Part 32
-
FTP - SFTP
Client & Python

Version: 2018-03-18

FTP client

There is no FTP client installed on the Raspbian OS. That is because SFTP client is the preferred way to transfer files. Therefor to use a simple FTP client, we must install it.

```
apt -y install ftp
```

This will install a command line FTP client that you just have to call to use

```
ftp
ftp>
```

To get the list of all commands, just type ?

```
ftp> ?
Commands may be abbreviated.  Commands are:

!               dir             mdelete         qc              site
$               disconnect      mdir            sendport        size
account         exit            mget            put             status
append          form            mkdir           pwd             struct
ascii           get             mls             quit            system
bell            glob            mode            quote           sunique
binary          hash            modtime         recv            tenex
bye             help            mput            reget           tick
case            idle            newer           rstatus         trace
cd              image           nmap            rhelp           type
cdup            ipany           nlist           rename          user
chmod           ipv4            ntrans          reset           umask
close           ipv6            open            restart         verbose
cr              lcd             prompt          rmdir           ?
delete          ls              passive         runique
debug           macdef          proxy           send
ftp>
```

The basic flow is to `open` a connection, then `put` or `get` a file or files, `close` the connection and `exit`

Use `cd` to change directory, `dir` or `ls` to list a directory

To get full explanations of all commands, type the following at your shell prompt

```
man ftp
```

SFTP client

As mentioned before, by default, Secure FTP or SSH FTP is installed by default on the current Raspbian OS.
The 2 clients that can be used are `sftp` and `scp`

sftp
To use sftp client, type something similar to this where `user1@something.com` should be replaced with your login account on the remote SFTP server. Note: you can omit `-P 22` as it is the default port for SSH FTP

```
sftp -P 22 user1@something.com
user1@something.com's password:
Connected to something.com.
sftp> mkdir www/
sftp> mkdir www/fileadmin/
sftp> mkdir www/fileadmin/subdir/
sftp> cd www/fileadmin/subdir/

sftp> put /tmp/test.html
Uploading /tmp/test.html to /home/user1/www/fileadmin/subdir/test.html
/tmp/test.html                                100%   12    10.9KB/s   00:00

sftp> ls -l
-rw-r--r--    1 user1    user1          12 Jan 10 23:53 test.html
sftp> exit
```

To upload the file in one batch you can execute:

```
sftp user1@something.com:www/fileadmin/subdir/ <<< $'put /tmp/test.html'
user1@something.com's password:
Connected to something.com.
Changing to: /home/user1/www/fileadmin/subdir/
sftp> put /tmp/test.html
Uploading /tmp/test.html to /home/user1/www/fileadmin/subdir/test.html
/tmp/test.html                                100%   12    15.7KB/s   00:00
```

The opposite for downloading is:

```
sftp user1@something.com:www/fileadmin/subdir/test.html /tmp/test.html
user1@something.com's password:
Connected to something.com.
Fetching /home/user1/www/fileadmin/subdir/test.html to /tmp/test.html
/home/user1/www/fileadmin/subdir/test.html       100%   12     8.1KB/s
00:00
```

SCP (Secure Copy)
`scp` is a command for sending files over SSH. This means you can copy files between computers, say from your Raspberry Pi to your desktop or laptop, or vice-versa.

Copying files from your Raspberry Pi to remote server
Eg. copy the file myfile.txt from the pi user's home folder on your Raspberry Pi to the SFTP server in the default directory which has IP 192.168.1.3 and for which you have an account user1

```
scp myfile.txt user1@192.168.1.3:
```

Copy the file to project directory which resides in the default directory of the SFTP server. Note : the project folder must already exist

```
scp myfile.txt user1@192.168.1.3:project/
```

Copying files from the SFTP server to your Raspberry Pi
Copy the file myfile.txt from the current directory on the SFTP server to the home directory of the user you are using normaaly pi:

```
scp user1@192.168.1.3:myfile.txt .
```

Copying multiple files
Copy multiple files by separating them with spaces:

```
scp myfile.txt myfile2.txt user1@192.168.1.3:
```

Alternatively, use a wildcard to copy all files matching a particular search with:

all files ending in .txt

```
scp *.txt user1@192.168.1.3:
```

all files starting with m

```
scp m* user1@192.168.1.3:
```

Filenames with spaces
Note that some of the examples above will not work for file names containing spaces. Names like this need to be encased in quotes:

```
scp "my file.txt" pi@192.168.1.3:
```

Python FTP module

You would be happy to know that ftplib is a built-in library that comes already installed with Python, all you need to do is import it in your script and you can start using its functions.

