File Transfer Protocol Impremented in C

Adrian-Valentin Panaintescu[0000−0002−5853−4764]

Computer Science at the “Alexandru Ioan Cuza” University of Iasi

Abstract. The File Transfer Protocol (FTP) is a standard network protocol used for the transfer of computer files between a client and server on a computer network. The programs can be used on Unix-like operating systems.

Keywords: File Transfer · Computer Network · Unix

1 Introduction

This project is a server / client application that allows the transfer of computer files through the network. The server provides clients with a minimum number of commands that allow authentication, directory and file operation. Each user has managed permissions for each action they can take. User passwords are encrypted before being transmitted. There is a blocklist-based system to prevent malicious actions.

2 The technologies used

The technologies involved in this project are TCP. The motivation for using this communication protocol is that TCP provides reliable, ordered, and error-checked delivery of a stream of bytes (bytes) between applications running on hosts communicating via an IP network and ensure data reception. [35]

3 Application architecture

The application is based on communication on a tcp protocol, server and clients. All files are stored on the server. After starting and configuring the server waits for the connection requests to come.

When a connection request arrives and the connection is accepted, a child case process is created. It will take care of processing the commands received from the client.

Communication between client and server is done through two sockets, one for control and one for the data that is created for each request.

The client application at startup initializes the login process and waits for commands entered by the user on the keyboard. The entered command is initially processed by removing an int that speeds up the command id and its arguments. The command is sent with the argument. The server responded with a status code preceded by an action or ignoring that and returning to the stage of entering the command by the user.

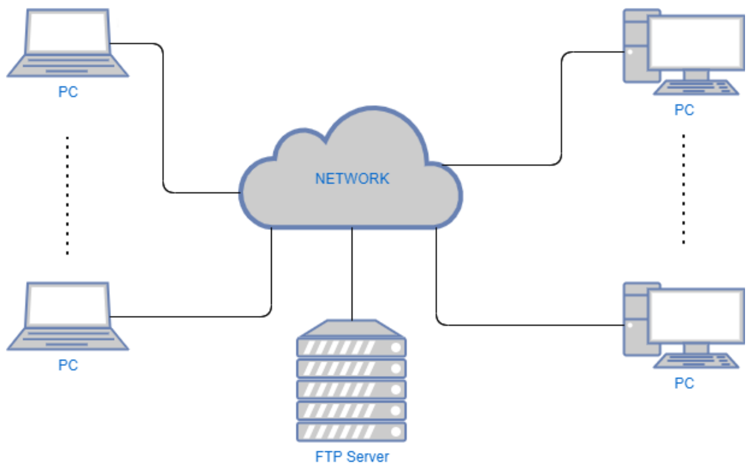

Fig. 1. Communication between server and clients

4 Implementation details

Communication is done through a dedicated socket for control. Orders are received and statuses are sent. The data exchange is performed on another socket, the data socket, which is created for each command. After processing orders it closes.

The main features of this project is file sharing. This logic is shown in the diagram above.

