Redes de Computadoras

Laboratorio - Programación con Sockets

Se debe implementar en Python una aplicación que mediante *sockets* permita enviar y recibir mensajes y archivos a otros pares, es decir, la aplicación en todo momento puede recibir *mensajes* y *archivos*, y también puede enviarlos a otra aplicación, para esto el programa debe bifurcarse, y una parte de este debe encargarse de recibir y otra de transmitir. Los *mensajes* tendrán un largo máximo definido.

```
MAX_LARGO_MENSAJE = 255
```

El programa será invocado de la siguiente manera:

```
python mensajeria.py port ipAuth portAuth
```

Levantará el argumento *port*, que será el puerto donde oirá el *receptor*, y escribirá el *emisor* y los argumentos *ipAuth* y *portAuth* que indicarán la dirección del servidor de autenticación.

El programa constará de dos grandes partes, una *emisora* y otra *receptora*.

- ### Receptora:

Esta oirá continuamente por un *puerto* que se le pasará por argumento al programa, al llegar los *mensajes* a ese *puerto*, los imprimirá en pantalla (*salida estándar*), junto con la *fecha y hora*, y el *nombre de usuario* del emisor, los mensajes serán *strings* que tendrán el *ip* del emisor, seguido de un espacio seguido del *nombre de usuario*, seguido de un espacio, seguido de "`dice: `" y el *mensaje*, estos mensajes, como ya se dijo, tendrán un largo máximo.

Ejemplos de impresión del receptor:

```
[2025.06.23 17:02] 172.23.20.23 nwirth dice: Feliz Cumple!!!!!
```

Al llegar archivos, los guardará en el directorio "*actual*" del programa en el *FileSystem* local.

Ejemplos de impresión del receptor:

```
[2025.06.23 17:05] 172.23.10.40 <Recibido ./foto.jpg de nwirth>
[2025.06.23 17:05] 172.23.10.40 <Error Recibiendo Archivo de nwirth>
```

- ### Emisora:

Esta enviará mensajes a un determinado *receptor* al puerto pasado como parámetro, levantará de la entrada estándar el destino del mensaje (*ip o nombre de host*) y el contenido, por ejemplo:

```
172.23.20.23 Feliz Cumple!!!!!
tecnoinf215.esi.edu.uy Feliz Cumple!!!!!
tecnoinf215 Feliz Cumple!!!!!
pcunix71.fing.edu.uy Feliz Cumple!!!!!
```

Si en lugar de poner un *ip o nombre de host* se pone un "`*`" se debe enviar el mensaje a todos los receptores presentes en la red del emisor.

```
* Gracias a todos los que se acordaron
```

Cuando un mensaje es enviado a todos todos (*Broadcast*) el *receptor* también debe imprimirlo (como con el resto de los mensajes).

Si en lugar de un mensaje se pone un "`&file`" y un "`path`" se debe enviar el archivo que está en *path* en el FS local.

Ejemplo de impresión del emisor:

```
* &file ./foto.jpg
172.23.20.25 &file ./foto.jpg
pcunix71.fing.edu.uy &file ./foto.jpg
```

Ambas partes del programa escribirán en la salida estándar, por lo que en pantalla aparecerán mezcladas las salidas del *emisor* y del *receptor*.

- ## Autenticación

Los usuarios del programa deben autenticarse, para ello se dispondrá de un ejecutable *redes-auth* este será invocado de la siguiente manera:

```
./redes-auth port
```

Este programa autenticará sobre un archivo de texto con el siguiente formato, cada línea será de la forma:

```
usuario-contraseña_condificada_en_md5;Nombres_Apellidos;
```

por ejemplo *(la contraseña es el nombre de usuario en md5)*:

```
aturing-50bc36a72e099d0ae1e78186a0859d46;Alan_Mathison_Turing;
alovelace-fd3b63b26dfd701c68eef561f464691e;Augusta_Ada_Byron;
jkonvalina-5fc2f043528bd2998123667d58f7a065;John_Konvalina;
jvneumann-941a833c2db9668cd995c74a3c8f3985;Neumann_Janos_Lajos;
nwirth-95f845845deacdf77cf85be58b5d744a;Niklaus_Wirth;
cbabbage-b6fe35296e44d8d2955ef7f609f90904;Charles_Babbage;
edijkstra-cc38c2a1e714e20bc9ffa725614e810a;Edsger_Wybe_Dijkstra;
```

El servidor al conectarse un cliente le responderá:

```
Redes 2025 - Laboratorio - Autenticación de Usuarios
```

Los clientes enviarán lo siguiente:

```
usuario-contraseña_condificada_en_md5
```

El servidor de autenticación responderá en caso afirmativo

```
SI
Nombres_Apellidos
```

Y en caso negativo:

```
NO
```

Los fines de línea enviados (y recibidos) por el servidor se codifican CRLF (\r\n)

- ## ¿IP o Nombre de Host?

