Práctica final

Criterio de puntuación:

Las prácticas 3 y 4 tienen un valor conjunto de 1 punto.

2 opciones para la última práctica:

Opción 1: Servidor de chat compartido 1,5 puntos.
Guión págs. 3 y 4 de este documento.

Opción 2: Selección de una práctica de la **LISTA** 2 puntos.
Práctica de realización libre.
Siempre cumpliendo las especificaciones

LISTA de posibles prácticas en la opción 2:

1. Hacer el servidor de chat compartido de la "opción 1", empleando la función poll( ).
2. Hacer un programa que envíe un fichero indicado por smtp a una dirección determinada por ejemplo con la sintaxis: **envia fichero servidordecorreo direccion**
3. Hacer un programa que cada x minutos ejecute un comando de un fichero y lo envíe por correo: comando comandfile x servidordecorreo dirección
4. Hacer un programa que cada vez que se conecte un usuario al ordenador envíe un correo con el nombre del usuario a una dirección de correo
5. Hacer un programa que chequea el mail de una cuenta cada x minutos y cuando tengas mas de 10 avise.
6. Hacer un programa que recoja el correo de una cuenta por POP (sin borrarlo) y guarde todas las direcciones de correo que vea.
7. Hacer un programa que descargue de una cuenta de correo el mensaje de mayor tamaño.
8. Hacer un programa que se baje un url cada x minutos y avise cuando cambia.
9. Hacer un programa que se baje un url y calcule la velocidad del enlace.
10. Hacer un programa que sirva una página web y evalúa la velocidad del enlace.
11. Hacer un programa que obtenga x peers de un tracker de bittorrent indicado.
12. Hacer un programa que sea capaz de hacer una petición de DNS construyendo sus paquetes UDP.
13. Hacer un programa que se sincronice con un servidor de tiempo y muestre el reloj en tiempo real: **sincroniza servidor [-t | -d]**, debe soportar usar un servidor de time (-t) y de daytime (-d) y tiene que ser UDP
14. Hacer un programa servidor UDP que envía el reloj a quien se lo pida en un paquete UDP + cliente UDP que utilice eso para llevar el reloj.
15. Hacer un programa servidor que lea un fichero de configuración con una lista de comandos y se le pueda indicar con paquetes UDP que ejecute cada uno.

16. Hacer un programa servidor que lea un fichero de configuración con una lista de comandos a elegir y los realice y mande los resultados cuando se le pidan accediendo al url: http://servidor/comando/<numero>
17. Hacer un programa servidor de chat UDP al que se le envían paquetes y el los envía a todos los que le han enviado un paquete hace menos de x tiempo.
18. Hacer un programa que escanee puertos, en una dirección ip, desde la línea de comandos: scan <ip> <puerto1> <puerto2> <puerto3> <puerto4> <puerto5>.
19. Hacer un programa que escanee puertos, en una dirección ip, leyendo en un fichero: scan <fichero>.

De modo que las 2 posibles combinaciones son:

Prácticas 3 y 4 + Opción 1 = 2,5 puntos

O bien

Prácticas 3 y 4 + Opción 2 = 3 puntos

El valor total de las prácticas es de 3 puntos. Podrán optar a alcanzar la máxima puntuación, sólo aquellos que elijan la "opción 2". La mayor puntuación de la opción 2, se corresponde con el grado de dificultad de las prácticas que se proponen.

En cualquier caso, tanto si se elige la opción 1 como la opción 2, la realización de esta última práctica, será por parejas. Se ha creado una wiki, para que os vayáis apuntando.

¡IMPORTANTE! A la hora de apuntarse en la wiki, debéis tener en cuenta que de la LISTA de la Opción2, cada práctica la podrán realizar como máximo 2 parejas.

Entrega de la práctica

Para entregar la práctica, tanto si se ha optado por la opción 1, como por la opción2, se debe crear igualmente un directorio `prac5` dentro de su directorio home (`/opt3/rc/rc<nºcuentacomun>/prac5`).

