IMPLEMENTATION OF FTP PROTOCOL USING SOCKET PROGRAMMING IN PYTHON

Report submitted to the SASTRA Deemed to be University as the requirement for the course

CSE302 COMPUTER NETWORKS

Submitted by

LEKIREDDY SAI BHARATH SIMHA REDDY

Reg.No:122015053,B-Tech,Information Technology

February 2021

SCHOOL OF COMPUTING

THANJAVUR, TAMIL NADU, INDIA – 613 401

SCHOOL OF COMPUTING

THANJAVUR – 613 401

Bonafide Certificate

This is to certify that the report titled **"Implementation of FTP Protocol using Socket Programming in Python** " submitted as a requirement for the course, **CSE302: COMPUTER NETWORKS** for B.Tech. is a bonafide record of the work done by **Shri.Lekireddy Sai Bharath Simha Reddy Reg.No:122015053,B-Tech,Information Technology** during the academic year 2020-21, in the School of Computing.

Project Based Work *Viva voce* held on 18/02/21

Examiner 1 **Examiner 2**

List of Figures

Abbreviations

FTP	File Transfer Protocol
TCP	Transmission Control Protocol
IP	Internel Protocol
P2P	Peer to Peer
GUI	Graphical User Interface

KeyWords

Tkinter	Tkinter is the inbuilt python module that is used to create GUI applications
Canvas	The Canvas is a rectangular area intended for drawing pictures or other complex layouts.
Os	The OS module in python provides functions for interacting with the operating system.
Socket	A network socket is a software structure within a network node of a computer network that serves as an endpoint for sending and receiving data across the network.
Shutil	The shutil in Python is a module that offers several functions to deal with operations on files and their collections.

ABSTRACT

Many day to day works require files to be shared or transfered across systems.There has been a requirement for a protocol to transfer files effectively and efficiently.In order to achieve easy and fast transfer of files between systems FTP protocol used.

The File Transfer Protocol is a standard network protocol used for the transfer of computer files from a server to a client on a computer network. FTP is built on a client-server model architecture using separate control and data connections between the client and the server.

This project implements FTP protocol by using Socket Programming in Python Language. Socket Programming uses sockets and manipulates them to create a conection between software. Two different codes are written in python namely server.py and client.py.Server file is executed in Server system and client file is executed in Client system.A connecion will be established between Client and Server.Through this project, files can be

(i) downloaded
(ii) uploaded
(iii)viewed
(iv)can access properties

of files residing in Server system from Client side.

Table of Contents

INTRODUCTION

FTP Protocol

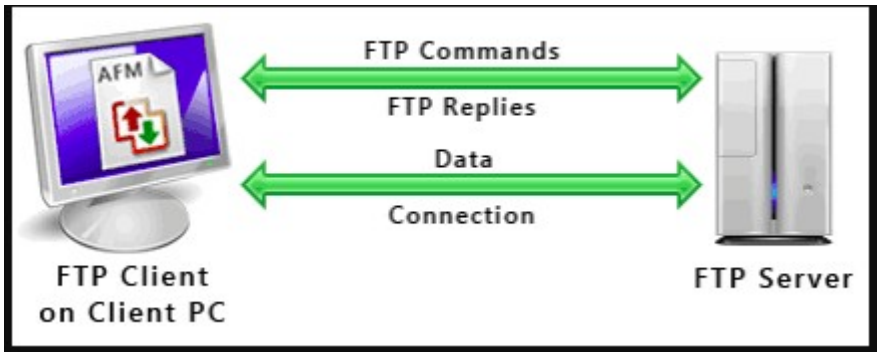
File Transfer Protocol is a standard network protocol used to transfer computer files between a client and a server across a computer network.FTP transfers files between Computers on TCP/IP network.

The first FTP client applications were command-line programs developed before operating systems had graphical user interfaces, and are still shipped with most Windows, Unix, and Linux operating systems. Many FTP clients and automation utilities have since been developed for desktops, servers, mobile devices, and hardware, and FTP has been incorporated into productivity applications, such as HTML editors.

FTP is built on a client-server model architecture using separate control and data connections between the client and the server.

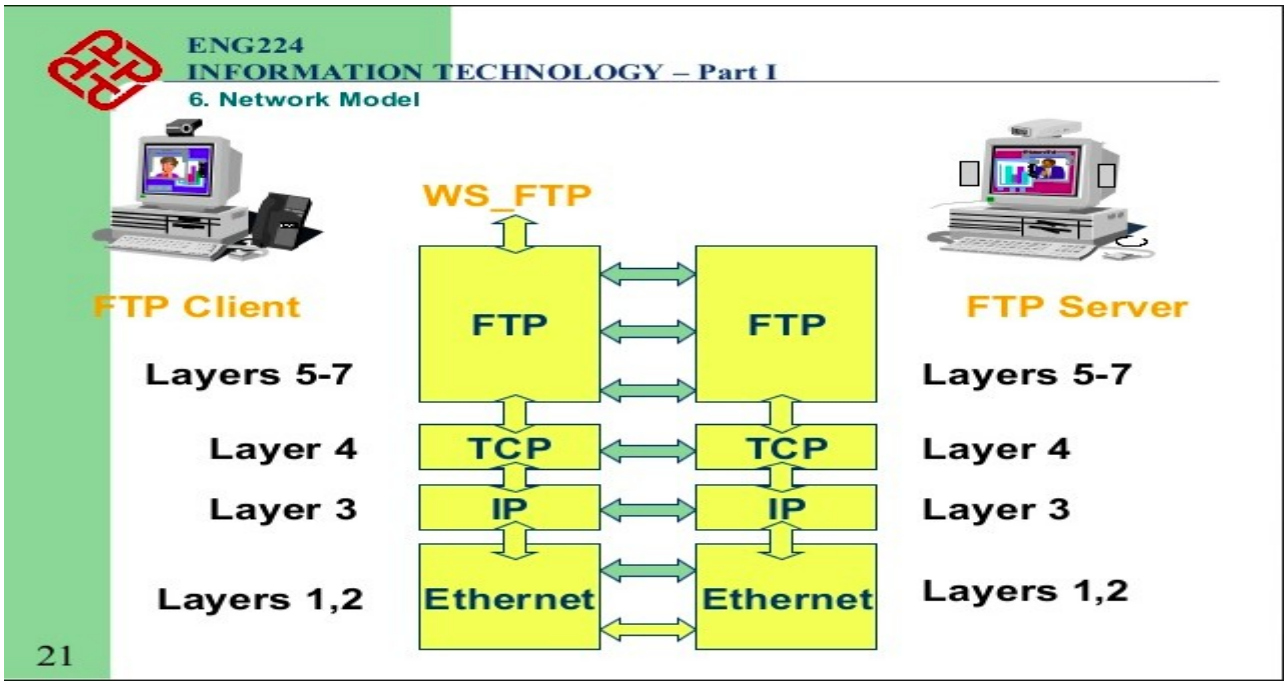
The objectives of FTP are
1) to promote sharing of files
2) to encourage indirect or implicit use of remote computers
3) to shield a user from variations in file storage systems among hosts
4) to transfer data reliably and efficiently. FTP, though usable directly by a user at a terminal, is designed mainly for use by programs.

FTP protocol uses TCP protocol for client server communication.
When an FTP client requests to connect to an FTP server, a TCP connection is being established using the application layer within TCP.
FTP uses and relies on TCP to ensure all the packets of data are sent correctly and to the proper destination.



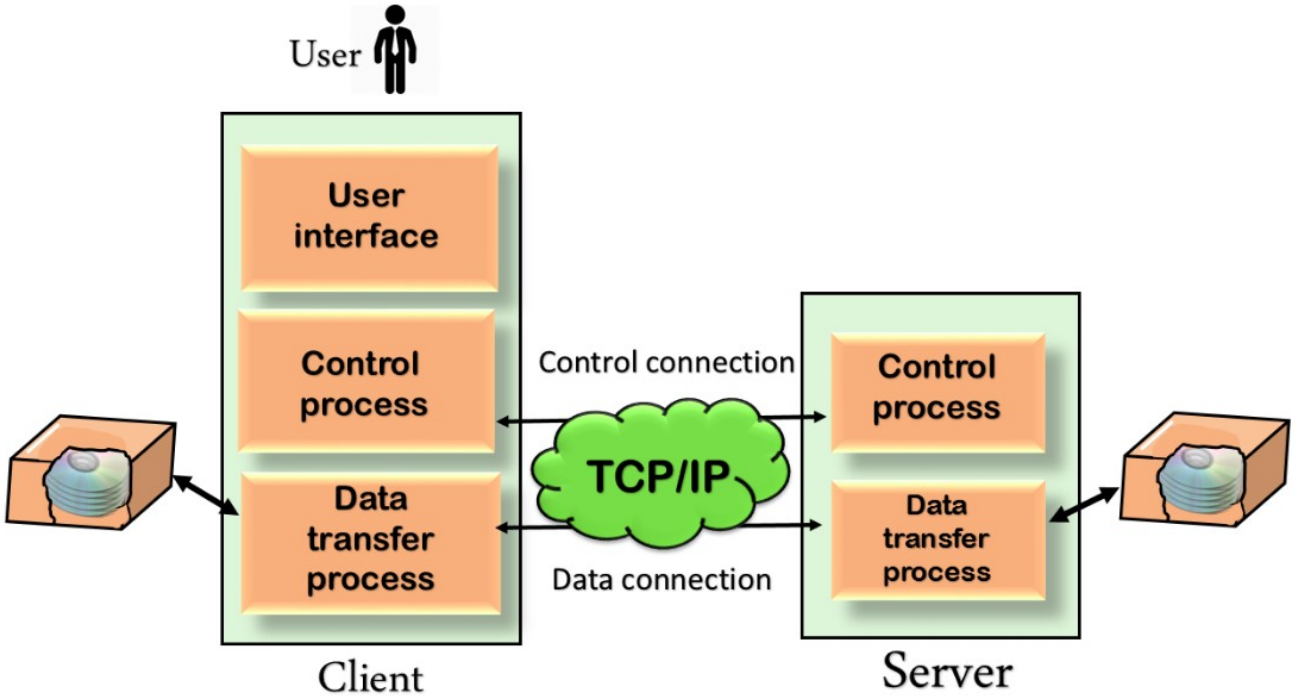
fig(i) Illustration of working of FTP protocol

