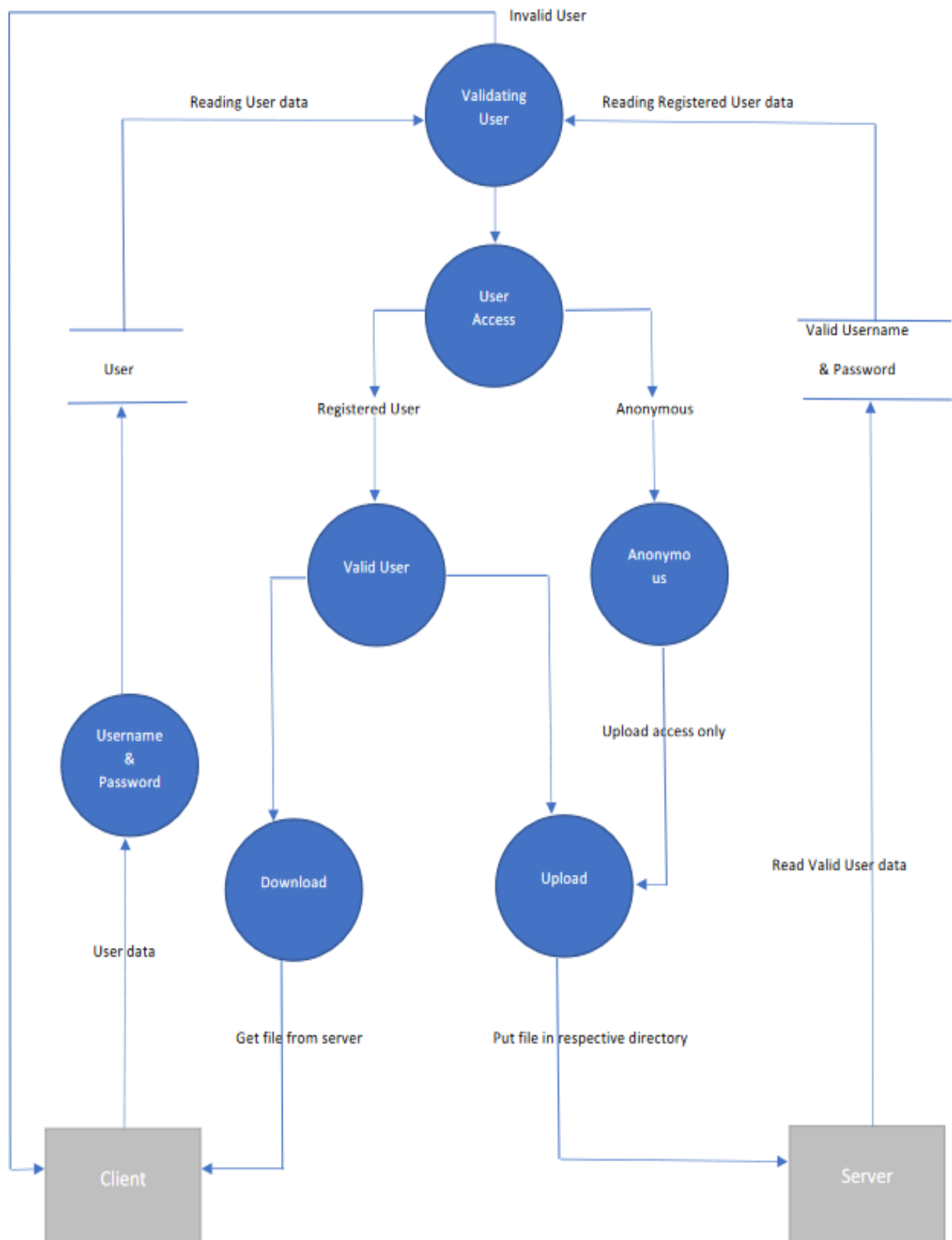
Data Flow Diagrams (DFD) : -

DFD LEVEL - 0



DFD level 0 of the FTP Application

DFD LEVEL – 1



DFD level 1 of the FTP Application

4. TOOLS REPORT:

4.1 Valgrind

- Server

```
==170266== Memcheck, a memory error detector
==170266== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==170266== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==170266== Command: ../bin/ftserve.exe 8021
==170266==
==170269==
==170269== HEAP SUMMARY:
==170269==   in use at exit: 0 bytes in 0 blocks
==170269==   total heap usage: 5 allocs, 5 frees, 9,256 bytes allocated
==170269==
==170269== All heap blocks were freed -- no leaks are possible
==170269==
==170269== For lists of detected and suppressed errors, rerun with: -s
==170269== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
^C==170266==
==170266== Process terminating with default action of signal 2 (SIGINT)
==170266==   at 0x49935D7: accept (accept.c:26)
==170266==   by 0x10A80F: socket_accept (common.c:62)
==170266==   by 0x109686: main (ftserve.c:26)
==170266==
==170266== HEAP SUMMARY:
==170266==   in use at exit: 0 bytes in 0 blocks
==170266==   total heap usage: 0 allocs, 0 frees, 0 bytes allocated
```

```
==170269== HEAP SUMMARY:
==170269==   in use at exit: 0 bytes in 0 blocks
==170269==   total heap usage: 5 allocs, 5 frees, 9,256 bytes allocated
==170269==
==170269== All heap blocks were freed -- no leaks are possible
==170269==
==170269== For lists of detected and suppressed errors, rerun with: -s
==170269== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
^C==170266==
==170266== Process terminating with default action of signal 2 (SIGINT)
==170266==   at 0x49935D7: accept (accept.c:26)
==170266==   by 0x10A80F: socket_accept (common.c:62)
==170266==   by 0x109686: main (ftserve.c:26)
==170266==
==170266== HEAP SUMMARY:
==170266==   in use at exit: 0 bytes in 0 blocks
==170266==   total heap usage: 0 allocs, 0 frees, 0 bytes allocated
==170266==
==170266== All heap blocks were freed -- no leaks are possible
==170266==
==170266== For lists of detected and suppressed errors, rerun with: -s
==170266== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

- Client

```

==170268== Memcheck, a memory error detector
==170268== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==170268== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==170268== Command: ../bin/ftclient.exe 127.0.0.1 8021
==170268==

==170268==
==170268== HEAP SUMMARY:
==170268==   in use at exit: 120 bytes in 1 blocks
==170268==   total heap usage: 6 allocs, 5 frees, 6,800 bytes allocated
==170268==
==170268== LEAK SUMMARY:
==170268==   definitely lost: 0 bytes in 0 blocks
==170268==   indirectly lost: 0 bytes in 0 blocks
==170268==   possibly lost: 0 bytes in 0 blocks
==170268==   still reachable: 120 bytes in 1 blocks