En el directorio `prac5` debe dejar las fuentes necesarias para compilar el programa de la práctica realizada, con todas las funciones que haya programado. Para la corrección se borrarán los programas ejecutables y se recompilarán de forma automática por lo que en el directorio debe existir un fichero `Makefile` que permita construir todos los programas lanzando el comando `make` sin argumentos en el directorio `prac5`.

Los programas deben cumplir exactamente las especificaciones y se probarán en la medida de lo posible mediante scripts de corrección que esperarán que los programas se comporten tal y como se pide. Asegúrese antes de entregar los programas que cumplen las especificaciones.

Al finalizar el curso se cerrarán las cuentas y se considerará entregada la práctica que haya en el directorio `prac5`.

Servidor de chat compartido

Objetivos

El objetivo de esta práctica es familiarizarse con el manejo de más de dos sockets simultáneamente y la construcción de un protocolo de nivel de aplicación de una aplicación de ejemplo, realizando un servidor de chat compatible con el cliente de la práctica anterior.

Servidor central de chat con varios clientes simultáneos

Realice un programa que permita conectar a un número determinado de clientes usando el programa de la práctica anterior para charlar entre ellos de forma que todos lean lo que escriben los demás. Las especificaciones serán las siguientes.

FORMATO

chatserver [-p <puerto>] [-n <numero de clientes>]

DESCRIPCION

El programa debe quedarse indefinidamente escuchando en el puerto especificado y permitirá mantener hasta **n** conexiones simultáneas de clientes.

Cada cliente puede enviar mensajes que son retransmitidos a todos los demás clientes que estén conectados.

Todos los usuarios pueden escuchar los mensajes de los demás pero sólo los que se han asignado un nombre pueden hablar. Cada cliente puede asignarse un nombre con el comando **\nombre** y una vez que lo haya hecho, aparecerá en las listas de usuarios y podrá enviar mensajes y recibir mensajes privados.

El servidor, al retransmitir los mensajes de un usuario al resto de usuarios, añadirá el nombre del usuario que ha escrito dicho mensaje, para que se sepa quién está hablando.

Además se considerará que si un cliente envía una línea que empieza con el carácter ****, la línea no es una frase para retransmitir, sino que es un comando que debe ser interpretado por el servidor. Puede añadir los comandos que considere útiles, pero estos son los que deben estar:

\nombre <nick>
El usuario elige su nombre o lo cambia

\lista
El servidor envía a ese cliente una lista de todos los usuarios conectados que se han asignado nombres y además dice cuántos usuarios invisibles (que no se han asignado nombre todavía, pero están escuchando) hay en el sistema.

\m <usuario> Mensaje...
Envía el mensaje sólo al usuario de nombre especificado.
En el caso de que un cliente cierre la conexión, por ejemplo haciendo **^C**, el servidor debe darse cuenta y reutilizar su sitio para atender a otros posibles clientes. Si un cliente intenta conectarse cuando todos los sitios están ocupados se le debe notificar con un mensaje.

EJEMPLOS DE USO

Para lanzar el servidor en el puerto 6789 permitiendo hablar a 4 usuarios:

En **10.1.1.26**]% chatserver -p 6789 -n 4

Mientras el servidor chatserver está funcionando se puede usar con el programa chat de la práctica anterior. Por ejemplo veamos varias conexiones (con > marcamos lo que escribe el usuario)

En **10.1.1.27**	En **10.1.1.28**
% chat -c 10.1.1.26 6789	% chat -c 10.1.1.26 6789
Conectado al servidor de chat	Conectado al servidor de Chat
>¿Hay alguien ahí?	
Elige un nombre antes de hablar	
>\nombre A	
>¿Hay alguien ahí?	
A: ¿Hay alguien ahí?	A: ¿Hay alguien ahí?
	>Hola
	Elige un nombre antes de hablar
>\lista	
Usuarios conectados: A y 1 invisible.	
	>\nombre B
Nuevo usuario visible B	
B: Hola	>Hola B: Hola
>\lista	
Usuarios conectados:	
A, B y 0 invisibles	