Figure(i) depicts the exact working of FTP protocol.First a connection is made between client and server and then commands will be recieved to server from client And server responds to the commands.



fig(ii) Representation of FTP application into network Layers

FTP application is divided into layes according to fig(ii)
i.e,Ethernet acts as Physical Layer and Link Layer
IP acts at Network Layer
TCP acts as Transport Layer
FTP acts at Application Layer.



fig(iii)

Figure (iii) represents exact blueprint of this project.
This project has a GUI at the client side.With the help of GUI our project Communicates with server,transfer files and upload files.

Socket Programming

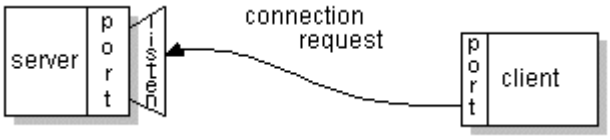
Socket programming involves using a list of commands to connect a socket from one computer to another.For example, for an instant messenger program to work, it must connect to a second computer. To make this connection, a socket is employed. By forging the connection, the two computers are now able to link together and speak to one another.

Peer-to-peer (P2P) programs are special cases when it comes to socket programming. Most programs act either as a client or a server. A P2P program acts as both, which is why users are able to download files from one person while files are downloaded from the user at the same time.

How does a Socket Work

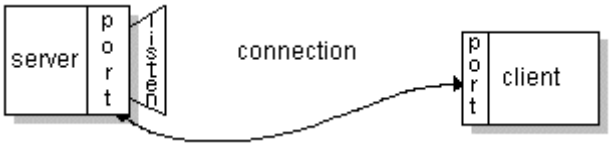
Normally, a server runs on a specific computer and has a socket that is bound to a specific port number. The server just waits, listening to the socket for a client to make a connection request.

On the client-side: The client knows the hostname of the machine on which the server is running and the port number on which the server is listening. To make a connection request, the client tries to rendezvous with the server on the server's machine and port. The client also needs to identify itself to the server so it binds to a local port number that it will use during this connection. This is usually assigned by the system.



fig(iv)

If everything goes well, the server accepts the connection. Upon acceptance, the server gets a new socket bound to the same local port and also has its remote endpoint set to the address and port of the client. It needs a new socket so that it can continue to listen to the original socket for connection requests while tending to the needs of the connected client.



fig(v)

On the client side, if the connection is accepted, a socket is successfully created and the client can use the socket to communicate with the server.

The client and server can now communicate by writing to or reading from their sockets.

An endpoint is a combination of an IP address and a port number. Every TCP connection can be uniquely identified by its two endpoints. That way you can have multiple connections between your host and the server.

Data Transfer

There are many types of files which are needed to transfer.FTP protocol has Different ways of transfering files.But in this project we only deal with single way of tranferring files.

So in this project for transfering data we initially convert the data into bytes And then send it through the socket connection.

After the data is reached to the other end of the connection it is converted back to its original form.

We follow this process through out the project.

Project Explanation:

Creation of Sockets and making conection

To work with sockets we need to import socket module from python.

Server is assigned a port number and it is bound to it

```
s = socket.socket()
try:
    s.bind(("192.168.43.152",9999))
except socket.error as e:
    print(e)
```

s.bind() is written under try-except block to ensure program does not stop if error prevails.

Similarly in client side set_connection is defined to make conections

```
def set_connection():
    print(Addr_etr.get())

    adr = (Addr_etr.get())
    pn = (port_etr.get())
    print(type(pn))
    s.connect((adr, int(pn)))
    print("Connected to server")
```

Multi Threading on Server Side

In this project Server need to make multiple connections simultaneously from different clients.

To address this issue we have to implement multi-threading on server side.
In order to achieve this first we hav to import _thread module from python.

To address this need we need to a function to service the clients needs.

So lets define do_service() function to service clients needs.

But there will be multiple clients at a time.
So we would write a while loop to accept the connections from the clients.

After new connection establishes we create a new thread for that connection and Pass that connection to do_service() function to service that connection needs.

```
while True:
    try:
        c, address = s.accept()
    except Exception as e:
        print(e)
    print('Connected with ', address)
    start_new_thread(do_service,(c,))
    ThreadCount+=1
    print('Thread Number:'+str(ThreadCount))
s.close()
```

start_new_thread() method is used to create a new thread.

Client GUI

In this project Client communicates with server through GUI.
Python tkinter module has been used for GUI programming.

As usual the necessary widgets such as IP_address,PortNo,Connect Button have been added to a frame and placed accordingly in the main window.

Each file is regarded as a menuButton in the client GUI.

Each file(MenuButton) has six options

1)Rename
2)Size
3)Date Info
4)Download
5)Type
6)Delete

All options are same for every directory except there is Open option instead of Download option.

There is also an upload Entry,Back button,Get Files button.

All these widgets have corresponding methods.
And each of this methods are serviced at server side by do_service method.

do_service method:

Every widget in the client side sends message at the beginning of the location
This method recieves a request from client socket.
According to the request it recieves it services the client by invoking respecrive function.

```
def do_service(c):
    while True:
        request = c.recv(1024).decode()

        if request[:5] == 'locn:':
            print(request[5:],'locn')
            if '$' not in request[5:]:
                send_file_names(c, request[5:])
        if request[:5] == 'down:':
            print(request[5:],'down')
            send_file(c,request[5:])
        if request[:4] == 'del:':
            print(request[4:],'del')
            del_file(c,request[4:])
        if request[:5] == 'deld:':
            print(request[5:],'deld:')
            del_dir(c,request[5:])
        if request[:5]=='upld:':
            print(request[5:],'upld:')
            recv_fl(c,request[5:])
    c.close()
```

For example
if get_files button is clicked then
Client Side
(i). client forms a message ‘locn’ and
(ii). adds it to the front of the location from while files are needed and
(iii). Sends it to the server side
(iv)Recieves the file names and add them to canvas as menuButtons by appending them
All necessary widgets as commands.

Server Side

(i)do_service() method recieves the message.
(ii). Extracts the front part of message
(iii). Invokes the send_file_names function by passing the connection and location as arguments.
(iv). send_file_name function sends file names and directory names which are under that directory.

Adding a Scroll Bar

Adding a scrollbar is not that simple task in tkinter.It can be done with the help of canvas.For that we need to follow a number of steps.
We can achieve this by following this six steps.

```
frameh = Frame(root)
frameh.pack()
```

i) *create a main frame*

```
main_frame = Frame(root)
main_frame.pack(fill=BOTH, expand=1)
```

ii) *create a canvas*

```
my_canvas = Canvas(main_frame)
my_canvas.pack(side=LEFT, fill=BOTH, expand=1)
```

iii) Add a scroll bar to canvas

```
scroll_bar = Scrollbar(main_frame, orient=VERTICAL, command=my_canvas.yview)
scroll_bar.pack(side=RIGHT, fill=Y)
```

iv) configure canvas

```
my_canvas.configure(yscrollcommand=scroll_bar.set)
my_canvas.bind('<Configure>', lambda e:
my_canvas.configure(scrollregion=my_canvas.bbox("all")))
```

v) create another frame in canvas

```
second_frame = Frame(my_canvas)
```

vi) *add the new frame to a window in the canvas*

```
my_canvas.create_window((0, 0), window=second_frame, anchor='nw')
```

In order to achieve scrolling we need to add every menuButton(files and directories) To second_frame which we created inside canvas.

get_file method():

We came to know about get_file method a little from do_service example in previous page.

Colour:
This method assigns different colour to files and directories.
Ex: All files will be in blue colour.
All directories will be in green colour.

Every message which is being send through sockets should be of type bytes.