==170268==   suppressed: 0 bytes in 0 blocks
==170268== Rerun with --leak-check=full to see details of leaked memory
==170268==
==170268== For lists of detected and suppressed errors, rerun with: -s
==170268== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)

```

4.2 Strace

% time	seconds	usecs/call	calls	errors	syscall
0.00	0.000000	0	4		read
0.00	0.000000	0	9		write
0.00	0.000000	0	4		close
0.00	0.000000	0	8		mmap
0.00	0.000000	0	3		mprotect
0.00	0.000000	0	1		munmap
0.00	0.000000	0	3		brk
0.00	0.000000	0	8		ioctl
0.00	0.000000	0	4		pread64
0.00	0.000000	0	1	1	access
0.00	0.000000	0	1		socket
0.00	0.000000	0	1		connect
0.00	0.000000	0	3		sendto
0.00	0.000000	0	4		recvfrom
0.00	0.000000	0	1		execve
0.00	0.000000	0	2	1	arch_prctl
0.00	0.000000	0	1		set_tid_address
0.00	0.000000	0	3		openat
0.00	0.000000	0	5		newfstatat
0.00	0.000000	0	1		set_robust_list
0.00	0.000000	0	1		prlimit64
0.00	0.000000	0	1		getrandom
0.00	0.000000	0	1		rseq

0.00	0.000000	0	9	write
0.00	0.000000	0	4	close
0.00	0.000000	0	8	mmap
0.00	0.000000	0	3	mprotect
0.00	0.000000	0	1	munmap
0.00	0.000000	0	3	brk
0.00	0.000000	0	8	ioctl
0.00	0.000000	0	4	pread64
0.00	0.000000	0	1	1 access
0.00	0.000000	0	1	socket
0.00	0.000000	0	1	connect
0.00	0.000000	0	3	sendto
0.00	0.000000	0	4	recvfrom
0.00	0.000000	0	1	execve
0.00	0.000000	0	2	1 arch_prctl
0.00	0.000000	0	1	set_tid_address
0.00	0.000000	0	3	openat
0.00	0.000000	0	5	newfstatat
0.00	0.000000	0	1	set_robust_list
0.00	0.000000	0	1	prlimit64
0.00	0.000000	0	1	getrandom
0.00	0.000000	0	1	rseq

100.00	0.000000	0	70	2 total

% time	seconds	usecs/call	calls	errors	syscall

0.00	0.000000	0	1		read
0.00	0.000000	0	3		close
0.00	0.000000	0	8		mmap
0.00	0.000000	0	3		mprotect
0.00	0.000000	0	1		munmap
0.00	0.000000	0	1		brk
0.00	0.000000	0	4		pread64
0.00	0.000000	0	1	1	access
0.00	0.000000	0	1		socket
0.00	0.000000	0	2	1	accept
0.00	0.000000	0	1		bind
0.00	0.000000	0	1		listen
0.00	0.000000	0	1		setsockopt
0.00	0.000000	0	1		clone
0.00	0.000000	0	1		execve
0.00	0.000000	0	2	1	arch_prctl
0.00	0.000000	0	1		set_tid_address
0.00	0.000000	0	2		openat
0.00	0.000000	0	2		newfstatat
0.00	0.000000	0	1		set_robust_list
0.00	0.000000	0	1		prlimit64

0.00	0.000000	0	3	close
0.00	0.000000	0	8	mmap
0.00	0.000000	0	3	mprotect
0.00	0.000000	0	1	munmap
0.00	0.000000	0	1	brk
0.00	0.000000	0	4	pread64
0.00	0.000000	0	1	1 access
0.00	0.000000	0	1	socket
0.00	0.000000	0	2	1 accept
0.00	0.000000	0	1	bind
0.00	0.000000	0	1	listen
0.00	0.000000	0	1	setsockopt
0.00	0.000000	0	1	clone
0.00	0.000000	0	1	execve
0.00	0.000000	0	2	1 arch_prctl
0.00	0.000000	0	1	set_tid_address
0.00	0.000000	0	2	openat
0.00	0.000000	0	2	newfstatat
0.00	0.000000	0	1	set_robust_list
0.00	0.000000	0	1	prlimit64
0.00	0.000000	0	1	rseq

100.00	0.000000	0	40	3 total

4.3 gprof

- Server

