Laboratorio: hget

Cátedra de Redes y Sistemas Distribuidos
Revisión 2012, Eduardo Sanchez
Revisión 2011, Eduardo Sanchez
Original 2009-2010, Natalia Bidart (C), Daniel Moisset (Python)

Objetivos

- Leer y comprender código Python no-trivial.
- Realizar un primer contacto con la programación de sockets.
- Familiarizarse con las llamadas necesarias para utilizar sockets.
- Entender la noción básica de comunicación cliente/servidor.
- Ver los aspectos esenciales de como funciona un cliente de una aplicación de redes.

Tarea

El laboratorio consiste en completar la aplicación `hget` que obtiene una página web desde un servidor y la almacena en un archivo. La aplicación se comunica con servidores usando el protocolo HTTP (definido en el RFC 2616). Es decir, hget es un *cliente HTTP*.

El código de hget está esencialmente ya escrito y se lo damos, con solamente una de las funciones incompletas: `connect_to_server()`.

Su tarea es completar el cuerpo de esta función para que el cliente funcione correctamente.

Ejemplo

A continuación daremos un ejemplo de la ejecución, suponiendo que `hget.py` se encuentra en un directorio sin ningún otro archivo:

```
$ ls
hget.py
$ python hget.py http://www.unc.edu.ar/
Contactando servidor 'www.unc.edu.ar'...
Contactando al servidor en 200.16.16.61...
Enviando pedido...
Esperando respuesta...
$ ls
download.html  hget.py
```

Luego de solicitar una página por su URL, obtenemos un archivo llamado `download.html`. Puede elegirse un nombre de archivo distinto con la opción `-o`:

```
$ python hget.py http://tools.ietf.org/html/rfc2616 -o estandar-http.html
Contactando servidor 'tools.ietf.org'...
Contactando al servidor en 208.66.40.242...
Enviando pedido...
Esperando respuesta...
$ ls
download.html  estandar-http.html  hget.py
```

Tests

En el código fuente que les entregamos, pueden ver que todas las funciones tienen un*docstring*, al principio de la función que describe su comportamiento.

Algunas de estas funciones tienen ejemplos de uso desde el *intérprete interactivo*obtenidos copiando y pegando una sesión. Estos ejemplos, además de funcionar como documentación para aclarar lo que hace la función, pueden ser utilizados como tests automáticos (llamados *doctests* por

ser tests que están incluidos en la documentación). Para ejecutar estos tests pueden hacer:

```
$ python -m doctest hget.py
Contactando al servidor en 200.16.17.104...
Contactando al servidor en 200.16.17.187...
```

Si el programa supera los tests, la salida de la ejecución de los mismos será vacía (excepto por los mensajes que impriman las funciones testeadas, como en el ejemplo de arriba).

Cuando la ejecución de los tests fallan, obtendrán un listado de los tests que fallaron, el error que se produjo, y un resumen de cuantos tests fallaron:

```
eduardo@drubi:~/docencia/redes/labs/lab0$ python -m doctest hget.py
**********************************************************************
File "hget.py", line 75, in hget.connect_to_server
Failed example:
    connect_to_server('www.famaf.unc.edu.ar') # doctest: +ELLIPSIS
Exception raised:
    Traceback (most recent call last):
      File "/usr/lib/python2.6/doctest.py", line 1253, in __run
        compileflags, 1) in test.globs
      File "", line 1, in
        connect_to_server('www.famaf.unc.edu.ar') # doctest: +ELLIPSIS
      File "hget.py", line 95, in connect_to_server
        sys.stderr.write("Contactando al servidor en %s...\n" % ip_address)
    NameError: global name 'ip_address' is not defined
**********************************************************************
File "hget.py", line 78, in hget.connect_to_server
Failed example:
    connect_to_server('no.exis.te') # doctest: +IGNORE_EXCEPTION_DETAIL
Expected:
    Traceback (most recent call last):
       ...
    gaierror: [Errno -5] No address associated with hostname
Got:
    Traceback (most recent call last):
      File "/usr/lib/python2.6/doctest.py", line 1253, in __run
        compileflags, 1) in test.globs
      File "", line 1, in
        connect_to_server('no.exis.te') # doctest: +IGNORE_EXCEPTION_DETAIL
      File "hget.py", line 95, in connect_to_server
        sys.stderr.write("Contactando al servidor en %s...\n" % ip_address)
    NameError: global name 'ip_address' is not defined
**********************************************************************
File "hget.py", line 83, in hget.connect_to_server
Failed example:
    connect_to_server('shiva.famaf.unc.edu.ar')
Exception raised:
    Traceback (most recent call last):
      File "/usr/lib/python2.6/doctest.py", line 1253, in __run
        compileflags, 1) in test.globs
      File "", line 1, in
        connect_to_server('shiva.famaf.unc.edu.ar')
      File "hget.py", line 95, in connect_to_server
        sys.stderr.write("Contactando al servidor en %s...\n" % ip_address)
    NameError: global name 'ip_address' is not defined
**********************************************************************
1 items had failures:
   3 of   3 in hget.connect_to_server
***Test Failed*** 3 failures.
eduardo@drubi:~/docencia/redes/labs/lab0$
```

