Lucrarea nr. 6
FTP

Mihai IVANOVICI

17 aprilie 2006

FTP (File Transfer Protocol) este unul dintre cele mai vechi şi cele mai utilizate protocoale în Internet. Acest protocol a fost proiectat pentru transferul fişierelor între diverse locaţii. Este un protocol bidirecţional, permiţând atât descărcarea, cât şi încărcarea fişierelor. De asemenea, FTP pune la dispoziţie o serie de comenzi pentru ştergerea sau redenumirea fişierelor, sau pentru crearea şi stergerea de directoare.

În această lucrare de laborator este exemplificată folosirea modulului `ftplib`, interfaţa Python pentru FTP. Cu ajutorul acestuia se pot transfera automat fişiere de la/spre diverse calculatoare aflate la distanţă, se poate obţine lista de directoare de pe un anumit server FTP sau se pot obţine informaţii despre acel server.

1 Introducere

1.1 Canalele de comunicaţie

FTP foloseşte două conexiuni TCP pentru a realiza transferul de fişiere. Una dintre conexiuni este cea de control sau de comenzi, şi este folosită pentru transmiterea de comenzi şi de răspunsuri scurte, cum ar fi confirmări sau coduri de eroare. Cea de-a doua conexiune este canalul de date. Acesta din urmă este folosit pentru transferul fişierelor sau alte informaţii, cum ar fi conţinutul unui director. Canalul de date este full duplex, ceea ce înseamnă că fişierele pot fi transferate în ambele direţii simultan.

Descărcarea unui fişier de la un server FTP are de regulă următoarea succesiune de paşi:

1. Clientul FTP stabileşte o conexiune pentru comenzi, conectându-se la portul FTP al serverului.
2. Clientul se autentifică.
3. Clientul schimbă directorul dorit pe server.

4. Clientul începe să asculte pe un nou port în vederea stabilirii conexiunii de date, după care clientul informează serverul despre acest port.

5. Serverul se conectează pe portul ales de client.

6. Fişierul este transmis şi conexiunea de date închisă.

Această succesiune de paşi funţiona perfect la începutul Internetului, când fiecare maşină care rula FTP avea o adresă IP publică şi când firewall-urile erau rar utilizate. În zilele noastre, firewall-urile sunt întâlnite peste tot, şi mulţi clienţi nu au o adresă IP publică.

Pentru a rezolva această situaţie, FTP suportă ceea ce se cheamă modul pasiv. În modul pasiv, serverul deschide un port şi îi spune clientului unde să se conecteze. În rest clientul şi serverul se comportă la fel.

Modul pasiv este implicit pentru cei mai mulţi clienţi FTP, cât şi pentru modulul Python `ftplib`.

1.2 Autentificarea şi transferul FTP anonim

De regulă, când un client se conectează la un server FTP, acesta va furniza serverului un nume de utilizator şi o parolă, pentru a avea acces la resursele sistemului. FTP pune la dispoziţie şi un al treilea element de autentificare, numit "account", care este rareori folosit.

Odată autentificat, clientul va avea aceleaşi permisiuni pe sistemul aflat la distanţă, ca şi când ar sta la un terminal în faţa serverului.

Mulţi au decis că ar fi folositor să permită utilizatorilor care nu deţin un cont pe server, să aibă totuşi acces în anumite zone ale serverului. Pentru aceasta, "vizitatorii" se autentifică folosind numele de utilizator "anonymous", iar în loc de parolă se foloseşte de regulă adresa de e-mail. Serverele FTP astfel configurate se numesc "servere FTP anonime". Foarte multe site-uri FTP mari folosesc modul anonim, astfel încât oricine vrea un anumit fişier să îl poată obţine. Conceptual, acest comportament este similar aceluia al serverelor de web, pentru care autentificarea nu este necesară.

2 Modulul Python `ftplib`

Dacă se doreşte doar descărcarea unui fişier, se poate folosi la fel de bine modulul Python `urllib2` deja prezentat, care suportă şi protocolul FTP. Pentru celelalte funcţii specifice FTP este necesară utilizarea modulului `ftplib`.