```
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        10      0.00      0.00  send_response
0.00    0.00    0.00         5      0.00      0.00  recv_data
0.00    0.00    0.00         4      0.00      0.00  trimstr
0.00    0.00    0.00         3      0.00      0.00  ftserve_recv_cmd
0.00    0.00    0.00         2      0.00      0.00  ftserve_start_data_conn
0.00    0.00    0.00         2      0.00      0.00  socket_connect
0.00    0.00    0.00         1      0.00      0.00  ftserve_check_user
0.00    0.00    0.00         1      0.00      0.00  ftserve_list
0.00    0.00    0.00         1      0.00      0.00  ftserve_login
0.00    0.00    0.00         1      0.00      0.00  ftserve_process
0.00    0.00    0.00         1      0.00      0.00  ftserve_pwd
0.00    0.00    0.00         1      0.00      0.00  socket_accept
0.00    0.00    0.00         1      0.00      0.00  socket_create

%           the percentage of the total running time of the
```

```
0.00    0.00    0.00         1      0.00      0.00  socket_create

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

cumulative  a running sum of the number of seconds accounted
seconds     for by this function and those listed above it.

self        the number of seconds accounted for by this
seconds     function alone. This is the major sort for this
           listing.

calls       the number of times this function was invoked, if
           this function is profiled, else blank.

self        the average number of milliseconds spent in this
ms/call     function per call, if this function is profiled,
           else blank.

total       the average number of milliseconds spent in this
ms/call     function and its descendents per call, if this
           function is profiled, else blank.

name        the name of the function. This is the minor sort
```

```
for this listing. The index shows the location of
the function in the gprof listing. If the index is
in parenthesis it shows where it would appear in
the gprof listing if it were to be printed.

Copyright (C) 2012-2022 Free Software Foundation, Inc.

Copying and distribution of this file, with or without modification,
are permitted in any medium without royalty provided the copyright
notice and this notice are preserved.

Call graph (explanation follows)

granularity: each sample hit covers 4 byte(s) no time propagated

index % time   self  children  called    name
0.00    0.00    0.00    1/10      ftserve_login [9]
0.00    0.00    0.00    2/10      ftserve_list [8]
0.00    0.00    0.00    2/10      ftserve_pwd [11]
0.00    0.00    0.00    2/10      ftserve_process [10]
0.00    0.00    0.00    3/10      ftserve_recv_cmd [4]
```



```

[1] 0.0 0.00 0.00 10 send_response [1]
-----
      0.00 0.00 2/5 ftserve_login [9]
      0.00 0.00 3/5 ftserve_recv_cmd [4]
[2] 0.0 0.00 0.00 5  recv_data [2]
-----
      0.00 0.00 4/4 ftserve_check_user [7]
[3] 0.0 0.00 0.00 4  trimstr [3]
-----
      0.00 0.00 3/3 ftserve_process [10]
[4] 0.0 0.00 0.00 3  ftserve_recv_cmd [4]
      0.00 0.00 3/5 recv_data [2]
      0.00 0.00 3/10 send_response [1]
-----
      0.00 0.00 2/2 ftserve_process [10]
[5] 0.0 0.00 0.00 2  ftserve_start_data_conn [5]
      0.00 0.00 2/2 socket_connect [6]
-----
      0.00 0.00 2/2 ftserve_start_data_conn [5]
[6] 0.0 0.00 0.00 2  socket_connect [6]
-----
      0.00 0.00 1/1 ftserve_login [9]
[7] 0.0 0.00 0.00 1  ftserve_check_user [7]
      0.00 0.00 4/4 trimstr [3]

```