Each command uses lambda function for passing arguments to the respective methods

For ex:

```
menubutton.menu.add_command(label="Size ", command=lambda i=i: size(locn[5:], text1[i]))
```

Here locn[5:] gives the path of the file and tex1[i] gives the file name or directory name.

Note: text1[] is a list which stores all the file names and dirctory names.

Rename command:

Rename command uses os module in python to rename a file.
After calling re_name() method it immediately geneates an alternate window
Which asks you to enter the new name of the file or directory.

It then takes this new_name and renames previous file or directory using os.rename('oldname','newNAME')

Size command:

Size command gives size of a file by using os.path.getsize(file_path).
For directory it uses os.walk() method to loop through all files and adds cumulative Size of file.

The resultant file size or directory size will be displayed in a alternative window.

Date command:

Date command uses time module to view
Last modified time - time.ctime(os.path.getmtime(file_path))
Last accessed time - time.ctime(os.path.getatime(file_path))
Created time – time.ctime(os.path.getctime(file_path))

It dispalys this info in alternate window

Type command:

Type command uses a dictionary which has keys as extensions and values as Type.
This command finds out the file extension in location path which is given to it as argument.

Delete Command:

Delete command sends a message (str('del:'+file_location)) to server side.
This message is accepted by do_service() method and the file at the given location is deleted by using os.remove(file_location)

Delete dir command:

Delete dir command sends a warning popup for confirmation.
If confirmed then it will be deleted by using shutil.rmtree(file_location) method

Open command:

Open command opens a directory by adding that directory name to the location entry widget and by calling get_files() method.

Back Button:

Back button calls Back commad which inturn removes last directory from the location Entry widget and by calling get_files() method.

Upload Button:

This method uploads a file which under uploadEntry location in the client side to LocationEntry location in the server side.
Upload Button calls up_load() method :
If given file is a file then up_load() method calls up_load_a() with k=1 as argument
If given file is a directory then up_load() method loops through files in directory and calls up_load_a() with k=2 as argument.

up_load_a method:

1)up_load_a method sends (location in locationEntry+file location under uploadEntry+str(k))
2)opens(file_in_the_clientside)
3)reads 1024 bytes at once from file and sends it server side(which is being recieved By recv())
4)calls get_files() method.

recv_fl() (server side):

```
def recv_fl(c,locn):

    k=int(locn[-1])
    if k>1:
        l=locn.split('/')
        ln=len(l)
        st=''
        for i in range(ln-1):
            st+=l[i]+'/'
            if os.path.isdir(st):
                pass
            else:
                os.mkdir(st)
    sz=0
    fil = open(locn[:-1], 'wb')
    while True:
        info=c.recv(1024)
        print(info,len(info))
        if len(info)<1024:
            print(info,len(info))
            fil.write(info)
            print(info)
            fil.close()
            print('quit')
            break

        sz+=len(info)
        print(info)
        fil.write(info)
    print(locn,'recieved')
    c.sendall('recv'.encode())
```

recv_fl uses ‘k’ value sent by up_load_a to determine if it is file or not.
If it is not a file then this method checks whether the given directory exists in the server or not.
If directory doesnot exists then it will create an empty directory and trasfers all files which were under that directory.

For ex:
Client side folder locn=’/home/bharath/Downloads’
Upload location(Server side)=’home/bharath’

If Downloads direcory does not exist in Server side the it creates Downloads folder And appends every file which is under that folder(through looping and uploading every file In the Downloads directory in client side)

Download method and send_file method:

1)Download method in client side sends message along with location to server side.
2)Empty file will be created in ‘wb’ mode
3)info from server side will be recieved and append it it to opened file.
4)close file

1)It is then sent to send_file method at server side
2)file handler opens file to be downloaded
3)info from fil will be read and sent to client side
4)closes file

These are the major methods in the source code of this project.

We also use try and except blocks wherever it is necessary.

```
def download(lokn,fl):
    locn='down:'+f'{lokn}/{fl}'
    s.sendall(locn.encode())
    fil=open(f'{fl}','wb')
    sz=0
    while True:
        info=s.recv(1024)
        print(info,len(info))
        if len(info)<1024:
            print(info,len(info))
            fil.write(info)
            fil.close()
            print('quit')
            break

        sz+=len(info)
        fil.write(info)

    print(sz,'characters copied')
```

```
def send_file(c,file):
    try:
        fl=open(file,'rb')
        size=0
        while True:
            info=fl.read(1024)
            if len(info)<1:
                #c.send(b'end')
                fl.close()
                print('quit')
                break
            size+=len(info)

            c.send(info)
    except Exception as e:
        print(e)
```

MERITS of Work:

i)We can transfer files easily and effectively.

ii)Can view properties of files such as file type,size,date information etc Without directly operating the server system.

iii)We can upload and download multiple files simultaneously.

iv)Many clients can access server at a time simultaneously.

v)It is connection oriented protocol and uses robust control commands.

vi)It sends data over separate TCP connection from the control commands. This enables fast data transfer

vii)It is simple in implementation and use.

viii)It is universal application due to its standardization. Hence it is widely used.

DeMerits of Work:

i)Security is less as it doesn't have any authentication.

ii)Lack of security as there is no encription.

iii) Multiple TCP/IP connections are used. Firewall hinders use of such connections.

iv)It is hard to filter active mode FTP traffic on client side when firewall is in place.

v)It has high latency due to its connection oriented nature.

vi)Not effective way of communicating errors.

Source Code

Client Code

```
import os
import subprocess
import sys
import socket
import time
from tkinter import *
from tkinter import ttk

s = socket.socket()
root = Tk()
root.title('Scroll test')
root.geometry("1410x600")


def set_connection():
    print(Addr_etr.get())

    adr = (Addr_etr.get())
    pn = (port_etr.get())
    print(type(pn))
    try:
        s.connect((adr, int(pn)))
    except Exception as e:
        print(e)
    # c.connect(('localhost', 9999))
    # name=input('Enter your name')
    # c.send(name.encode())
    # print(c.recv(1024).decode())
    print("Connected to server")


def re_rename(rt, lokn, fl, etr):
    try:
        os.rename(f'{lokn}/{fl}', f'{lokn}/{etr}')
        rt.destroy()
        get_files()
    pass
    except Exception as e:
        print(e)
```

```
def re_name(lokn, fl):
    rt = Tk()
    rt.geometry("400x200")
    lbl = Label(rt, text='Current Name: ')
    lbl.grid(row=0, column=0, sticky=E, padx=5, pady=5)
    lbl1 = Label(rt, text=fl)
    lbl1.grid(row=0, column=1, sticky=W, padx=5, pady=5)
    lbl2 = Label(rt, text="Enter New Name: ")
    lbl2.grid(row=1, column=0, sticky=E, padx=5, pady=5)
    etr = Entry(rt)
    etr.grid(row=1, column=1, sticky=W, padx=5, pady=5)
    btn = Button(rt, text='Rename', command=lambda rt=rt, lokn=lokn, fl=fl:
re_rename(rt, lokn, fl, etr.get()))
    btn.grid()

    rt.mainloop()


# = 0
def size(locn, name):
    sz = 0
    if os.path.isdir(f'{locn}/{name}'):
        for path, dirs, files in os.walk(f'{locn}/{name}'):
            for f in files:
                fp = os.path.join(path, f)
                sz += os.path.getsize(fp)
    else:
        sz = os.path.getsize(f'{locn}/{name}')
    print(sz)
    if sz > 1024 * 1024 * 1024:
        sz = str(round(sz / (1024 * 1024 * 1024), 2)) + 'GB'
    elif sz > 1024 * 1024:
        sz = str(round(sz / (1024 * 1024), 2)) + 'MB'
    elif sz > 1024:
        sz = str(round((sz / 1024), 2)) + 'KB'
    else:
        sz = str(sz) + 'B'
    print(sz)
    rt = Tk()
    rt.geometry('300x200')
    lbl = Label(rt, text=f'Size of {name}: ')
    lbl.grid(row=0, column=0, sticky=E, padx=5, pady=5)
    lbl1 = Label(rt, text=f'{sz}')
    lbl1.grid(row=0, column=1, sticky=W, padx=5, pady=5)
    rt.mainloop()
```

```
def date(locn, name):
    adt = time.ctime(os.path.getatime(f'{locn}/{name}'))
    mdt = time.ctime(os.path.getmtime(f'{locn}/{name}'))
    cdt = time.ctime(os.path.getctime(f'{locn}/{name}'))
    rt = Tk()
    lbl = Label(rt, text=f'last accessed time of {name}: ')
    lbl1 = Label(rt, text=f'last modified time of {name}: ')
    lbl2 = Label(rt, text=f'{name} created time: ')
    lbl.grid(row=2, column=0, sticky=E, padx=5, pady=5)
    lbl1.grid(row=1, column=0, sticky=E, padx=5, pady=5)
    lbl2.grid(row=0, column=0, sticky=E, padx=5, pady=5)
    lbl3 = Label(rt, text=f'{adt}')
    lbl4 = Label(rt, text=f'{mdt}')
    lbl5 = Label(rt, text=f'{cdt}')
    lbl5.grid(row=0, column=1, sticky=W, padx=5, pady=5)
    lbl4.grid(row=1, column=1, sticky=W, padx=5, pady=5)
    lbl3.grid(row=2, column=1, sticky=W, padx=5, pady=5)
    rt.mainloop()


def open_dir(lokn, fl):
    lcn_etr.delete(0, END)
    lcn_etr.insert(0, f'{lokn}/{fl}')
    get_files()


def type_fl(lokn,fl):
    if '.' not in fl:
        if os.path.isdir(os.path.join(lokn, fl)):
            tp='Directory'
        else:
            tp='File'
    ext=fl.split('.')
    ext='.'+ext[-1]
```