În continuare este prezentat un exemplu de cod care se conectează la un server FTP, afişează mesajul de bun venit al acestuia şi tipăreşte directorul curent.

```
#Simple FTP connection - ftpconnect.py

from ftplib import FTP

f = FTP( 'ftp.ibiblio.org' )
print "Welcome: ", f.getwelcome()
f.login()

print "CWD: ", f.pwd()
f.quit()
```

Mesajul de bun venit este de interes doar pentru un program interactiv. Funcţia `login()` primeşte ca parametri numele utilizatorului, parola şi contul. Apelată fără parametri, funţia va folosi modul anonim de autentificare. Funcţia `pwd()` returnează calea către directorul curent, iar funcţia `quit()` deloghează clientul şi închide conexiunea.

Rulând exemplu de mai sus, veţi obţine următorul rezultat:

```
$ python2.4 ftpconnect.py
Welcome:  220 ProFTPD Server (Bring it on...)
CWD:  /
```

3 Descărcarea fişierelor ASCII

Transferul FTP are loc în două moduri: ASCII şi binar. În modul ASCII, fişierele sunt transferate line cu line, ceea ce permite clientului să salveze fişierele cu terminaţia de linie caracteristică sistemului de fişiere pe care rulează. Iată un exemplu de client FTP care descarcă un fişier în modul ASCII:

```
#FTP ASCII mode - ftpascii.py
#Downloads README from server

from ftplib import FTP

def writeline( data ):
   fd.write( data + "\n" )

f = FTP( 'ftp.kernel.org' )
f.login()

f.cwd( '/pub/linux/kernel' )
fd = open( 'README', 'wt' )
f.retrlines( 'RETR README', writeline )
```

```
fd.close()

f.quit()
```

Funcţia `cwd()` schimbă directorul curent pe server, apoi funţia `retrlines()` va realiza transferul propriu-zis al fişierului README. Primul parametru al acestei funcţii specifică comanda care se va trimie şi executa pe server, care de obicei este `RETR`, urmată de numele fişierului. Cel de-al doilea parametru este o funcţie care este apelată pentru fiecare linie din fişierul transferat.

4 Descărcarea fişierelor binare

Transferul fişierelor binare este foarte asemanator cu cel al fişierelor text. Iată un exemplu:

```
#FTP Binary mode - ftpbinary.py

from ftplib import FTP

f = FTP( 'ftp.kernel.org' )
f.login()

f.cwd( '/pub/linux/kernel/v1.0' )
fd = open( 'patch8.gz', 'wb' )
f.retrbinary( 'RETR patch8.gz', fd.write )
fd.close()

f.quit()
```

În acest exemplu, un fişier binar este transferat. Funcţia `retrbinary()` trimite blocuri de date funcţiei specificate ca parametru.

Modulul `ftplib` pune la dispoziţie o a doua funcţie care poate fi folosită pentru transferul fişierelor binare: funţia `ntransfercmd()`. Această funcţie oferă o interfaţă la un nivel mai jos decât funcţia `retrbinary`. Exemplul de mai jos ilustrează folosirea acestei funcţii.

```
#Advanced Binary Download - advanced_binary.py

from ftplib import FTP
import sys

f = FTP( 'ftp.kernel.org' )
f.login()
```

```
f.cwd( '/pub/linux/kernel/v1.0' )
f.voidcmd( "TYPE I" )

datasock, estsize = f.ntransfercmd( "RETR linux-1.0.tar.gz" )
transbytes = 0
fd = open( 'linux-1.0.tar.gz', 'wb' )
while 1:
   buf = datasock.recv( 2048 )
   if not len( buf ):
    break
   fd.write( buf )
   transbytes += len( buf )
   sys.stdout.write( "Received %d " % transbytes )

   #This "if" only passes if estsize is nonzero and isn't None.
   #That's exactly what we want, since if it's zerp, we'd get a
   #divide-by-zero error.
   if estsize:
      sys.stdout.write( "of %d bytes (%.1f%%)\r" % \
         (estsize, 100.0 * float( transbytes) / float(estsize)))
   else:
      sys.stdout.write( "bytes\r" )
   sys.stdout.flush()

sys.stdout.write( "\n" )
fd.close()
datasock.close()
f.voidresp()

f.quit()
```

Funcţia `voidcmd()` trimite o comandă către server şi verifică eventualele erori, dar nu returnează nimic. Apelul său din exemplul de mai sus trimite comanda `TYPE I`, care setează transferul în modul imagine sau binar. În exemplul precedent, funcţia `retrbinary()` trimitea această comandă, fără ca utilizatorul să ştie.