```

-----
      0.00 0.00 1/1 ftserve_process [10]
[8] 0.0 0.00 0.00 1  ftserve_list [8]
      0.00 0.00 2/10 send_response [1]
-----
      0.00 0.00 1/1 ftserve_process [10]
[9] 0.0 0.00 0.00 1  ftserve_login [9]
      0.00 0.00 2/5 recv_data [2]
      0.00 0.00 1/10 send_response [1]
      0.00 0.00 1/1 ftserve_check_user [7]
-----
      0.00 0.00 1/1 main [21]
[10] 0.0 0.00 0.00 1  ftserve_process [10]
      0.00 0.00 3/3 ftserve_recv_cmd [4]
      0.00 0.00 2/10 send_response [1]
      0.00 0.00 2/2 ftserve_start_data_conn [5]
      0.00 0.00 1/1 ftserve_login [9]
      0.00 0.00 1/1 ftserve_list [8]
      0.00 0.00 1/1 ftserve_pwd [11]
-----
      0.00 0.00 1/1 ftserve_process [10]
[11] 0.0 0.00 0.00 1  ftserve_pwd [11]
      0.00 0.00 2/10 send_response [1]
-----

```

Each entry in this table consists of several lines. The line with the index number at the left hand margin lists the current function.

The lines above it list the functions that called this function, and the lines below it list the functions this one called.

This line lists:

index A unique number given to each element of the table.
Index numbers are sorted numerically.
The index number is printed next to every function name so it is easier to look up where the function is in the table.

% time This is the percentage of the 'total' time that was spent in this function and its children. Note that due to different viewpoints, functions excluded by options, etc, these numbers will NOT add up to 100%.

self This is the total amount of time spent in this function.

children This is the total amount of time propagated into this function by its children.

called This is the number of times the function was called. If the function called itself recursively, the number only includes non-recursive calls, and is followed by

children This is the amount of time that was propagated from the function's children into this parent.

called This is the number of times this parent called the function '/' the total number of times the function was called. Recursive calls to the function are not included in the number after the '/'.

name This is the name of the parent. The parent's index number is printed after it. If the parent is a member of a cycle, the cycle number is printed between the name and the index number.

If the parents of the function cannot be determined, the word '<spontaneous>' is printed in the 'name' field, and all the other fields are blank.

For the function's children, the fields have the following meanings:

self This is the amount of time that was propagated directly from the child into the function.

children This is the amount of time that was propagated from the

child's children to the function.

called This is the number of times the function called this child '/' the total number of times the child was called. Recursive calls by the child are not listed in the number after the '/'.

name This is the name of the child. The child's index number is printed after it. If the child is a member of a cycle, the cycle number is printed between the name and the index number.

If there are any cycles (circles) in the call graph, there is an entry for the cycle-as-a-whole. This entry shows who called the cycle (as parents) and the members of the cycle (as children.) The '+' recursive calls entry shows the number of function calls that were internal to the cycle, and the calls entry for each member shows, for that member, how many times it was called from other members of the cycle.

Copyright (C) 2012-2022 Free Software Foundation, Inc.

Copying and distribution of this file, with or without modification,

If there are any cycles (circles) in the call graph, there is an entry for the cycle-as-a-whole. This entry shows who called the cycle (as parents) and the members of the cycle (as children.) The '+' recursive calls entry shows the number of function calls that were internal to the cycle, and the calls entry for each member shows, for that member, how many times it was called from other members of the cycle.

Copyright (C) 2012-2022 Free Software Foundation, Inc.

Copying and distribution of this file, with or without modification, are permitted in any medium without royalty provided the copyright notice and this notice are preserved.

Index by function name

[7] ftserve_check_user	[4] ftserve_rcv_cmd	[6] socket_connect
[8] ftserve_list	[5] ftserve_start_data_conn	[13] socket_create
[9] ftserve_login	[2] rcv_data	[3] trimstr
[10] ftserve_process	[1] send_response	
[11] ftserve_pwd	[12] socket_accept	

- Client

```

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        5      0.00      0.00  read_reply
0.00      0.00      0.00        4      0.00      0.00  read_input
0.00      0.00      0.00        3      0.00      0.00  ftclient_read_command
0.00      0.00      0.00        2      0.00      0.00  ftclient_open_conn
0.00      0.00      0.00        2      0.00      0.00  ftclient_send_cmd
0.00      0.00      0.00        2      0.00      0.00  print_reply
0.00      0.00      0.00        2      0.00      0.00  socket_accept
0.00      0.00      0.00        2      0.00      0.00  socket_create
0.00      0.00      0.00        1      0.00      0.00  ftclient_list
0.00      0.00      0.00        1      0.00      0.00  ftclient_pwd