Notas sobre los test

- La función que tienen que completar incluye tests, lo que les va a ayudar a detectar algunos de los errores que podrían cometer.
- La batería de tests que les damos, ha sido desarrollada para correr en el laboratorio. En particular, el test que prueba que `shiva.famaf.unc.edu.ar` no permite conexiones al puerto 80. Tengan en cuenta que si lo prueban en su casa o en otro ámbito, el test posiblemente falle debido a que internamente espera un `[Errno 111] Connection refused`y recibe un `[Errno 110] Connection timed out`.

- Si su versión de Python es la 2.6.6, no utilicen acentos (á, í, ó) en los comentarios del código porque el `doctest` se queja!!

```
eduardo@drubi:~/docencia/redes/labs/lab0$ python -m doctest hget.py
Contactando al servidor en 200.16.17.104...
Contactando al servidor en 200.16.17.187...

Traceback (most recent call last):
:
UnicodeDecodeError: 'ascii' codec can't ,decode byte 0xc3 in position 52: ordinal not in range(128)
```

Estilo de Codificación

Deben seguir el estilo de codificación PEP 8 usado en los archivos que les entregamos.

Recomendamos usar la herramienta "Python PEP 8 code style checker" `pep8` para controlar que su código siga el estilo de codificación que les pedimos.

Ejemplo de uso

```
eduardo@drubi:~/docencia/redes/labs/lab0$ pep8 hget.py
hget.py:6:80: E501 line too long (80 characters)
hget.py:21:15: E261 at least two spaces before inline comment
hget.py:24:1: E302 expected 2 blank lines, found 1
hget.py:56:33: E225 missing whitespace around operator
hget.py:67:1: W291 trailing whitespace
hget.py:85:22: E211 whitespace before '('
hget.py:236:1: W391 blank line at end of file
eduardo@drubi:~/docencia/redes/labs/original$
```

Si el estilo de codificación es correcto, la salida es nula:

```
eduardo@drubi:~/docencia/redes/labs/lab0$ pep8 hget.py
eduardo@drubi:~/docencia/redes/labs/lab0$
```

Requisitos del código a entregar

- NO se solicita un informe para este laboratorio. Pero el código que escriban debe contener *comentarios* con detalles de lo que hicieron.
- La entrega se realizará por e-mail, con fecha límite del lunes 20 de marzo a las 23:59:59. El e-mail debe ir dirigido a `profredes-famaf@googlegroups.com`, tener asunto `Lab0 Nombre Apellido`, y contener como adjunto el archivo `nombre_apellido_hget.py` modificado.
- El programa entregado debe superar los tests incluidos en `hget.py` y en `hget-test.py`.
- El programa entregado debe seguir el estilo de codificación PEP8.
- El trabajo del Lab0 es *individual*. No pueden compartir código entre alumnos, aún si después vayan a formar un grupo.
- La biblioteca de sockets de Python permite saltearse algunos pasos y omitir algunos argumentos si estamos usando los valores por defecto. No usen eso (para este Lab), así se ven los valores explícitos.

Material de Referencia

Programación con sockets

Las siguientes guías son de programación de sockets en C; pero como en realidad las primitivas de sockets usualmente se implementan al nivel del sistema operativo, son esencialmente las mismas que en Python:

- http://beej.us/guide/bgnet: Una guía de programación con sockets.
- http://www.arrakis.es/~dmrq/beej/home.htm: Una traducción al castellano del anterior, es posible que algunas páginas estén ausentes en la documentación online. La documentación

disponible para descarga se encuentra completa.
- http://www.lowtek.com/sockets/
- http://es.tldp.org/Tutoriales/PROG-SOCKETS/prog-sockets.html

Para dudas específicas sobre cada una de las operaciones de sockets, pueden consultar las páginas de manual de Linux asociadas a cada una de ellas (en C).

La documentación para la implementación de sockets en Python está dentro de lareferencia de la biblioteca estándar.

HTTP

- RFC2616: El estándar de HTTP/1.1.
- Hypertext Transfer Protocol en Wikipedia.

Python

- The Python Tutorial.
- El Tutorial de Python: traducción al castellano del anterior; disponible también en fotocopiadora. En PDF.
- The Python Standard Library: referencia de la biblioteca estándar de Python, en inglés.
- PyAr: Página de la comunidad Python en Argentina.
- Estilo de Codificación PEP 8.