Funţia `ntransfercmd()` returnează un dublet format dintr-un socket, corespunzător canalului de date, şi o dimensiunea estimativă a fişierului. Dacă dimensiunea estimativă nu este furnizată de serverul FTP, atunci ea va fi `None`.

După primirea fişierului este important să închidem socketul şi să apelăm funcţia `voidresp()`. Aceasta din urmă citeşte răspunsul serverului şi indică o excepţie în cazul unei eventuale erori.

Rulaţi acest exemplu.

5 Încărcarea unui fişier

Similar cu descărcarea fişierelor de pe server, pentru încărcare există două funcţii ce pot fi folosite: `storbinary()` şi `storlines()`. Ambele funcţii primesc ca parametru o comandă FTP şi un obiect de tip fişier.

Funcţia `storbinary()` va apela funcţia `read()` pentru obiectul de tip fişier, pe când funcţia `storlines()` va apela `readline()`.

Iată un exemplu de cod care încarcă un fişier în modul binar:

```
#Binary upload - ftpupload.py
#Arguments: host, username, localfile, remotepath

from ftplib import FTP
import sys, getpass, os.path

host, username, localfile, remotepath = sys.argv[1:]
password = getpass.getpass( "Enter password for %s on %s: " % \
       (username, host) )

f = FTP( host )
f.login( username, password )
f.cwd( remotepath )
fd = open( localfile, 'rb' )
f.storbinary( 'STOR %s' % os.path.basename(localfile), fd )
fd.close()

f.quit()
```

6 Scanarea directoarelor

FTP oferă două modalităţi de a afla informaţii despre fişierele de pe server şi directoare. Acestea sunt implementate în funcţiile `nlst()` şi `dir()`, din modulul `ftplib`.

Funcţia `nlst()` returnează o listă cu intrările într-un anumit director, listă care conţine doar numele fişierelor din director. Nu conţine informaţii despre tipul intrărilor (fişier sau director), dimensiune etc.

Funţia `dir()` returnează o listare a directorului de pe server, într-un format care depinde de sistemul de operare ce rulează pe server. Listarea va conţine numele fişierelor, dimensiunea, data modificării şi tipul. Pe serverele de tip Linux, listarea este de regulă ieşirea apelului `ls -l` sau `ls -al`.

Următorul exemplu foloseşte funcţia `nlst()`:

```
#NLST example - ftpnlst.py
```

```
from ftplib import FTP

f = FTP( 'ftp.kernel.org' )
f.login()

f.cwd( '/pub/linux/kernel' )
entries = f.nlst()
entries.sort()

print "%d entries: " % len( entries )
for entry in entries:
   print entry

f.quit()
```

Un exemplu care foloseşte funcţia `dir()` este prezentat în continuare:

```
#dir example - ftpdir.py

from ftplib import FTP

f = FTP( 'ftp.kernel.org' )
f.login()

f.cwd( '/pub/linux/kernel' )
entries = []
f.dir( entries.append )

print "%d entries: " % len( entries )
for entry in entries:
   print entry

f.quit()
```

Funţia `dir()` primeşte ca parametru o funcţie care este apelată pentru fiecare linie, la fel ca funcţia `retrlines()`.

Rulaţi cele două exemple şi observaţi diferenţele.

7 Manipularea fişierelor şi a directoarelor

Pentru a şterge un fişier se foloseşte funcţia `delete()`, iar pentru a şterge un director se foloseşte funcţia `rmd()`. De regulă serverele cer ca directorul

ce urmează a fi şters să fie gol. Ambele funţii primesc ca argument numele fişierului sau al directorului ce se doreşte a fi şters.

Funcţia `mkd()` se foloseşte pentru crearea unui nou director. Numele noului director este parametru al acestei funcţii.

Pentru redenumirea fişierelor se foloseşte funcţia `rename()` care primeşte doi parametri: numele unui fişier existent şi noul nume al acestuia. Este echivalentă cu funcţia UNIX `mv`.