d ={'.apk': 'Android Package$Executable File ','.cpp':'C plus plus file$Executable File','.java':'Java File$Executable File','.bat': 'Batch file$Executable File ', '.bin': 'Binary file$Executable File ', '.cgi': 'Perl script file$Web File ', '.com': 'MS-DOS command file$Executable File ', '.exe': 'Executable file$Executable File ', '.jar': 'Java Archive file$Executable File ', '.py': 'Python file$Executable File ', '.wsf': 'Windows Script File$Executable File ', '.aif': 'AIF/Audio Interchange audio file$Audio File ', '.cda': 'CD audio track file$Audio File ', '.iff': 'Interchange File Format$Audio File ', '.mid': 'MIDI audio file$Audio File ', '.midi': 'MIDI audio file$Audio File ', '.mp3': 'MP3 audio file$Audio File ', '.mpa': 'MPEG-2 audio file$Audio File ', '.wav': 'WAVE file$Audio File ', '.wma': 'Windows Media audio file$Audio File ', '.wpl': 'Windows Media Player playlist$Audio File ', '.avi': 'Audio Video Interleave File$Audio File ', '.flv': 'Adobe Flash Video File$Video File ', '.h264': 'H.264 video File$Video File ', '.m4v': 'Apple MP4 video File$Video File ', '.mkv': 'Matroska Multimedia Container$Video File ', '.mov': 'Apple QuickTime movie File$Video File ', '.mp4': 'MPEG-4 Video File$Video File ', '.mpg': 'MPEG video File$Video File ', '.mpeg': 'MPEG video File$Video File ', '.rm': 'Real Media File$Video File ', '.swf': 'Shockwave flash File$Video File ', '.vob': 'DVD Video Object File$Video File ', '.wmv': 'Windows Media Video File$Video File ', '.3g2': '3GPP2 Multimedia File$Video File ', '.3gp': '3GPP multimedia File$Video File ', '.doc': 'Microsoft Word File$Text File ', '.docx': 'Microsoft Word file$Text File ', '.odt': 'OpenOffice Writer document file$Text File ', '.msg': 'Outlook Mail Message$Text File ', '.pdf': 'PDF file$Text File ', '.rtf': 'Rich Text Format File$Text File ', '.tex': 'A LaTeX document file$Text File ', '.txt': 'Plain text file$Text File ', '.wks': 'Microsoft Works Word Processor Document file$Text File ', '.wps': 'Microsoft Works Word Processor Document file$Text File ', '.wpd': 'WordPerfect document$Text File$Text File ', '.ods': 'OpenOffice Calc spreadsheet file$Spreadsheet File ', '.xlr': 'Microsoft Works spreadsheet file$Spreadsheet File ', '.xls': 'Microsoft Excel file$Spreadsheet File ', '.xlsx': 'Microsoft Excel Open XML spreadsheet file$Spreadsheet File ', '.key': 'Keynote presentation$Presentation File ', '.odp': 'OpenOffice Impress presentation file$Presentation File ', '.pps': 'PowerPoint slide show$Presentation File ', '.ppt': 'PowerPoint presentation$Presentation File ', '.pptx': 'PowerPoint Open XML presentation$Presentation File ', '.accdb': 'Access 2007 Database File$Database File ', '.csv': 'Comma separated value file$Database File ', '.dat': 'Data file$Database File ', '.db': 'Database file$Database File ', '.dbf': 'Database file$Database File ', '.log': 'Log file$Database File ', '.mdb': 'Microsoft Access database file$Database File ', '.pdb': 'Program Database$Database File ', '.sav': 'Save file (e.g. game save file)$Database File ', '.sql': 'SQL/Structured Query Language database file$Database File ', '.tar': 'Linux / Unix tarball file archive$Database File ', '.bak': 'Backup file$System File ', '.cab': 'Windows Cabinet file$System File ', '.cfg': 'Configuration file$System File ', '.cpl': 'Windows Control panel file$System File ', '.cur': 'Windows cursor file$System File ', '.dll': 'DLL file$System File ', '.dmp': 'Dump file$System File ', '.drv': 'Device driver file$System File ', '.icns': 'macOS X icon resource file$System File ', '.ico': 'Icon file$Image File ', '.ini': 'Initialization file$System File ', '.lnk': 'Windows shortcut file$System File ', '.msi': 'Windows installer package$System File ', '.sys': 'Windows system file$System File ', '.tmp': 'Temporary file$System File ',

```
'.asp': 'Active Server Page file$Web File ', '.aspx': 'Active Server Page file$Web File ',
'.cer': 'Internet security certificate$Web File ', '.cfm': 'ColdFusion Markup file$Web File
', '.pl': 'Perl script file$Web File ', '.css': 'Cascading Style Sheet file$Web File ', '.htm':
'HTML/Hypertext Markup Language file$Web File ', '.html': 'HTML/Hypertext
Markup Language file$Web File ', '.js': 'JavaScript file$Web File ', '.jsp': 'Java Server
Page file$Web File ', '.part': 'Partially downloaded file$Web File ', '.php': 'PHP Source
Code file$Web File$Web File ', '.rss': 'RSS/Rich Site Summary file$Web File ',
'.xhtml': 'XHTML / Extensible Hypertext Markup Language file$Web File ', '.ai':
'Adobe Illustrator file$Image File ', '.bmp': 'Bitmap image File$Image File ', '.gif': 'GIF/
Graphical Interchange Format image$Image File ', '.jpeg': 'JPEG image$Image File ',
'.jpg': 'JPEG image$Image File ', '.max': '3ds Max Scene File$Image File ', '.obj':
'Wavefront 3D Object File$Image File ', '.png': 'PNG / Portable Network Graphic
image$Image File ', '.ps': 'PostScript file$Image File ', '.psd': 'PSD / Adobe Photoshop
Document image$Image File ', '.svg': 'Scalable Vector Graphics file$Image File ', '.tif':
'TIFF image$Image File ', '.tiff': 'TIFF image$Image File ', '.3ds': '3D Studio
Scene$Image File ', '.3dm': 'Rhino 3D Model$Image File '}


tp=d.get(f'{ext}','File$File')
ind=tp.index('$')
rt=Tk()
lbl=Label(rt,text='File Type: ')
lbl.grid(row=0, column=0, sticky=E, padx=5, pady=5)
lbl1=Label(rt,text=f'{tp[ind+1:]}')
lbl1.grid(row=0, column=1, sticky=W, padx=5, pady=5)
if ext!=fl:
    lbl2=Label(rt,text=f'Extension {ext}')
    lbl2.grid(row=1, column=0, sticky=E, padx=5, pady=5)
    lb3=Label(rt,text=f'{tp[:ind]}')
    lb3.grid(row=1, column=1, sticky=W, padx=5, pady=5)
rt.mainloop()
```

```
def download(lokn,fl):
    try:
        locn='down:'+f'{lokn}/{fl}'
        s.sendall(locn.encode())
        fil=open(f'{fl}','wb')
        sz=0
        while True:
                info=s.recv(1024)
                print(info,len(info))
                if len(info)<1024:
                        print(info,len(info))
                        fil.write(info)
                        fil.close()
                        print('quit')
                        break


        sz+=len(info)
        fil.write(info)
        print(sz,'characters copied')
    except Exception as e:
        print(e)


def delete(locn,fl):
    lokn='del:'+f'{locn}/{fl}'
    s.sendall(lokn.encode())
    msg=s.recv(1024).decode()
    if msg=='done':
       print('deleted')
       get_files()

nb = 0
def delete_dir(locn,fl):

    rt=Tk()
    lbl=Label(rt,text=f'Do you really want to delete {fl} ')
    lbl.grid(row=0,column=0,padx=5,pady=5)
    lbl1 = Button(rt, text=f'Yes ',command=lambda locn=locn,fl=fl:del_dir(rt,locn,fl))
    lbl2 = Button(rt, text=f'No ',command=lambda :rt.destroy())
    lbl1.grid(row=1, column=0, padx=5, pady=5)
    lbl2.grid(row=1, column=1, padx=5, pady=5)
```

```
def del_dir(rt,locn,fl):
    lokn='deld:'+f'{locn}/{fl}'
    s.sendall(lokn.encode())
    msg=s.recv(1024).decode()
    rt.destroy()
    if msg=='done':
        print('deleted')
        get_files()

def up_load_a(lokn_c,k):
    fl=open(lokn_c,'rb')

    if k==1:
        s.sendall(('upld:'+lcn_etr.get()+'/'+lokn_c.split('/')[-1]+f'{k}').encode())
    else:
        s.sendall(('upld:' + lcn_etr.get() + '/' + lokn_c.split('/')[-2]+'/'+lokn_c.split('/')[-
1]+f'{k}').encode())
    size=0
    while True:
        info=fl.read(1024)
        print(info)
        if len(info)<1:
            #c.send(b'end')
            fl.close()
            print('quit')
            break
        size+=len(info)

        s.send(info)
    print(size)
    msg=s.recv(1024).decode()
    if msg=='recv':
        print('uploaded',lokn_c.split('/')[-1])

def up_load():
    lokn=up_etr.get()
    if os.path.isfile(lokn):
        k=1
        up_load_a(lokn,k)
        get_files()

    if os.path.isdir(lokn):
        for file in os.listdir(lokn):
            print(file)
            k=2
            up_load_a(lokn+'/'+file,k)
        get_files()
```

```
def back():
    lcn=lcn_etr.get()
    l=lcn.split('/')
    s=''
    for i in range(len(l)-1):
        s+=l[i]+'/'
    lcn_etr.delete(0,len(lcn))
    lcn_etr.insert(0,s)
    get_files()
def get_files():
    global nb
    # global i
    locn = 'locn:' + lcn_etr.get()
    s.sendall(locn.encode())
    # con=0
    for widget in second_frame.winfo_children():
        widget.destroy()
    text1 = []
    d = {}
    i = 0
    while True:

        data = s.recv(1024)
        print(data,'data')
        data=data.decode()
        # con+=1
        # print(con)
        if data == 'endlast':
            break
        t = 0

        for file in data.split('&'):
            if file == 'endlast':
                t = 1
                break
            if file == '':
                continue
            if '$' in file:
                continue
            text1.append(f'{file}')
```

```
 if os.path.isdir(os.path.join(locn[5:], file)):
            menubutton = Menubutton(second_frame, width=50, text=f'{file}',
fg='#ff1944', bg='green')
            menubutton.grid(row=nb, column=1, padx=1, pady=1)
            menubutton.menu = Menu(menubutton, tearoff=0, activeborderwidth=25)

            menubutton["menu"] = menubutton.menu

            menubutton.menu.add_command(label="Rename", command=lambda i=i:
re_name(locn[5:], text1[i]))

            menubutton.menu.add_command(label="Size ", command=lambda i=i:
size(locn[5:], text1[i]))
            menubutton.menu.add_command(label="Date Info", command=lambda i=i:
date(locn[5:], text1[i]))
            menubutton.menu.add_command(label="Open", command=lambda i=i:
open_dir(locn[5:], text1[i]))
            menubutton.menu.add_command(label="Type", command=lambda i=i:
type_fl(locn[5:],text1[i]))
            menubutton.menu.add_command(label="Delete",command=lambda i=i:
delete_dir(locn[5:],text1[i]))
            nb = nb + 1
            i=i+1
            print(i)
            continue

        # if os.path.isdir(os.path.join(locn[5:], file)):
        menubutton = Menubutton(second_frame, width=50, text=f'{file}',
                                bg='blue')  # ,command=btn_clicked(os.path.join(locn, file)))
        menubutton.grid(row=nb, column=1, padx=1, pady=1)
        menubutton.menu = Menu(menubutton, tearoff=0, activeborderwidth=25)
        menubutton["menu"] = menubutton.menu

        menubutton.menu.add_command(label="Rename", command=lambda i=i:
re_name(locn[5:], text1[i]))

        menubutton.menu.add_command(label="Size ", command=lambda i=i:
size(locn[5:], text1[i]))
```

```
menubutton.menu.add_command(label="Date info", command=lambda i=i:
date(locn[5:], text1[i]))
        menubutton.menu.add_command(label="Download",command=lambda
i=i:download(locn[5:],text1[i]))
        menubutton.menu.add_command(label="Type", command=lambda i=i:
type_fl(locn[5:],text1[i]))
        menubutton.menu.add_command(label="Delete", command=lambda i=i:
delete(locn[5:], text1[i]))

            nb = nb + 1
            i = i + 1
            print(i)
            #    continue
            # file_btn = tk.Button(frame1, width=300, text=f'{file}')
            # file_btn.pack()
            print(file)

        if t:
            print(nb)
            break

frameh = Frame(root)
frameh.pack()
# create a main frame
main_frame = Frame(root)
main_frame.pack(fill=BOTH, expand=1)

# create a canvas
my_canvas = Canvas(main_frame)
my_canvas.pack(side=LEFT, fill=BOTH, expand=1)

# Add a scroll bar to canvas
scroll_bar = Scrollbar(main_frame, orient=VERTICAL, command=my_canvas.yview)
scroll_bar.pack(side=RIGHT, fill=Y)

# configure canvas
my_canvas.configure(yscrollcommand=scroll_bar.set)
my_canvas.bind('<Configure>', lambda e:
my_canvas.configure(scrollregion=my_canvas.bbox("all")))

# create another frame in canvas
second_frame = Frame(my_canvas)

# add the new frame to a window in the canvas
my_canvas.create_window((0, 0), window=second_frame, anchor='nw')
```

```
Addr_label = Label(frameh, text="Server Address : ")
Addr_label.grid(row=0, column=0, sticky=E, padx=10, pady=10)
Addr_etr = Entry(frameh)
Addr_etr.grid(row=0, column=1, sticky=W, padx=10, pady=10)
port_label = Label(frameh, text="Port No : ")
port_label.grid(row=0, column=2, sticky=E, padx=10, pady=10)

port_etr = Entry(frameh)
port_etr.grid(row=0, column=3, sticky=W, padx=10, pady=10)
cnt_button = Button(frameh, text="Connect", width=20, command=set_connection)
cnt_button.grid(row=0, column=4,padx=10,pady=10)
lcn_label = Label(frameh, text="Enter Location : ")
lcn_label.grid(row=0, column=5, sticky=E,padx=10,pady=10)
lcn_etr = Entry(frameh)
lcn_etr.grid(row=0, column=6, sticky=W,padx=10,pady=10)
get_files_btn = Button(frameh, text="GET files", command=get_files)
get_files_btn.grid(row=0, column=7,padx=10,pady=10)
up_label=Label(frameh,text='File location: ')
up_label.grid(row=1,column=1,sticky=E,padx=10,pady=10)
up_etr=Entry(frameh)
up_etr.grid(row=1,column=2,sticky=W,padx=10,pady=10)
up_btn=Button(frameh,text='UPLOAD',width=20,command=up_load)
up_btn.grid(row=1,column=3,padx=10,pady=10)
bc_btn=Button(frameh,text='BACK',width=20,command=back)
bc_btn.grid(row=1,column=0,sticky=W,padx=10,pady=10)
# for thing in range(100):
#    Button(second_frame,text=f'Button {thing}
bro').grid(row=thing+1,column=0,pady=10,padx=10)

root.mainloop()
```