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

cumulative  a running sum of the number of seconds accounted
seconds     for by this function and those listed above it.

```

```

cumulative  a running sum of the number of seconds accounted
seconds     for by this function and those listed above it.

self        the number of seconds accounted for by this
seconds     function alone. This is the major sort for this
            listing.

calls       the number of times this function was invoked, if
            this function is profiled, else blank.

self        the average number of milliseconds spent in this
ms/call     function per call, if this function is profiled,
            else blank.

total       the average number of milliseconds spent in this
ms/call     function and its descendents per call, if this
            function is profiled, else blank.

name        the name of the function. This is the minor sort
            for this listing. The index shows the location of
            the function in the gprof listing. If the index is
            in parenthesis it shows where it would appear in
            the gprof listing if it were to be printed.

```

```

Copyright (C) 2012-2022 Free Software Foundation, Inc.

Copying and distribution of this file, with or without modification,
are permitted in any medium without royalty provided the copyright
notice and this notice are preserved.

Call graph (explanation follows)

granularity: each sample hit covers 4 byte(s) no time propagated

index % time   self  children  called   name
[1]    0.0     0.00    0.00     5/5      main [18]
            0.00    0.00     5        read_reply [1]
-----
            0.00    0.00     1/4      main [18]
            0.00    0.00     3/4      ftclient_read_command [3]
[2]    0.0     0.00    0.00     4        read_input [2]
-----
            0.00    0.00     3/3      main [18]
[3]    0.0     0.00    0.00     3        ftclient_read_command [3]

```

	0.00	0.00	3/4	read_input [2]	
	0.00	0.00	2/2	main [18]	
[4]	0.0	0.00	0.00	2	ftclient_open_conn [4]
	0.00	0.00	2/2	socket_create [8]	
	0.00	0.00	2/2	socket_accept [7]	
	0.00	0.00	2/2	main [18]	
[5]	0.0	0.00	0.00	2	ftclient_send_cmd [5]
	0.00	0.00	2/2	main [18]	
[6]	0.0	0.00	0.00	2	print_reply [6]
	0.00	0.00	2/2	ftclient_open_conn [4]	
[7]	0.0	0.00	0.00	2	socket_accept [7]
	0.00	0.00	2/2	ftclient_open_conn [4]	
[8]	0.0	0.00	0.00	2	socket_create [8]
	0.00	0.00	1/1	main [18]	
[9]	0.0	0.00	0.00	1	ftclient_list [9]
	0.00	0.00	1/1	main [18]	
[10]	0.0	0.00	0.00	1	ftclient_pwd [10]

the total amount of time spent in each function and its children.

Each entry in this table consists of several lines. The line with the index number at the left hand margin lists the current function. The lines above it list the functions that called this function, and the lines below it list the functions this one called.

This line lists:

index	A unique number given to each element of the table. Index numbers are sorted numerically. The index number is printed next to every function name so it is easier to look up where the function is in the table.
% time	This is the percentage of the 'total' time that was spent in this function and its children. Note that due to different viewpoints, functions excluded by options, etc, these numbers will NOT add up to 100%.
self	This is the total amount of time spent in this function.
children	This is the total amount of time propagated into this function by its children.
called	This is the number of times the function was called. If the function called itself recursively, the number

called	This is the number of times the function was called. If the function called itself recursively, the number only includes non-recursive calls, and is followed by a '+' and the number of recursive calls.
name	The name of the current function. The index number is printed after it. If the function is a member of a cycle, the cycle number is printed between the function's name and the index number.

For the function's parents, the fields have the following meanings:

self	This is the amount of time that was propagated directly from the function into this parent.
children	This is the amount of time that was propagated from the function's children into this parent.
called	This is the number of times this parent called the function '/' the total number of times the function was called. Recursive calls to the function are not included in the number after the '/'.

```

    name      This is the name of the parent.  The parent's index
              number is printed after it.  If the parent is a
              member of a cycle, the cycle number is printed between
              the name and the index number.

If the parents of the function cannot be determined, the word
'<spontaneous>' is printed in the 'name' field, and all the other
fields are blank.

