Experiment 9

IMPLEMENTATION OF FILE TRANSFER PROTOCOL (FTP)

AIM: To implement File Transfer Protocol

DESCRIPTION: FTP is a standard network protocol used to transfer (upload/download) files between a client and server on a computer network as shown in Figure 9.1. It is built on client-server model architecture and uses a separate control and data connections between the client and the server. FTP users may authenticate themselves with a clear-text sign-in protocol, normally in the form of a username and password, but can connect anonymously if the server is configured to allow it. For secure transmission that protects the username and password, and encrypts the content, FTP is often secured with SSL/TLS (FTPS). SSH File Transfer Protocol (SFTP) sometimes also used instead.

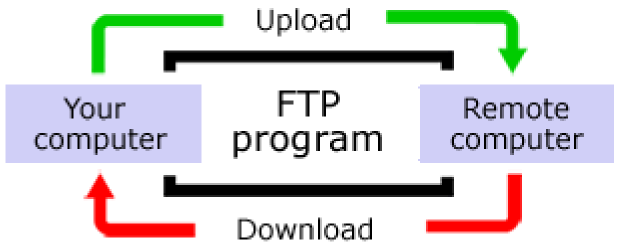

Figure 9.1: FTP program

It actually uses two channels (connections), one channel that uses port number **20**, used as a control channel and the other is used for data transmission that uses the port number **21** as shown in Figure 9.2

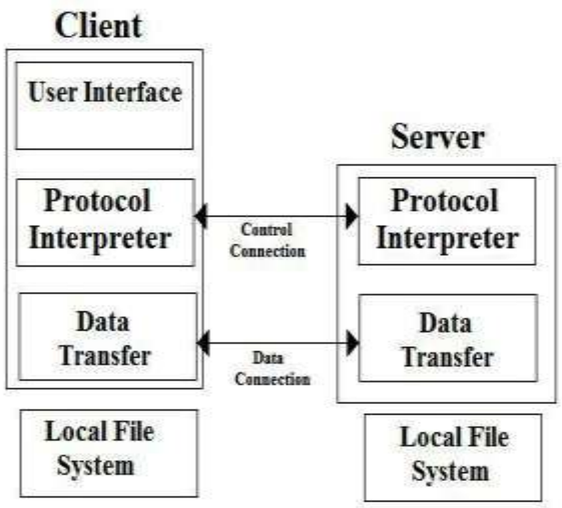

Figure 9.2: FTP control and data channels

FTP commands: now a days, GUI based FTP programs are available from where we can use the mouse. For CUI, FTP provides several commands. Some of them are:

Table 9.1: FTP commands

Command	Purpose
?	Request help or information about FTP commands
ascii	To set the mode of file transfer to ASCII
binary	To set the mode f file transfer to binary
bye	To exit the FTP environment(same as quit)
cd	To change directory on the remote machine
rm	to remove a file in the remote machine
get	To copy one file from the remote machine to the local machine
help	To request a list of all available FTP commands
lcd	To change directory on your local machine
ls	To list the names of files in the remote directory
mkdir	To make a new directory within the current remote directory
mget	To copy multiple files from the remote machine to the local machine
mput	To multiple files from the local machine to the remote machine
put	To copy one file from the local machine to the remote machine
pwd	To find ot the pathname of the current directory on the remote machine
rmdir	To remove (delete) a directory in the current remote directory
quit	To exit from FTP environment

Data representations modes: During data transfer over the network FTP uses the following modes:

1. **ASCII mode**: Used for text. Data is converted into 8-bit ASCII before transmission
2. **Image mode/Binary mode:** Sends the data as byte streams
3. **EBCDIC mode:** Used for plain text between hosts using EBCDIC character set.
4. **Local mode:** Allows two computers with identical setups to send data in a proprietary format without the need to convert it to ASCII