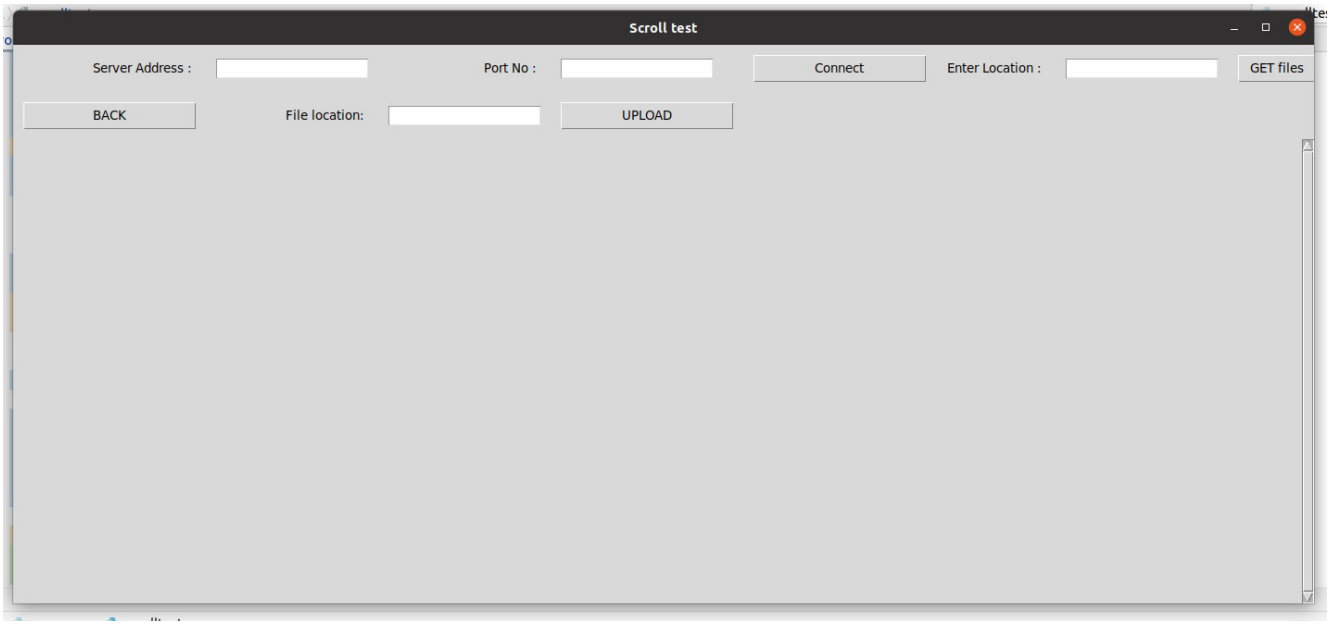
Server Code

```
import socket
import os
import shutil
from _thread import *

#  Socket creation
s = socket.socket()
try:
        s.bind(("192.168.43.152",9999))
        s.listen(5)
ThreadCount=0

print('socket created')

print('waiting for connections')

global address

def do_service(c):

   while True:
      request = c.recv(1024).decode()

      if request[:5] == 'locn:':
         print(request[5:],'locn')
         if '$' not in request[5:]:
            try:
               send_file_names(c, request[5:])
             except Exception as e:
               print(e)
      if request[:5] == 'down:':
         print(request[5:],'down')
         try:
               send_file(c,request[5:])
         except Exception as e:
               print(e)
      if request[:4] == 'del:':
         print(request[4:],'del')
         try:
               del_file(c,request[4:])
         except Exception as e:
               print(e)
```