For the function's children, the fields have the following meanings:

    self      This is the amount of time that was propagated directly
              from the child into the function.

    children  This is the amount of time that was propagated from the
              child's children to the function.

    called    This is the number of times the function called
              this child '/' the total number of times the child
              was called.  Recursive calls by the child are not
              listed in the number after the '/'.

    name      This is the name of the child.  The child's index

```

If there are any cycles (circles) in the call graph, there is an entry for the cycle-as-a-whole. This entry shows who called the cycle (as parents) and the members of the cycle (as children.) The '+' recursive calls entry shows the number of function calls that were internal to the cycle, and the calls entry for each member shows, for that member, how many times it was called from other members of the cycle.

Copyright (C) 2012-2022 Free Software Foundation, Inc.

Copying and distribution of this file, with or without modification, are permitted in any medium without royalty provided the copyright notice and this notice are preserved.

Index by function name

[9] ftclient_list	[5] ftclient_send_cmd	[7] socket_accept
[4] ftclient_open_conn	[6] print_reply	[8] socket_create
[10] ftclient_pwd	[2] read_input	
[3] ftclient_read_command	[1] read_reply	

5. UNIT TESTING:

5.1 Cunit :

```

CUnit - A unit testing framework for C - Version 2.1-3
http://cunit.sourceforge.net/

Suite: Suite Check_user file
Test: test of check_user() in Sunny cases ...passed
Test: test of check_user() in Rainy cases ...passed

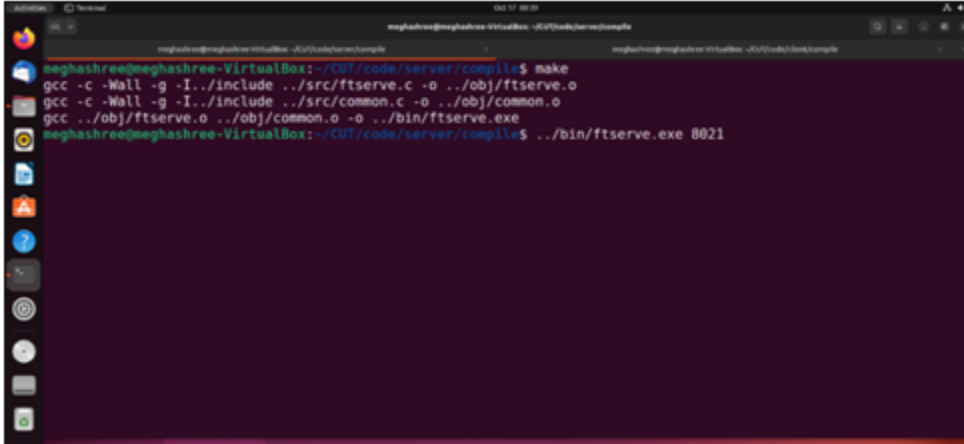
Run Summary:
  Type  Total  Ran  Passed  Failed  Inactive
  suites      1      1    n/a      0      0
  tests       2      2      2      0      0
  asserts     6      6      6      0    n/a

Elapsed time = 0.000 seconds

```

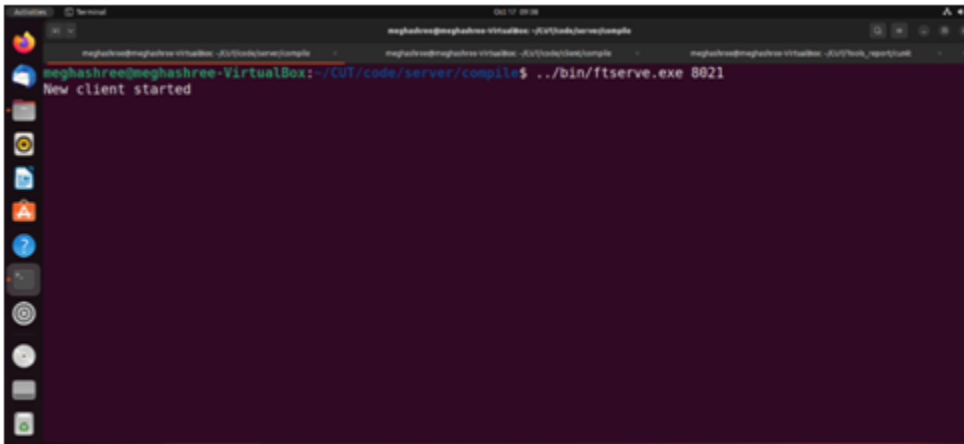
6. Integration Testing

6.1 Server starting at boot.

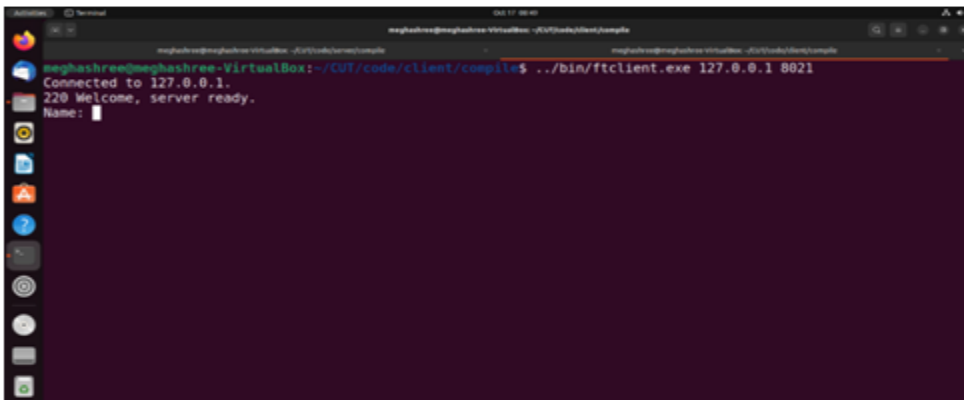


