Directory

1. System documentation

1.1 Routines

For client.c:

1. error(const char *msg): This function is responsible for displaying an error message and exiting the program.
2. main(int argc, char *argv[]): The main function of the client code. It takes command-line arguments 'argc' and 'argv' and performs the following tasks:
   - Checks if the required command-line arguments (host and port) are provided.
   - Creates a socket and establishes a connection to the server.
   - Reads user input from the terminal and sends it to the server.
   - Receives the response from the server and displays it.

For server.c:

1. error(const char *msg): This function is responsible for displaying an error message and exiting the program.
2. changeToUpper(char *filename): This function converts the content of a file to uppercase. It takes a filename as input, opens the file, reads its content, converts the characters to uppercase, and writes the result to an output file.
3. countCharacter(char targetChar, char *filename): This function counts the occurrences of a specific character in a file. It takes a target character and a filename as input, opens the file, reads its content, counts the occurrences of the target character, and writes the count to an output file.
4. main(int argc, char *argv[]): The main function of the server code. It takes command-line arguments 'argc' and 'argv' and performs the following tasks:
   - Checks if the required command-line argument (port) is provided.
   - Creates a socket, binds it to the specified port, and listens for client connections.
   - Accepts a client connection, reads the request from the client, and performs the corresponding action (convert to uppercase or count character).
   - Sends the response back to the client.

1.2 Implementation details.

For client.c:

(1) User input is read from the terminal using the fgets() function, and the resulting string is sent to the server using the write() function.
(2) The client code receives the response from the server using the read() function. The response is stored in a buffer, and then it is displayed on the terminal.

For server.c

(1) The server code reads the request from the client using the read() function. The request is stored in a buffer.
(2) The server code parses the request to determine the action to be performed (convert to uppercase or count character) and the associated parameters (filename and target character).

```
sscanf(buffer, "%*s < %s >", filename);
changeToUpper(filename);
```

Figure 1 parses the request

(3) Depending on the request, the server code calls the corresponding functions (changeToUpper() or countCharacter()) to perform the required operation on the file.

(4) Use while loop in changeToUpper() and countCharacter to finish multiple lines of input.

```
while (fgets(line, sizeof(line), file) != NULL)
{
    // Counts the occurrences of the target character in each line
    for (int i = 0; line[i] != '\0'; i++)
    {
        if (line[i] == targetChar)
        {
            count++;
        }
    }
}
```

Figure 2 multiple lines of input

(5) The server code sends the response back to the client using the 'write()' function. The response is read from the output file and sent in chunks until the end of the file is reached.

2. Test documentation

2.1 Test on one host(by ip or host name)

Input:

./server 54555

./client 127.0.0.1 54555

toUpper < intext.txt >

./server 54556

./client h9-virtual-machine 54556

./client count < u, intext.txt >

Output:

```
root@h9-virtual-machine:/home/h9/ICSI412/Assignment/Assignment5# ./server 54555
Run client by providing host and port
root@h9-virtual-machine:/home/h9/ICSI412/Assignment/Assignment5# ./server 54556
Run client by providing host and port
root@h9-virtual-machine:/home/h9/ICSI412/Assignment/Assignment5#
```

Figure 3 1host_server

```
root@h9-virtual-machine:/home/h9/ICSI412/Assignment/Assignment5# ./client 127.0.0.1 54555
Enter the request: toUpper < intext.txt >
Server response:
SOURCE CODE
REPRESENTS THE PART OF
PROCESS THAT
CONTAINS THE PROGRAMMING
LANGUAGE ITSELF. YOU MAY
USE A TEXT EDITOR TO
WRITE YOUR SOURCE CODE FILE.

root@h9-virtual-machine:/home/h9/ICSI412/Assignment/Assignment5# hostname
h9-virtual-machine
root@h9-virtual-machine:/home/h9/ICSI412/Assignment/Assignment5# ./client h9-virtual-machine 54556
Enter the request: count < u, intext.txt >
Server response:
6
```

Figure 4 1host_client

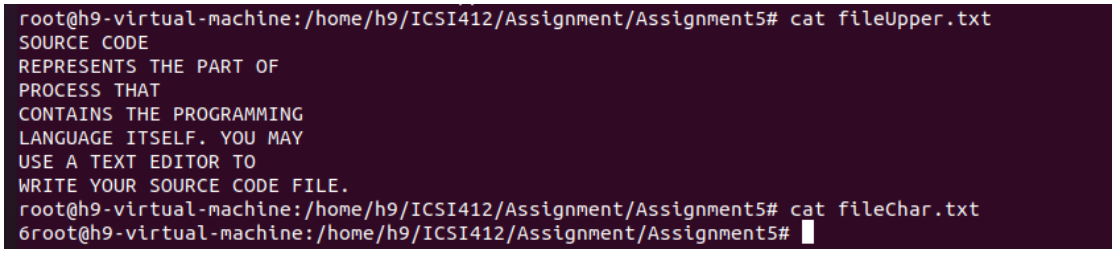

Figure 5 1host_output

2.2 Test on two hosts

Input:

./server 54557

./client 192.168.10.128 54557

./client count < u, intext.txt >

Output:

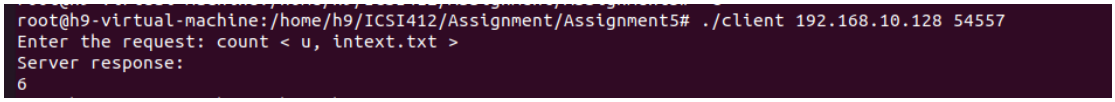

Figure 6 2hosts

3. User documentation

3.1 Source from

Assignment5 sample codes

3.2 How to run

1) In first terminal: Your-Prompt> ./server port-number
2) In second terminal: Your-Prompt> ./client server-IP-address port-number or Your-Prompt> ./client server-host-name port-number
3) In second terminal: input request as toUpper < file.txt > or count < char, file.txt >

(Notice the Spaces before and after the symbols)

3.3 Paraments

Server:

char request[256]; // The request

char filename[256]; // The target file name

char targetChar; // The target character

(The remaining variables are given by the sample code)