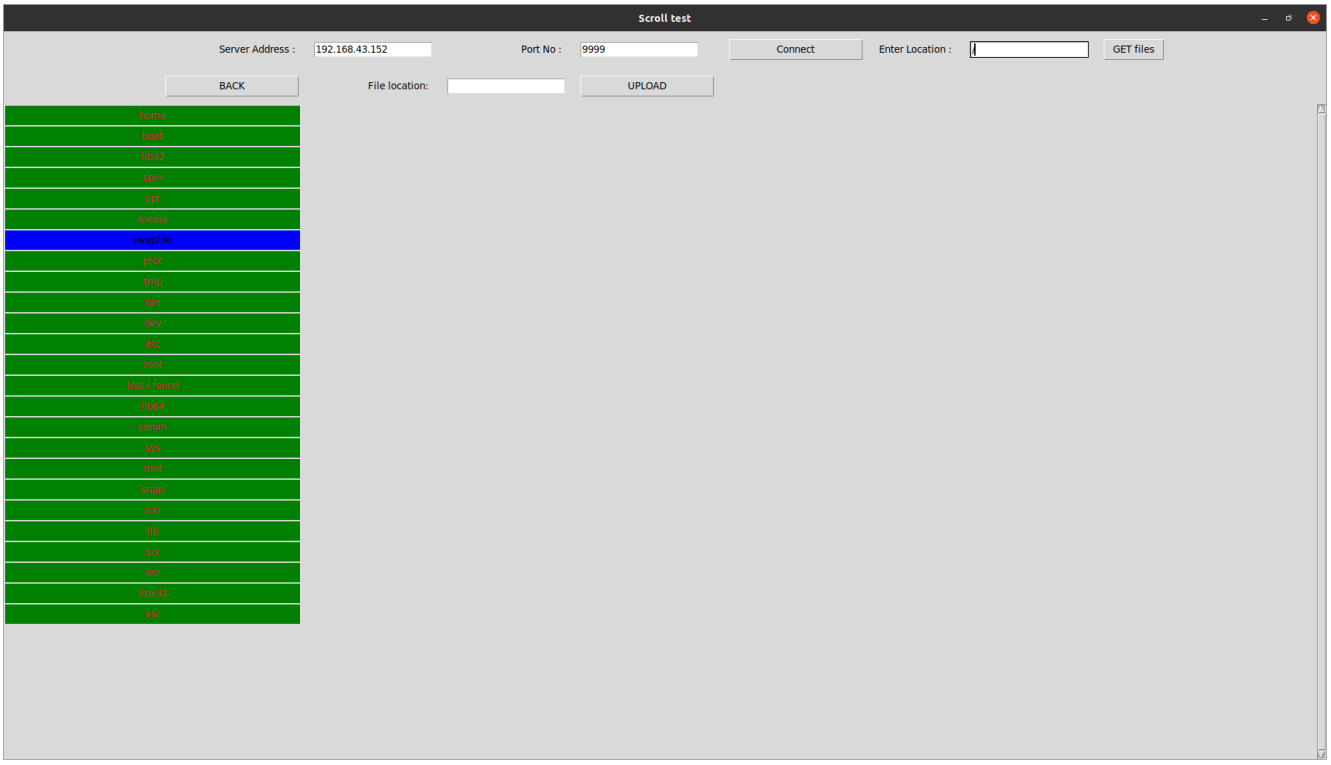
```
 if request[:5] == 'deld:':
            print(request[5:],'deld:')
            try:
                  del_dir(c,request[5:])
            except Exception as e:
                  print(e)
        if request[:5]=='upld:':
            print(request[5:],'upld:')
            try:
                  recv_fl(c,request[5:])
            except Exception as e:
                  print(e)
     c.close()

def recv_fl(c,locn):

    k=int(locn[-1])
    if k>1:
        l=locn.split('/')
        ln=len(l)
        st=''
        for i in range(ln-1):
            st+=l[i]+'/'
            if os.path.isdir(st):
                pass
            else:
                os.mkdir(st)
    sz=0
    fil = open(locn[:-1], 'wb')
    while True:
        info=c.recv(1024)
        print(info,len(info))
        if len(info)<1024:
            print(info,len(info))
            fil.write(info)
            print(info)
            fil.close()
            print('quit')
            break

        sz+=len(info)
        print(info)
        fil.write(info)
    print(locn,'recieved')
    c.sendall('recv'.encode())
```

```
def del_dir(c,dir):
    shutil.rmtree(dir)
    c.sendall('done'.encode())
def del_file(c,file):
    os.remove(file)
    c.sendall('done'.encode())

def send_file(c,file):
    fl=open(file,'rb')
    size=0
    while True:
        info=fl.read(1024)
        if len(info)<1:
            #c.send(b'end')
            fl.close()
            print('quit')
            break
        size+=len(info)

        c.send(info)



def send_file_names(c, dire):
    for file in os.listdir(dire):
        print(file)
        c.sendall(file.encode())
        c.sendall('&'.encode())
    c.sendall('endlast'.encode())


while True:
    try:
        c, address = s.accept()
    except Exception as e:
        print(e)
    print('Connected with ', address)
    start_new_thread(do_service,(c,))
    ThreadCount+=1
    print('Thread Number:'+str(ThreadCount))
s.close()
```

