Xarxes II
Escola Universitària Politècnica
Departament d'Informàtica

---

1. Objetivo de la práctica

El objetivo de la primera práctica es familiarizarse con la programación en sockets. El alumno deberá implementar un sencillo protocolo de chat. En líneas generales, cada usuario (a través de su cliente) deberá *registrarse* en un servidor y podrá enviar un mensaje a todos los usuarios *conectados* o a uno en particular.

El cliente se encargará de recoger las peticiones del usuario e interaccionar con el servidor y otros clientes.

A continuación, se describe la interacción del usuario con el cliente, el protocolo de comunicación entre cliente-servidor y cliente-cliente, y los mensajes permitidos en el protocolo.

2. Acciones del usuario

Una vez iniciado el cliente, se entrará en una fase de configuración en la que se pedirán el identificador de usuario a utilizar en la fase de registro (ver sección 3), la dirección del servidor y el puerto UDP por le que se recibirán los mensajes privados.

Ejemplo:

```
\$ ./cliente
cliente> esperando un identificador de usuario:
cliente> pepito    # respuesta del usuario
cliente> esperando la direccion de un servidor:
cliente> localhost # respuesta del usuario
cliente> esperando puerto UDP para recepcion mensajes privados:
cliente> 7790
cliente> Conexion, identificacion y registro, serv:‘‘localhost’’, usuario:‘‘pepito’’...
```

Una vez finalizada la conexión, identificación y registro en el servidor, el cliente espera del usuario sentencias del siguiente tipo:

```
cliente> cadena1: cadena2
```

El campo *cadena2* será el texto a transmistir, si es necesario.
El campo *cadena1* determinará la acción a realizar, y tendrá los siguientes valores:

1. la palabra clave "todos":
   El usuario desea enviar *cadena2* a todos los usuarios conectados al servidor.

   Ejemplo:

   ```
   cliente> todos: Hola, alguien para un privado?
   ```

   El mensaje "Hola, alguien para un privado?" deberá ser enviado al servidor para que lo difunda entre los clientes actualmente conectados.

2. el identificador de un usuario:

   El usuario desea enviar *cadena2* sólo al usuario *cadena1*. Para satisfacer esta orden, el cliente deberá consultar en el servidor la dirección del usuario *cadena1* y enviar *cadena2* directamente al cliente de *cadena1* (ver sección 4).

   Ejemplo:

```
cliente> manuela: de donde eres?
```

El mensaje "de donde eres?" debe ser enviado únicamente al usuario "manuela".

3. la palabra clave "ayuda":

   Muestra al usuario las opciones disponibles. Observación: el campo *cadena2* es irrelevante.

   Ejemplo:

```
cliente> ayuda:
```

4. la palabra clave "salir":

   El usuario desea terminar el servicio. Observación: el campo *cadena2* es irrelevante.

   Ejemplo:

```
cliente> salir:
```

Cualquier otra combinación será descartada, y se indicará al usuario que ha cometido un error.

3. Protocolo cliente-servidor

A continuación se enumeraran los posibles estados e interacciones del cliente y el servidor.

1. El servidor espera conexiones en el puerto TCP/8642.
2. El cliente se encontrará consecutivamente en los siguientes estados, se describen también los estados del servidor para el cliente en cuestión.

   a) Estado de configuración (ver sección 2).

   b) Estado de conexión:
   El cliente se conecta al servidor en el puerto TCP/8642 y el servidor entra en espera del mensaje de *identificación*. Si se da cualquier error el cliente termina la sesión.

   c) Estado de identificación:
   1) El cliente envia un mensaje de *identificacion* y espera el mensaje *OK* del servidor.
   2) El servidor envia mensaje *OK* y entra en espera del mensaje *registro*.

   d) Estado de registro:
   1) El cliente envia un mensaje de *registro*.
   2) El servidor da de alta al usuario, (guardando el identificador de usuario, la dirección IP desde la que se conecta y el puerto UDP por el que recibirá los mensajes privados), envia un mensaje *OK* y entra en un bucle de espera de peticiones.