```
meghashree@meghashree-VirtualBox: ~/CUT/code/server/compile$ make
gcc -c -Wall -g -I../include ../src/ftserve.c -o ../obj/ftserve.o
gcc -c -Wall -g -I../include ../src/common.c -o ../obj/common.o
gcc ../obj/ftserve.o ../obj/common.o -o ../bin/ftserve.exe
meghashree@meghashree-VirtualBox: ~/CUT/code/server/compile$ ../bin/ftserve.exe 8021
```

6.2 Connection established, client has been started and listening to port 8021.



```
meghashree@meghashree-VirtualBox: ~/CUT/code/server/compile$ ../bin/ftserve.exe 8021
New client started
```



```
meghashree@meghashree-VirtualBox: ~/CUT/code/client/compile$ ../bin/ftclient.exe 127.0.0.1 8021
Connected to 127.0.0.1.
220 Welcome, server ready.
Name: 
```

6.3 Server side concurrency.

```
meghashree@meghashree-VirtualBox: ~/CUT/code/server/compile$ make
gcc -c -Wall -g -I../include ../src/ftserve.c -o ../obj/ftserve.o
gcc -c -Wall -g -I../include ../src/common.c -o ../obj/common.o
gcc ../obj/ftserve.o ../obj/common.o -o ../bin/ftserve.exe
meghashree@meghashree-VirtualBox: ~/CUT/code/server/compile$ ./bin/ftserve.exe 8021
New client entered
New client entered
```

```
meghashree@meghashree-VirtualBox: ~/CUT/code/client/compile$ make
gcc -c -Wall -g -I../include ../src/ftclient.c -o ../obj/ftclient.o
gcc -c -Wall -g -I../include ../src/common.c -o ../obj/common.o
gcc ../obj/ftclient.o ../obj/common.o -o ../bin/ftclient.exe
meghashree@meghashree-VirtualBox: ~/CUT/code/client/compile$ ./bin/ftclient.exe 127.0.0.1 8021
Connected to 127.0.0.1.
220 Welcome, server ready.
Name: anonymous
Password:
Anonymous Mode
successful login
ftclient> ls
ls command is not allowed for anonymous user!!
ftclient> bye
221 Goodbye!
meghashree@meghashree-VirtualBox: ~/CUT/code/client/compile$ ./bin/ftclient.exe 127.0.0.1 8021
Connected to 127.0.0.1.
220 Welcome, server ready.
Name: megha
Password:
successful login
ftclient> ls
pwd.txt
```

6.4 Client authentication and using ls, pwd, get, put and bye commands.

```
meghashree@meghashree-VirtualBox: ~/CUT/code/server/compile$ ./bin/ftserve.exe 8021
220 welcome, server ready.
Name: megh
Password:
Invalid username/password.
meghashree@meghashree-VirtualBox: ~/CUT/code/client/compile$ ./bin/ftclient.exe 127.0.0.1 8021
Connected to 127.0.0.1.
220 Welcome, server ready.
Name: amar
Password:
successful login
ftclient> ls
tmp.txt
ftclient> pwd
/home/meghashree/CUT/code/server/src/amar
ftclient> get a
550 Requested action not taken. File unavailable.
ftclient> get tmp.txt
226 Closing data connection. Requested file action successful.
ftclient> put amar
226 Closing data connection. Requested file action successful.
ftclient> put a
550 Requested action not taken. File unavailable.
ftclient> bye
221 Goodbye!
meghashree@meghashree-VirtualBox: ~/CUT/code/client/compile$
```


6.5 Authenticated users and working the commands ls, put, pwd, get and bye.

