PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

IIC2233 - Programación Avanzada
1^{er} semestre 2017

Actividad 14

Materia: Networking

Introducción

Te encuentras en casa sin nada que hacer cuando decides navegar por la *deep web* de Programación Avanzada. Allí te encuentras con una extraña carpeta llamada FTProgra. El nombre te parece sospechoso, así que decides usar tus conocimiento de *networking* para realizar una conexión estable entre el servidor y tu computador y copiar todos esos archivos.

Objetivos

Para esta actividad, los objetivos son:

- Conectar el archivo `server_ftp.py` al `cliente.py` que están en carpetas distintas mediante el uso de *sockets* con el fin de enviar y recibir comandos en ambos programas.
- Completar las líneas de código necesarias en ambos archivos con el fin de que los comandos enviados por el cliente sean escuchados, ejecutados y respondidos por el servidor

Instrucciones

Como el servidor es antiguo, éste siempre se mantiene en ejecución y puede escuchar solo a un computador a la vez. Al momento de realizar la conexión con otro usuario, ya no escuchará ni se conectará a otro computador hasta que el cliente envíe el comando `logout`. En ese caso el servidor vuelve a estar disponible para otra posible conexión.

Librería `os`

La librería `os` posee los métodos `isdir(path)`, `isfile(path)`, `listdir(path)`, y `sep` que permiten verificar si se trata de un directorio, de un archivo, ver que contiene una carpeta, y generar el separador de carpetas dependiendo del sistema operativo.

Comandos

Curiosamente todas las líneas de comandos perdidas corresponden a la parte de conexión con el servidor y a los comandos que requerían una respuesta del servidor. A continuación se explicará cual era la respuesta del servidor ante los comandos que lograste rescatar:

- `ls`. El servidor despliega una lista de archivos y carpetas que se encuentran en el directorio de trabajo.
- `get`: Dado el nombre de algún archivo ubicado en el directorio de trabajo del servidor y el nombre de un archivo para guardarlo, el servidor enviará ese archivo serializado y el cliente lo guardará bajo el nombre entregado. En caso que no existe dicho archivo en el servidor, se imprimirá un mensaje en consola indicando que no existe o que corresponde a una carpeta. Por otro lado, si el archivo existe en el directorio del cliente, simplemente será reemplazado.

  Ejemplo: `get server.gif nuevo_gif.gif` El servidor buscará un archivo llamado `server.gif` y en caso de encontrarlo lo enviará al cliente y este lo guardará con el nombre `nuevo_gif.gif`.
- `send`: Recibe el nombre de algún archivo ubicado en el directorio de trabajo de cliente y el nombre para guardarlo en el servidor. El cliente le enviará ese archivo al servidor para que lo guarde bajo el nombre entregado. En caso de que no exista el archivo en el directorio del cliente, se imprimirá en consola una alerta para el usuario, al igual si el nombre dado al servidor para que guarde el archivo ya existe. El servidor no tiene la autoridad para reemplazar archivos.

  Ejemplo: `send nuevo_gif.gif server_1.gif` El cliente buscará un archivo llamado `nuevo_gif.gif` y en caso de encontrarlo lo enviará al servidor y este lo guardará con el nombre `server_1.gif` siempre y cuando no esté ocupado.
- `logout`. El cliente se desconecta del servidor permitiéndole estar de nuevo disponible para otra conexión.

Requerimientos

- (2.00 pts) Conexión y protocolos
  - (1.00 pts) Conexión entre el Servidor y Cliente.
  - (0.50 pts) Protocolo para enviar datos.
  - (0.50 pts) Protocolo para recibir de datos.
- (4.00 pts) Comandos
  - (0.50 pts) `ls`
  - (1.50 pts) `get` y avisar ante algún error.
  - (1.50 pts) `send` y avisar ante algún error.
  - (0.50 pts) `logout`

Notas

- Puede usar los métodos del cliente que alcanzó a descargar para realizar las funciones en el lado del servidor.

Entrega

- **Lugar:** GIT - Carpeta: Actividades/AC14
- **Hora:** 16:55