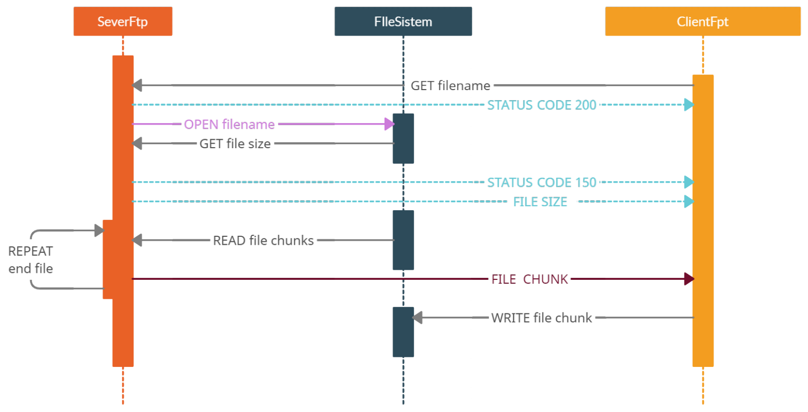

Fig. 2. File sharing comunication

```
$config \leftarrow readconfig()$
$socket_{listen} \leftarrow socketCreate(config["port"])$
while 1 do
    $socket_{control} \leftarrow socketAccept(socket_{listen})$
    if $fork()$ then
        $close(socket_{listen})$
        $sendResponse(code_{ok})$
        $login()$
        $getPermision()$
        while 1 do
            $r \leftarrow reciveCommands()$
            $r \leftarrow checkPermision(r)$
            if $r = code_{ok}$ then
                $socket_{data} \leftarrow socketConnect(socket_{control})$
                $executeCommand()$
                $close(socket_{data})$
            if $r = code_{exit}$ then
                $break$
            if $r = code_{unauthorized}$ then
                $contiune$
        $close(socket_{control})$
        $a \leftarrow b$
    $close(socket_{control})$
$close(socket_{listen})$
```

4.1 Utilitarian Functions

A number of utilitarian functions are involved in both applications. These functions are found in a common file, together with which these two applications will be compiled. These functions are meant to shorten the length of the written code and to avoid duplicating the code.

int socket_create(int port) returns an open descriptor socket on the port provided as a parameter or -1 in case of error.

int socket_accept(int sd) returns a new socked descriptor about after the connection is accepted.

int socket_connect(int port, char *host) connect to remote host at given port and returns socket descriptor on success, -1 on error.

int recv_data(int sockfd, char* buf, int bufsize) receive data on sockfd. Returns -1 on error, number of bytes received on success.

int recv_code(int sockfd) receive a response from sockfd. Returns -1 on error, return code on success.

int send_response(int sockfd, int rc) send resposne code on sockfd. Returns -1 on error, 0 on success.

void print_reply(int rc) print the message for response code.

int file_recive(int sock_data, int sock_control, char path, int int s_prog) receive a file on sock_data and save in path, comunicate with sock_control. Return 0 on success, -1 on error.

int file_send(int sock_data, int sock_control, char * path, int s_prog) send a file through sock_data. Return 0 on succes, -1 on error.

void strtrim(char *str)) trim whiteshpace and line ending characters from string.

void strlow(char *str)) lower each string characters.

void read_input(char * buffer, int size) read input from command line.

int readconfig(char * dest, char * path, char * key) copy key value on destination under the key in the file from path. Return bityes length copied.

void tree(char *basePath, const int root, FILE * file) print in a file a tree view of base path.

void cripto(char * mess, char * key, int encoding) encript mess with key. For each $character_i$ from is added to $caracter_i$ % key lenght if key is not 0 otherwise it decreases.

char * statkey(char * srt) return a key based the string recived as parameter.

int is_file(const char *path) return 1 if it is a file else 0.

int remove_directory(const char *path) remove directory recursively. Return -1 on error.

int progressbar(long int p,long int t) show progress bar

int filesize(char *path) return filesize.

5 Conclusions

The project illustrates the capability and advantages of a networked computer. I hope that the project will be enjoyed by a small group of people. Although it is insignificant compared to what software are launched on the market.

For the future the application can be improved and add new features as well as added optimizations. Some tests must be done to determine how efficient the server is and to know what computational resources we need depending on the average number of clients and requests received per day / minute. In the client we can add functionalities for productivity or a text dye depending on the file format.