In this Python programming sample, we cover how to do FTP (file transfer protocol) transfers with ftplib. We'll cover both uploading and downloading files with a remote server.
To start, import it

```
from ftplib import FTP

#domain name or server ip:
ftp = FTP('123.server.ip')
ftp.login(user='username', passwd = 'password')
```

The above will connect you to your remote server. You can then change into a specific directory with:

```
ftp.cwd('/whyfix/')
```

Now, let's show how we might download a file:

```
filename = 'example.txt'

localfile = open(filename, 'wb')
ftp.retrbinary('RETR ' + filename, localfile.write, 1024)

localfile.close()
```

So there are a few things here, so let's walk through it. First, we assign the file name to a variable. Then, we prepare our local file to be written in accordance with whatever the remote file contains. Next, we retrieve the binary data from the remote server, then we write to the local file what we find. The last parameter there, the 1024, is in reference to buffering. Basically, how much data at a time will we do? So at 1024, 1024 byte chunks will be transferred at a time until the full operation is complete.
Next, uploading a file

```
# we assume exampleFile.txt exists in the current directory
filename = 'exampleFile.txt'
ftp.storbinary('STOR '+filename, open(filename, 'rb'))

# close connection nicely
ftp.quit()
```

About the same here, we take file name and assign it to a variable, then we store the binary data to the filename, with the read data from the file name locally.
And then we close the connection.

Full documentation on API of the ftplib module can be found here
https://docs.python.org/3/library/ftplib.html

Python Secure FTP module

What is it?
pysftp is an easy to use sftp module that utilizes paramiko and pycrypto.
It provides a simple interface to sftp. Some of the features are:

- Gracefully handles both RSA and DSS private key files automatically
- Supports encrypted private key files.
- Logging can now be enabled/disabled

It is used to securely exchange files over the Internet.
Using pysftp is easy and we will show some examples on how you can use it

How do I install it?
Make sure you are 'root'

```
sudo -i
```

First update, upgrade, install libssl-dev and install libffi

```
apt-get -y update
apt-get -y upgrade
apt-get -y install libssl-dev
apt-get -y install libffi-dev
```

pysftp is listed on PyPi and can be installed using pip.

```
pip install pysftp
pip3 install pysftp
```

This can take a long, long time. It needs to build wheels for the encryption. Be patient.

Add/Create keys
You will need private and public keys to allow pysftp to work properly and savely. You are still logged in as root, so first let's do this for root as you need to do this for every user who will use the SSH protocol

```
ssh-keygen
```

Now add the host(s) you will access over SSH (with root account)

```
ssh-keyscan <host> >>~/.ssh/known_hosts
```

<host> can be an IP address or a FQDN (Fully Qualified Domain Name)

Hit Crtl-D which will bring you back to your pi account. Do the same as for root.

How to use it?

Here is a sample Python script that can be used as guideline

```
nano test-pysftp.py


import pysftp

host="your.SFTP.server.be"

# Loads .ssh/known_hosts
cnopts = pysftp.CnOpts()

cnopts.hostkeys = None

with pysftp.Connection(host, username="your_username",
password="your_password", cnopts = cnopts) as sftp:

    # change to another directory on remote server
    sftp.cwd('remote_directory')

    # upload file = this script
    sftp.put("test-pysftp.py")

    # list all files and subdirectories in the remote directory
    data = sftp.listdir()

    # Download the file from the remote server
    sftp.get("test-pysftp.py")

    # Closes the connection
    sftp.close()

# Prints out the subdirectories and files, line by line
for i in data:
    print i
```

Full documentation on pysftp can be found here https://pysftp.readthedocs.io