   Registrarse con un identificador de usuario presente actualmente en el sistema se considera en error. Cuando se acaba la conexión con un cliente por el motivo que sea, se le da de baja en el registro.

   e) Estado de peticiones al servidor:
   1) mensaje *pregunta*, el cliente demanda la dirección IP de un usuario y el puerto UDP por el que recibe los mensajes privados. El servidor responde con un mensaje *respuesta*. Si el usuario no existe el servidor enviará como dirección IP "null" y puerto UDP 0. Nota: el cliente iniciará por su cuenta una comunicación con el cliente en el puerto UDP y dirección IP obtenidas (ver sección 4).
   2) mensaje *difusion*, se demanda al servidor que envie un mensaje a todos los clientes conectados actualmente. El servidor envia a todos los usuarios conectados excepto a la fuente el mensaje *difusion*, y envia un mensaje de *OK* a la fuente una vez satisfecha la petición.

3) mensaje *salir*, se informa al servidor del abandono de la sesión.

Observaciones generales:

a) Si se incumple el protocolo por parte del cliente, ya sea poque el mensaje enviado es sintácticamente o semánticamente incorrecto el servidor envia un mensaje de *ERROR* al cliente, y termina la sesión cone el cliente.

b) Si se incumple el protocolo por parte del servidor, ya sea poque el mensaje enviado es sintácticamente o semánticamente incorrecto el cliente envia un mensaje de *SALIR* al servidor, y el cliente termina la sesión.

c) La recepción de un mensaje de *ERROR* obliga al cliente a terminar la sesión.

d) El envio de un mensaje *SALIR* por parte del cliente obliga al cliente a terminar la sesión.

e) Si el cliente detecta que el servidor ha caido, termina la sesión.

f) Si el servidor detecta que el cliente ha caido o bien recibe un mensaje *SALIR*, termina la sesión con el cliente, esto implica dar de baja al usuario registrado.

g) Cualquier otra situación no contempalada en este protocolo tanto en el servidor como en el cliente conduce a terminar la sesión ya sea en el servidor o en cliente.

h) No hay timeouts.

4. Protocolo cliente-cliente

Como hemos indicado previamente el usuario puede enviar un mensaje privado a un usuario tras obtener su dirección IP y el puerto UDP, a través del servidor. Una vez obtenida la dirección, se envia un mensaje *privado*.

Por lo tanto, los clientes pueden recibir mensajes tanto por la conexión establecida con el servidor como por el puerto UDP que hayan indicado el usuario.

Todos estos mensajes se mostrarán al usuario y se especificará la fuente.

Observación:

Cualquier recepción por el puerto UDP que no corresponda a un mensaje *privado* se descartará automáticamente.

5. Mensajes

A continuación se detalla la sintaxis de los mensajes que se utilizan en nuestro protocolo.

- (mensaje *OK*):

  ```
  100
  ```

  La cadena “100”.

- (mensaje *ERROR*):

  ```
  200
  ```

  La cadena “200”.

- (mensaje *identificacion*):

  ```
  HOLA
  ```

  La cadena “HOLA”.

- (mensaje *registro*):

```
300 idUsuario puertoUDP
```

La cadena “300” seguida de un espacio, el identificador del usuario, un espacio y el puerto UDP.

- (mensaje *pregunta*):

```
400 idUsuario
```

La cadena “400” seguida de un espacio y el identificador de usuario.

- (mensaje *respuesta*):

```
500 direccionIP puertoUDP
```

La cadena “500” seguida de un espacio, la dirección IP, un espacio y el puerto UDP .

- (mensaje *privado*):

```
600 texto
```

La cadena “600” seguida de un espacio, y un texto.

- (mensaje *difusion*):

```
700 texto
```

La cadena “700” seguida de un espacio, y un texto.

- (mensaje *salir*):

```
800
```

La cadena “800”.

Observar que no se ha especificado como se delimitan los mensajes. Consultar el siguiente enlace y actuar en consecuencia:

http://en.wikipedia.org/wiki/Newline

6. Entrega

La práctica se puede realizar en grupos de hasta tres personas. El material a entregar es:

1. Códigos fuente y makefiles.
2. Un documento que explique como compilar y ejecutar vuestros programas.
3. Un documento que explique las decisiones de diseño fundamentales en vuestra solución, y en caso de existir, nuevas funcionalidades que hayais incorporado.

7. Evaluación

Se valorará la correción, robustez y estilo de programación.

Como parte del proceso de evaluación se os hará entrega de la práctica de otro grupo. Básicamente, debereis analizar la interacción de vuestros servidores y clientes con los del otro grupo e identificar fallos tanto en su solución como en la vuestra, en caso de existir. Se abrirá una actividad en sakai para que podais entregar un documento de no más de 2 páginas que recoja vuestras observaciones.