Computación 2

Trabajo Recuperatorio

“File Server“

2020

Enunciado

Escribir un programa cliente-servidor en Python que permita enviar y recibir archivos hacia y desde el servidor.

El servidor recibirá por argumento en su ejecución el modificador **-p** o **--port** para especificar el puerto donde atenderá el servicio.

El cliente se lanzará y recibirá los siguientes modificadores:

- **-h | --host** *host* IP donde atiende el servidor.
- **-p | --port** *port* Puerto donde atiende el servicio.

Una vez conectado, el cliente le brindará al usuario una línea de comandos para enviar comandos al servidor. Los comandos disponibles serán los siguientes:

Comando	Descripción
`pwd`	Retorna el directorio actual del servidor (comando remoto)
`lpwd`	Retorna el directorio actual del cliente (comando local)
`cd /ruta/`	Cambia la ruta actual en el servidor (comando remoto)
`lcd /ruta/`	Cambia la ruta actual en el cliente (comando local)
`ls`	Lista el contenido del directorio actual en el servidor.
`lls`	Lista el contenido del directorio actual en el cliente.
`get filename`	Permite descargar un archivo remoto desde el servidor al cliente.
`put filename`	Permite subir un archivo local desde el cliente al servidor.
`help`	Breve info de los comados disponibles.
`exit`	Cierra la conexión con el servidor y termina el cliente.

Ejemplo de uso del cliente:

```
python3 file_cliente.py -h 192.168.0.123 -p 1234
> help
Comandos disponibles:
pwd                 Retorna el directorio actual del servidor (comando remoto)
lpwd                Retorna el directorio actual del cliente (comando local)
cd /ruta/           Cambia la ruta actual en el servidor (comando remoto)
lcd /ruta/          Cambia la ruta actual en el cliente (comando local)
ls                  Lista el contenido del directorio en el servidor.
Lls                 Lista el contenido del directorio en el cliente.
get filename        Permite descargar un archivo remoto desde el servidor al cliente.
put filename        Permite subir un archivo local desde el cliente al servidor.
exit                Cierra la conexión con el servidor y termina el cliente.
help                Muestra esta ayuda.
>
> pwd
200 ok
```

```
/home/mr.robot/

> cd /tmp/remoto
200 ok

> pwd
200 ok

/tmp/remoto

> lpwd
200 ok
/home/diego

> lcd /tmp/local
200 ok

> ls
200 ok

archivo.txt
otroarchivo.txt

> lls
200 ok

ejemplo_local.txt

> get archivo.txt
200 ok

> lls
200 ok

archvio.txt
ejemplo_local.txt

> put ejemplo_local.txt
200 ok

> ls
200 ok

archivo.txt
ejemplo_local.txt
otroarchivo.txt

> exit
200 ok
Conexion cerrada.
```

Especificaciones:

- La comunicación entre clientes y servidor será mediante **sockets INET STREAM** en un puerto servidor elegido por el alumno (documentado debidamente).
- Los comandos remotos deberán ser ejecutados en el servidor, y su salida será enviada al cliente.
- Los comandos locales se ejecutarán en el cliente, sin requerir ningún envío de información al servidor.
- El servidor responderá al cliente con uno de dos códigos. El cliente mostrará un OK o un ERROR según sea la respuesta del servidor.
  - **200**: Ejecución satisfactoria del comando, sigue el resultado, si lo hay.
  - **500**: Ejecución errónea del comando.
- El servidor podrá atender **conexiones de varios clientes simultáneamente**. Podrá implementar multiproceso o multihilo. Deberá justificar su elección.
- Deberá hacer un uso apropiado para los mecanismos de comunicación entre procesos.
- El código deberá publicarse en un directorio en su repositorio GIT de la materia.
- Tanto los códigos de servidor como los del cliente deben estar separados en archivos por funcionalidad (NO armar todo en un solo archivo, recordar módulos y packages python).
- El directorio del repositorio deberá contener los siguientes elementos:
  - servidor/ Directorio con el source del servidor.
  - cliente/ Directorio con el source del cliente.
  - README.md Archivo de información general del sistema. Deberá incluir:
    - Justificación de decisiones de implementación (procesos, hilos, etc).
    - Incluir instrucciones de instalación y ejecución de servidor y cliente.
    - Incluir dependencias python necesarias para la ejecución (si las hubiere).
    - Sección *TODO* con los puntos que considera que pueden mejorarse en futuras actualizaciones del software, tanto para servidor como para cliente.
- Deberán realizarse commits periódicos de avance del proyecto.
- El trabajo deberá ser presentado con fecha límite antes de la fecha límite establecida (ver campus). Realizar el tag correspondiente en tiempo y forma.
- Dudas y consultas:
  - @d1cor en Telegram.
  - diego.cordoba@um.edu.ar