Los usuarios pueden optar por enviar los mensajes usando el *ip* del destino, o en su defecto el *nombre del host* destino. Se debe tener en cuenta que los paquetes van con el *ip*, por lo que en el caso que se escriba el *nombre de host*, este se debe resolver previo a mandar el mensaje.

Ejemplo de sesión del servidor de autenticación:

```
> Redes 2025 - Laboratorio - Autenticación de Usuarios
< alovelace-bd5f3cbebf74ce4f5126e4536fdacb80
> SI
> Augusta_Ada_Byron
```

(*los "<" y ">" no forman parte de la salida, indican si son mensajes entrantes o salientes*)

Ejemplo de sesión del programa de mensajería:

```
 ./mensajeria 22764 172.23.1.240 3456
Usuario: aturing
Clave: aturing
Bienvenido Alan_Mathison_Turing
[2025.06.23 17:02] 172.23.20.23 nwirth dice: Feliz Cumple!!!!!
 172.23.20.23 Gracias!!!!!
[2025.06.23 17:03] 172.23.20.23 nwirth dice: Merece!!!!!
172.23.1.25 No me tenés que decir nada hoy??
[2025.06.23 17:05] 172.23.1.25 edijkstra dice: Por?
172.23.1.25 Es mi cumple!!!
[2025.06.23 17:06] 172.23.1.25 edijkstra dice: ahh que boludo!!
[2025.06.23 17:06] 172.23.1.25 edijkstra dice: me pensaba que era en mayo!!!
[2025.06.23 17:07] 172.23.1.25 edijkstra dice: Feliz Cumple!!!
* Gracias a todos los que se acordaron
[2025.06.23 17:08] 172.23.1.10 aturing dice: Gracias a todos los que se acordaron
CTRL + C Recibido.... Cerrando Sesión

./mensajeria 22764 172.23.1.240 3456
Usuario: aturing
Clave:
Bienvenido Alan_Mathison_Turing
172.23.20.23 &file ./foto.jpg
172.23.20.23 Te llegó el archivo???
[2025.06.23 18:06] 172.23.20.23 nwirth dice: Si, impecable!!!!!
 CTRL + C Recibido.... Cerrando Sesión
```

El programa deberá atender las señales y alarmas del sistema, y se terminará al recibir un CTRL + C o recibir un KILL o TERM. Al recibir estas señales deberá cerrar de manera correcta los recursos que haya solicitado, *sockets* memoria, etc.

El comportamiento de entradas y salidas debe ser el que figura en los ejemplos.

Sobre la Entrega:

- **El trabajo es obligatorio.**
- **Se pueden obtener hasta 20 puntos.**
- **El trabajo se realizará en grupos de hasta 4 personas.**
- **Se deben inscribir los grupos enviando un mail a mzabalza@fing.edu.uy indicando los integrantes datos (CI, Nombre y Apellido). Plazo hasta el *lunes 19 de mayo inclusive*. Quienes no se inscriban se considerará que desisten de hacer el obligatorio con la consecuente pérdida del curso.**
- **Se deberá entregar:**
  - `mensajeria.py`

Este contendrá la solución al problema y un comentario con los nombres y números de documento de todos los integrantes del grupo.

  - `comentarios.txt`

Contendrá comentarios sobre la solución, errores, etc.

- **Estos archivos deberán venir comprimidos en uno que se llamará:**
  - `redes-lab.tar.gz`
- **Hay tiempo hasta el día domingo 22 de junio a las 23:59 UYT.**
- **Se dispondrá de un entregador en el Campus Virtual.**
- **Se coordinará una pequeña defensa con todos los integrantes del grupo.**

- ***Lecturas Recomendadas:***
  - ***Bifurcación:***
- https://docs.python.org/es/3.8/library/threading.html

  - ***Programación Básica con Sockets:***
- https://docs.python.org/es/3/howto/sockets.html
- https://docs.python.org/3/library/socket.html
- https://realpython.com/python-sockets/

  - ***Señales:***
- https://old.chuidiang.org/clinux/senhales/senhales.php
- https://docs.python.org/es/3/library/signal.html