Lucrarea nr. 1

Programarea în reţea. Introducere

Mihai IVANOVICI

27 februarie 2006

1 Un exemplu de client

Exemplul de mai jos reprezintă unul din cei mai simpli clienţi de reţea, care pot fi folosiţi într-o reţea reală. Implementează protocolul Gopher, larg răspândit înaintea apariţiei WWW-ului.

```
#Simple Gopher Client - gopherclient.py

import socket, sys

port = 70 # Gopher uses port 70
host = sys.argv[ 1 ]
filename = sys.argv[ 2 ]

s = socket.socket( socket.AF_INET, socket.SOCK_STREAM )
s.connect((host, port))

s.sendall( filename + "\r\n" )

while 1:
   buf = s.recv( 2048 )
   if not len( buf ):
      break
   sys.stdout.write( buf )
```

Programul primeşte ca intrare doi parametri: numele unui server Gopher şi numele unui fişier, iar rolul său este de a cere respectivul fişier de la server şi de a-l tipări pe ecran.

Funcţionarea programului este următoarea: crează un **socket**, invocând metoda `socket.socket()` din modulul `socket`. Argumentele funcţiei indică faptul că se doreşte o comunicaţie TCP folosing un socket pentru streaming.

Apoi programul se conectează la server şi îi trimite acestuia numele fişierului care se doreşte a fi transferat.

Rulaţi exemplul, folosind următoarea linie de comandă:

```
$ python2.3 gopherclient.py quux.org /
```

Rezultatul rulării va fi afişarea unei liste a conţinutului directorului "root" a serverului Gopher quux.org.

1.1 Erori şi excepţii

Python se ocupă de tratarea excepţiilor, la fel ca majoritatea limbajelor de programare. Încercaţi să rulaţi acelaşi program, specificând un nume de server inexistent:

```
$ python2.3 gopherclient.py nonexistant.example.com /
Traceback (most recent call last):
  File "gopherclient.py", line 9, in ?
    s.connect((host, port))
  File "<string>", line 1, in connect
socket.gaierror: (-2, 'Name or service not known')
```

Python a detectat o eroare şi a generat o excepţie `socket.gaierror`. Deoarece programul nu tratează în mod explicit excepţiile, acesta s-a terminat în mod abrupt şi a tipărit informaţii despre unde a apărut eroarea (în funcţia `s.connect((host,port))` şi care a fost cauza erorii ("Nume sau serviciu necunoscut").

```
#Simple Gopher Client - gopherclient2.py

import socket, sys

port = 70 # Gopher uses port 70
host = sys.argv[ 1 ]
filename = sys.argv[ 2 ]

s = socket.socket( socket.AF_INET, socket.SOCK_STREAM )

try:
   s.connect((host, port))
except socket.gaierror, e:
   print "Error connecting to server: %s" % e
   sys.exit( 1 )

s.sendall( filename + "\r\n" )
```

```
while 1:
   buf = s.recv( 2048 )
   if not len( buf ):
      break
   sys.stdout.write( buf )
```

Rulaţi programul `gopherclient2.py` şi observaţi modul în care a fost tratată eroarea şi faptul că programul s-a terminat în mod normal.

1.2 Obiecte de tip fişier

Python oferă suport pentru lucrul cu fişiere şi obiecte de tip fişier, şi funcţii de tipul `readline()`, `write()` şi `read()`.

Obiectele de tip `socket` prin apelarea metodei `makefile()` generează un obiect de tip fişier. Funcţia primeşte două argumente: modul de deschidere a fişierului (read, write sau read/write) şi modul de bufferare (0 indică dezactivarea buffer-ului intermediar).

```
#Simple Gopher Client - gopherclient3.py

import socket, sys

port = 70 # Gopher uses port 70
host = sys.argv[ 1 ]
filename = sys.argv[ 2 ]

s = socket.socket( socket.AF_INET, socket.SOCK_STREAM )
s.connect((host, port))

fd = s.makefile( 'rw', 0 )

fd.write( filename + "\r\n" )

for line in fd.readlines():
   sys.stdout.write( line )
```

2 Un exemplu de server

În continuare este prezentat codul Python al unui server:

```
#Simple Server - server.py

import socket
```

```
host = ''#Bind to all interfaces
port = 55999

s = socket.socket( socket.AF_INET, socket.SOCK_STREAM )
s.setsockopt( socket.SOL_SOCKET, socket.SO_REUSEADDR, 1 )
s.bind((host, port))
s.listen( 1 )

print "Server is running on port %d; press Ctrl-C to terminare." % port

while 1:
   clientsock, clientaddr = s.accept()
   clientfile = clientsock.makefile( 'rw', 0 )
   clientfile.write( "Welcome, " + str( clientaddr ) + "\n" )
   clientfile.write( "Please enter a string: " )
   line = clientfile.readline().strip()
   clientfile.write( "You entered %d characters.\n" % len( line ))
   clientfile.close()
   clientsock.close()
```

La fel ca clientul, serverul va deschide un socket, invocând `socket.socket()`, socketul fiind reutilizabil. Următorul pas este de a lega socketul de un port, de exemplu portul 55999. Parametrul `host` este vid ceea ce înseamnă că serverul acceptă conexiuni de la orice client. Apoi apelul funcţiei `listen()` spune serverului să aştepte conexiuni de la clienţi, fiind capabil să procese o singură conexiune la un moment dat.

Bucla principală a programului începe cu un apel `accept()`: programul se va opri în acest punct până la apariţia unei cereri de conexiune din partea unui client. Când un client se conectează, funcţia `accept` returnează un nou socket conectat cu clientul şi adresa IP a acestuia.

Serverul va trimite clientului câteva mesaje, după care va trimite un răspuns. În final va închide socketul deschis şi în prealabil obiectul fişier.

Rulaţi serverul, folosind linia de comandă:

```
$ python2.3 server.py
```

Într-un alt terminal, folosiţi utilitarul `telnet` pentru a vă conecta la server pe portul 55999:

```
$ telnet localhost 55999
```

Sesiunea telnet va arăta după cum urmează:

```
$ telnet localhost 55999
Trying 127.0.0.1...
Connected to localhost.localdomain (127.0.0.1).
Escape character is '^]'.
Welcome, ('127.0.0.1', 33085)
Please enter a string: Hello Mr. Server
You entered 16 characters.
Connection closed by foreign host.
```

3 Probleme

Problema 1: Într-un terminal porniţi interpretorul Python, tastând `python2.3`. Utilizând funcţia `help` încercaţi să înţelegeţi funcţiile folosite în exemplele prezentate în lucrare. De exemplu: `>>> help(socket.socket.accept())`.

Problema 2: rulaţi clientul (`telnet`) şi serverul (`server.py`) pe maşini diferite şi observaţi diferenţele.