Data transfer modes

1. **Stream mode:** Data is sent as a continuous stream, relieving FTP doing any processing. All processing is left up to TCP. No EOF is needed unless the data is divided into records.
2. **Block mode**: FTP breaks the data into several blocks (block header, byte count and data filed) and then passes it on to TCP
3. **Compressed mode**: Data is compressed using a simple algorithm (usually run-length encoding)

Problem Description:

Write two separate programs named ***ftpclient.c*** and ***ftpserver.c***, to implement a simplified version of FTP. The server process should work in connection-oriented and concurrent-server mode. The sever process needs to be started in a server host and publish its host name and port number. A user from another machine can then issue a command like

$myftp <host-ID> <port-No> to download a file from or to upload a file o the server. After accepting the above *myftp* command, client process should respond with the prompt ***ftp>,*** waiting for user's ftp commands

Requirements:

1. Your executable FTP program name should be *myftp*
2. When you run the above program, it should prompt as ftp>
3. Need to implement the following ftp commands:

put <file_name>	to upload a file named file_name to the server
get *<file_name>*	to download a file named file_name from the server
ls	to list the file under the present directory of the server
cd	to change the present working directory of the server
pwd	to display the current working directory on of the server
!ls	to list the files in the current directory of the client
!cd	to change the present working directory of the client
!pwd	to display the current working directory of the client
quit	to quit from the ftp session and return to the system prompt

4. Other commands except the above should be considered as invalid ftp commands.
5. When put or get a non-existed file, your program should respond with "*File does not exist*"

Pseudocode for the ftp client:

```
sfd=socket(AF_INET, SOCK_STREAM,.....)
read the host_ID and the port number from the command line (argv[1], argv[2])
connect(sfd, &server_sock_address,...)
while(1)
{
        prompt as ftp>
        read a command from the user

        if the command is "put an existing file" to a remote machine
            1.   send the command to the server
            2.   open the file
            3.   read the contents from the file and write to the socket
            4.   close the file

        if the command is "get a file" from the remote machine
            1.   send the command to the server
            2.   read a line from the socket (server) and check whether the file is existing or not existing
            3.   if existed
                    a.   create a file in the current local directory
                    b.   read the contents from the server and write to the newly created local file
                    c.   read the contents from the file and write to the socket
                    d.   close the file
            4.  if non-existed
                    a.  display "filename: Does not exist"

        if the command is "cd", "ls", or "pwd"
            1.  send the command to the server
            2.  read the reply from the server (socket) to see if the command is successfully executed
            3.  read the response of the command and display correspondingly

        if the command is "!ls", or "!pwd"
            1.  call the system call locally

        if the command is "!cd"
            1.  call chdir(directory) locally

        if the command is "quit"
            1.  close the socket
            2.  break or exit

        otherwise show "An invalid command"
}
```

Pseudocode for the ftp concurrent server:

```
sfd=socket(AF_INET, SOCK_STREAM,.....)
bind the socket with the server address
listen(sfd,....)

while (1)
{
   nsfd=accept(sfd, ..................);     //take a client request
   pid=fork();
   if (pid==0)                              //child)
   while(1)
    {
        read a request from the client via nsfd

        if the command is "put a file"
            1.  create a file
            2.  read the file contents from the socket
            3.  write to the newly created file
            4.  close the file

        if the command is "get a file"
            1.  open the file
            2.  send "existed" to the client
                    a.  read the file contents
                    b.  write to the client via socket
                    c.  close the file
        if the command is "ls", or "pwd"
            1.  fp=popen(command, "r")
            2.  read the result from fp
            3.  if no result from fp then reply as "wrong command usage!" to the client, otherwise reply
                "successfully executed!" to the client
            4.  send the result to the client

        if the command is "cd directory"
            1.  call chdir(directory)
            2.  reply to the client if command is successfully executed

        if the command is "ls", or "quit"
            1.  close socket
            2.  exit
    }                       //end of inner while
}                           //end of outer while
```

Note:

- **Develop the code for the FTP server and client**
- **Test the programs and write your comments about the results**

CONCLUSIONS:

- **Write your conclusions**

References:

1. https://en.wikipedia.org/wiki/File_Transfer_Protocol