```
meghashree@meghashree-VirtualBox: ~/CUT/code/client/compile$ ./bin/ftclient.exe 127.0.0.1 8021
Invalid username/password.
meghashree@meghashree-VirtualBox:~/CUT/code/client/compile$ ./bin/ftclient.exe 127.0.0.1 8021
Connected to 127.0.0.1.
220 Welcome, server ready.
Name: kavya
Password:
successful login
ftclient> pwd
/home/meghashree/CUT/code/server/src/kavya
ftclient> ls
pwd.txt
tmp.txt
ftclient> get pwd.txt
226 Closing data connection. Requested file action successful.
ftclient> put megha
226 Closing data connection. Requested file action successful.
ftclient> ls
megha
pwd.txt
tmp.txt
ftclient> get abc
550 Requested action not taken. File unavailable.
ftclient> bye
221 Goodbye!
meghashree@meghashree-VirtualBox:~/CUT/code/client/compile$
```

```
meghashree@meghashree-VirtualBox:~/CUT/code/client/compile$ ./bin/ftclient.exe 127.0.0.1 8021
Connected to 127.0.0.1.
220 Welcome, server ready.
Name: megha
Password:
successful login
ftclient> ls
megha
pwd.txt
tmp.txt
ftclient> pwd
/home/meghashree/CUT/code/server/src/megha
ftclient> get megha
226 Closing data connection. Requested file action successful.
ftclient> put pwd.txt
226 Closing data connection. Requested file action successful.
ftclient> ls
megha
pwd.txt
tmp.txt
ftclient> bye
221 Goodbye!
meghashree@meghashree-VirtualBox:~/CUT/code/client/compile$
```

```
ftclient> ls
megha
pwd.txt
tmp.txt
ftclient> get abc
550 Requested action not taken. File unavailable.
ftclient> bye
221 Goodbye!
meghashree@meghashree-VirtualBox:~/CUT/code/client/compile$ ./bin/ftclient.exe 127.0.0.1 8021
Connected to 127.0.0.1.
220 Welcome, server ready.
Name: sneha
Password:
successful login
ftclient> pwd
/home/meghashree/CUT/code/server/src/sneha
ftclient> ls
pwd.txt
tmp.txt
ftclient> get pwd.txt
226 Closing data connection. Requested file action successful.
ftclient> put megha
226 Closing data connection. Requested file action successful.
ftclient> bye
221 Goodbye!
meghashree@meghashree-VirtualBox:~/CUT/code/client/compile$
```

```
meghashree@meghashree-VirtualBox: ~/CUT/code/client/compile
ftclient> get pwd.txt
226 Closing data connection. Requested file action successful.
ftclient> put megha
226 Closing data connection. Requested file action successful.
ftclient> bye
221 Goodbye!
meghashree@meghashree-VirtualBox:~/CUT/code/client/compile$ ./bin/ftclient.exe 127.0.0.1 8021
Connected to 127.0.0.1.
220 Welcome, server ready.
Name: vibha
Password:
successful login
ftclient> pwd
/home/meghashree/CUT/code/server/src/vibha
ftclient> ls
pwd.txt
tmp.txt
ftclient> get tmp.txt
226 Closing data connection. Requested file action successful.
ftclient> put megha
226 Closing data connection. Requested file action successful.
ftclient> get a
550 Requested action not taken. File unavailable.
ftclient> bye
221 Goodbye!
meghashree@meghashree-VirtualBox:~/CUT/code/client/compile$
```

6.6 Anonymous user (without password).

```
meghashree@meghashree-VirtualBox:~/CUT/code/client/compile$ make
gcc -c -Wall -g -I../include ./src/ftclient.c -o ../obj/ftclient.o
gcc -c -Wall -g -I../include ./src/common.c -o ../obj/common.o
gcc ../obj/ftclient.o ../obj/common.o -o ../bin/ftclient.exe
meghashree@meghashree-VirtualBox:~/CUT/code/client/compile$ ./bin/ftclient.exe 127.0.0.1 8021
Connected to 127.0.0.1.
220 Welcome, server ready.
Name: anonymous
Password:
Anonymous Mode
successful login
ftclient> ls
ls command is not allowed for anonymous user!!
ftclient> pwd
/home/meghashree/CUT/code/server/src/var/ftp/pub
ftclient> get pwd.txt
get command is not applicable in anonymous mode
ftclient> put pwd.txt
226 Closing data connection. Requested file action successful.
ftclient> pwd
/home/meghashree/CUT/code/server/src/var/ftp/upload
ftclient> bye
221 Goodbye!
meghashree@meghashree-VirtualBox:~/CUT/code/client/compile$
meghashree@meghashree-VirtualBox:~/CUT/code/client/compile$
```

6.7 Invalid username/password.

```
meghashree@meghashree-VirtualBox:~/CUT/code/client/compile$
meghashree@meghashree-VirtualBox:~/CUT/code/client/compile$
meghashree@meghashree-VirtualBox:~/CUT/code/client/compile$ ./bin/ftclient.exe 127.0.0.1 8021
Connected to 127.0.0.1.
220 Welcome, server ready.
Name: megh
Password:
Invalid username/password.
meghashree@meghashree-VirtualBox:~/CUT/code/client/compile$
```