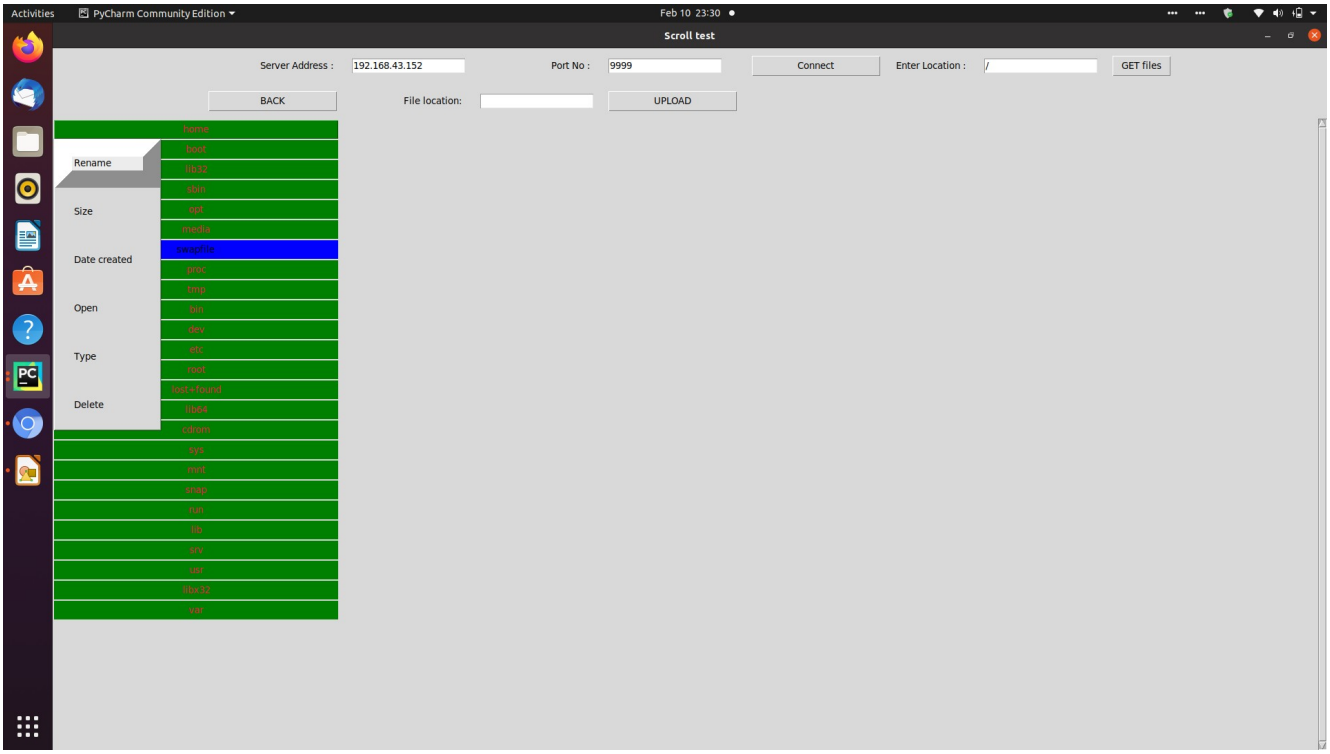
Snapshots



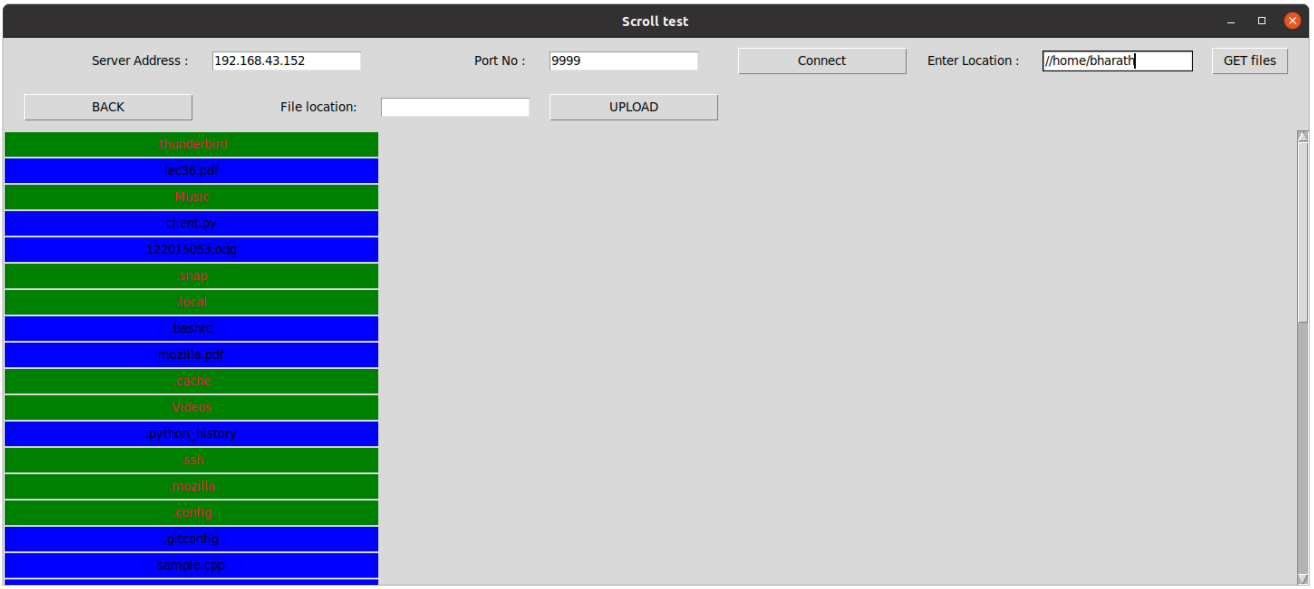
Client.py GUI



Files in / directory

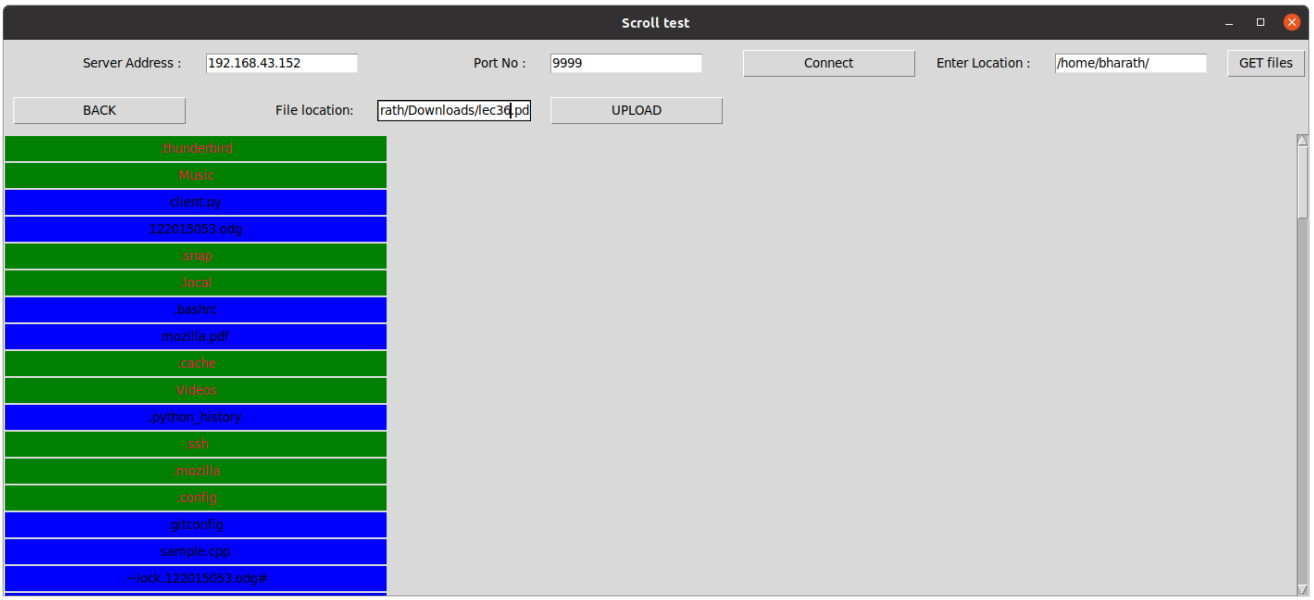


Available options for a file



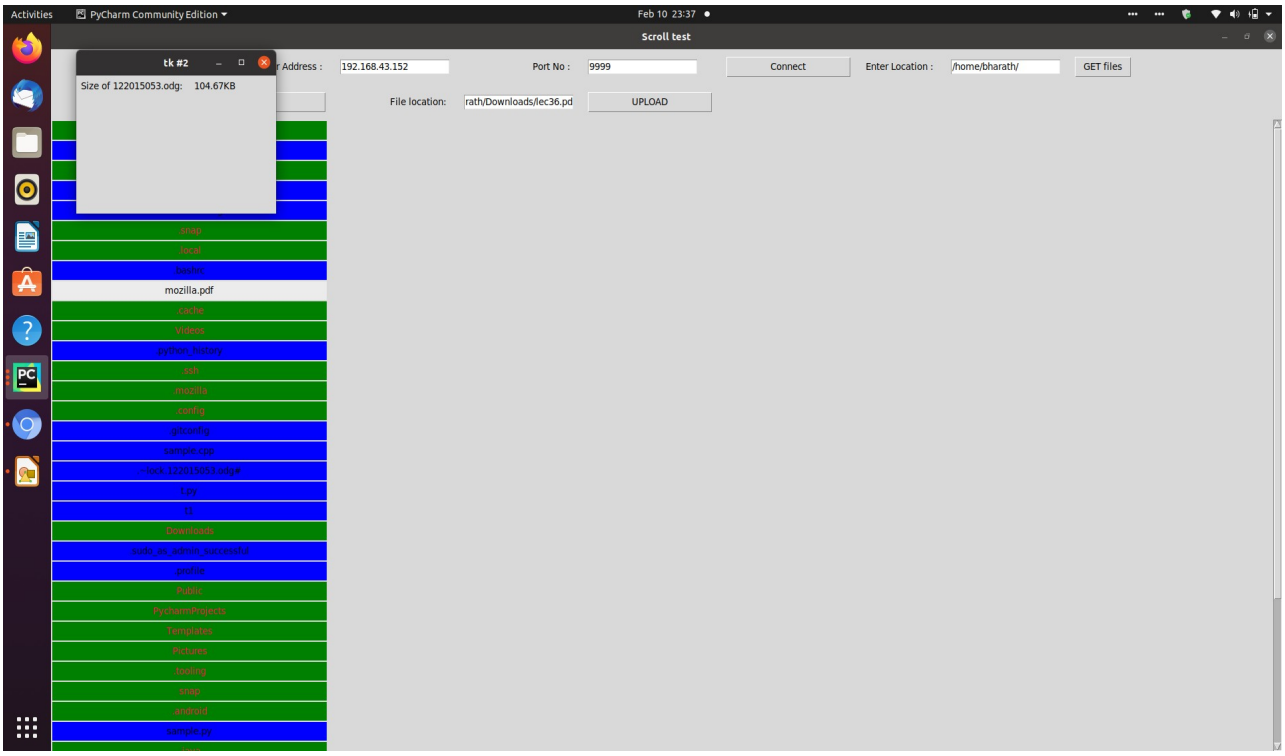
Opening /*home*/bharath through GUI

Opening *home*/bharath/Downloads via GUI

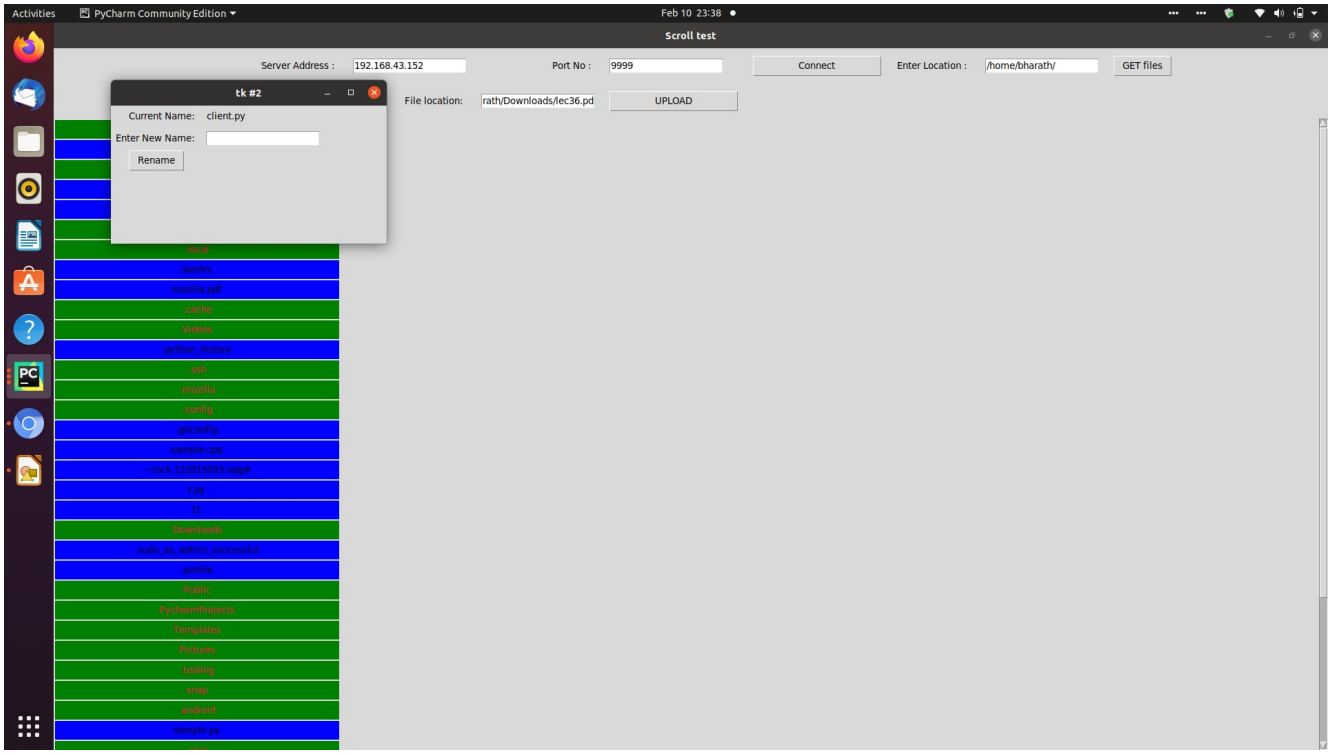


Coming back to *home*/bharath by clicking Back button

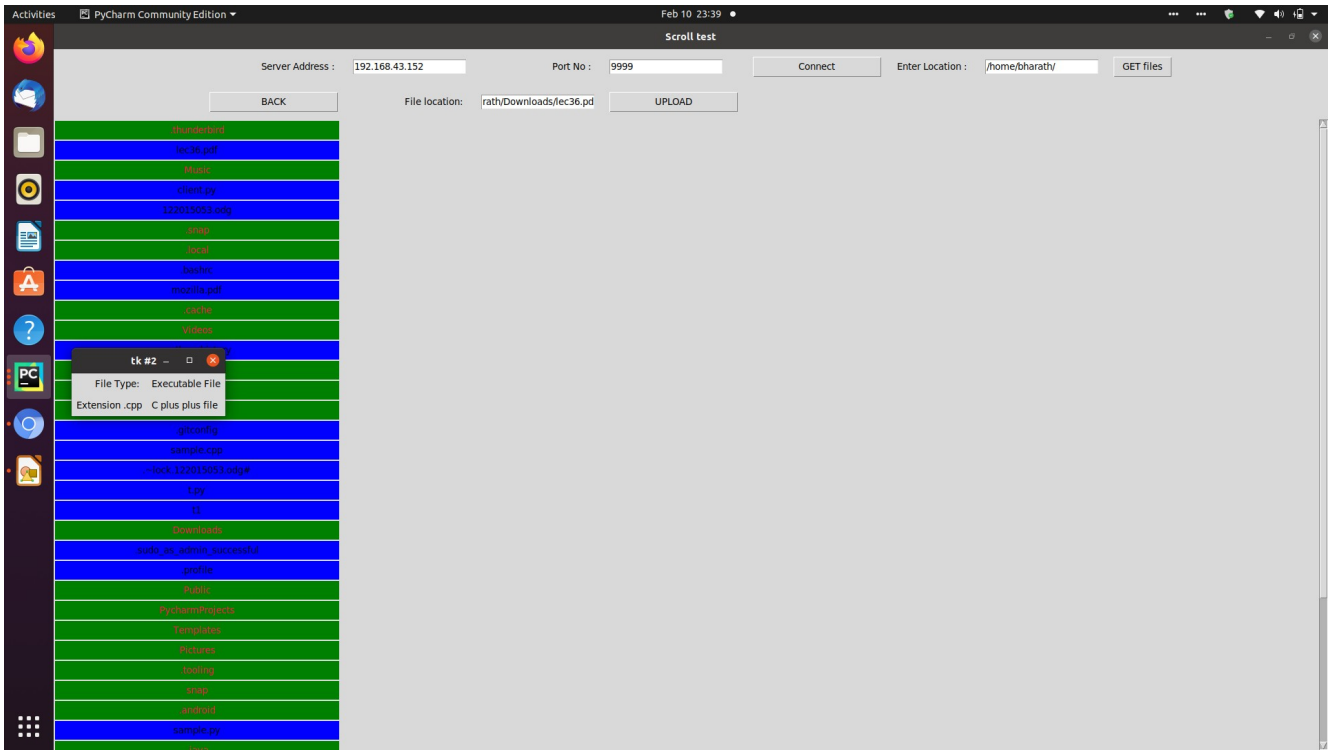
Uploading lec36.pdf file to *home*/bharath



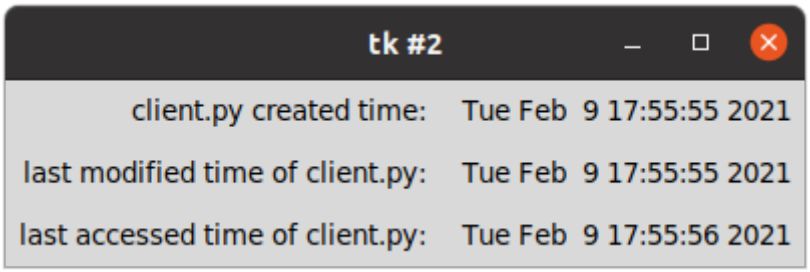
Viewing Size of a file



Viewing Rename widget



Viewing File Type Widget



Date info of Cliet.py

Conclusions and Future Plans

In this project FTP protocol was implemented by creating two python programs each one running on a different computer.Server Client connection is established to upload and download files and to share data between each other.

Multi threading is enabled on server side such that any number of clients could be connected to the server.Various python modules such as tkinter,os,time,shutil,socket has been used in the programs.

This project has helped me to understand socket programming and networking better. During the implementation of the project I gained in-depth knowledge of socket programming and python modules such as socket,os modules etc.

There can be further improvement to this project.

Future plans are

i)To encrypt the entire data going through and from ports so that data will be secured.

ii)To integrate a chat window with this project to enable communication with Server administrator so that suitable permissions can be granted for clients to read or write to or from certain locations in the Server side.

iii)Adding authentication such that only authourised clients can make the connections.

REFERENCES

https://docs.oracle.com/javase/tutorial/networking/sockets/definition.html

https://en.wikipedia.org/wiki/File_Transfer_Protocol#History_of_FTP_servers

https://www.easytechjunkie.com/what-is-socket-programming.htm

https://monovm.com/blog/what-is-ftp-and-its-uses/