References

1. Vivek Gite. Linux see directory tree structure using tree command. URL: https://www.cyberciti.biz/faq/linux-show-directory-structure-command-line/.
2. GNU. Bibliography management with bibtex. URL: https://www.gnu.org/software/libc/manual/html_node/Cryptographic-Functions.html.
3. GNU. Standard and generic ways to silence make. URL: https://www.gnu.org/software/automake/manual/html_node/Tricks-For-Silencing-Make.html.
4. GNU. Variadic macros. URL: https://gcc.gnu.org/onlinedocs/cpp/Variadic-Macros.html.
5. Jace H. Hall. C implementation of cryptographic algorithms. URL: https://www.ti.com/lit/an/slaa547b/slaa547b.pdf?ts=1609271172170.
6. includehelp. Working with hexadecimal values in c programming language. URL: https://www.includehelp.com/c/working-with-hexadecimal-values-in-c-programming-language.aspx.
7. Linux man page. getpeername(3). URL: https://linux.die.net/man/3/getpeername.
8. Linux manual page. recv(2). URL: https://man7.org/linux/man-pages/man2/recv.2.html.
9. Linux manual page. system(3). URL: https://man7.org/linux/man-pages/man3/system.3.html.
10. Prasoon Mishra. chdir() in c language with examples. URL: https://www.geeksforgeeks.org/chdir-in-c-language-with-examples/.
11. Overleaf. Bibliography management with bibtex. URL: https://www.overleaf.com/learn/latex/bibliography_management_with_bibtex.
12. Overleaf. Bibtex bibliography styles. URL: https://www.overleaf.com/learn/latex/bibtex_bibliography_styles.
13. Overleaf. Hyperlinks. URL: https://www.overleaf.com/learn/latex/hyperlinks.
14. Overleaf. Understanding underfull and overfull box warnings. URL: https://www.overleaf.com/learn/how-to/Understanding_underfull_and_overfull_box_warnings.
15. Overleaf. Using bibliographies. URL: https://www.overleaf.com/learn/how-to/Using_bibliographies_on_Overleaf.

16. Pankaj Prakash. C program to list all files in a directory recursively. URL: https://codeforwin.org/2018/03/c-program-to-list-all-files-in-a-directory-recursively.html.
17. Programiz. C file handling. URL: https://www.programiz.com/c-programming/c-file-input-output.
18. Programiz. C tolower(). URL: https://www.programiz.com/c-programming/library-function/ctype.h/tolower.
19. StackOverflow. Checking if a file is a directory or just a file. URL: https://stackoverflow.com/questions/4553012/checking-if-a-file-is-a-directory-or-just-a-file.
20. StackOverflow. Get file size in c. URL: https://stackoverflow.com/questions/238603/how-can-i-get-a-files-size-in-c.
21. StackOverflow. Removing a non empty directory programmatically in c or c++. URL: https://stackoverflow.com/questions/2256945/removing-a-non-empty-directory-programmatically-in-c-or-c.
22. ASCII Table. Extended ascii code. URL: https://theasciicode.com.ar/extended-ascii-code/box-drawings-single-vertical-line-character-ascii-code-179.html.
23. techonthenet.com. C language: strtol function (convert string to long integer). URL: https://www.techonthenet.com/c_language/standard_library_functions/stdlib_h/strtol.php.
24. trytoprogram.com. C program to encrypt and decrypt the string. URL: http://www.trytoprogram.com/c-examples/c-program-to-encrypt-and-decrypt-string/.
25. tutorialspoint. C library function - fflush(). URL: https://www.tutorialspoint.com/c_standard_library/c_function_fflush.htm.
26. tutorialspoint. C library function - fread(). URL: https://www.tutorialspoint.com/c_standard_library/c_function_fread.htm.
27. tutorialspoint. C library function - fseek(). URL: https://www.tutorialspoint.com/c_standard_library/c_function_fseek.htm.
28. tutorialspoint. C library function - rename(). URL: https://www.tutorialspoint.com/c_standard_library/c_function_rename.htm.
29. tutorialspoint. make - unix, linux command. URL: https://www.tutorialspoint.com/unix_commands/make.htm.
30. unicode.org. The unicode standard, version 13.0. URL: https://www.unicode.org/charts/PDF/U2500.pdf.
31. Jesus Vigo. 16 terminal commands every user should know. URL: https://www.techrepublic.com/article/16-terminal-commands-every-user-should-know/.
32. Wikipedia. List of ftp server return codes. URL: https://en.wikipedia.org/wiki/List_of_FTP_server_return_codes.
33. Wikypedia. Box-drawing character. URL: https://en.wikipedia.org/wiki/Box-drawing_character#Unicode.
34. Wikypedia. List of ftp commands. URL: https://en.wikipedia.org/wiki/List_of_FTP_commands.
35. Wikypedia. Transmission control protocol. URL: https://en.wikipedia.org/wiki/Transmission_Control_Protocol.