Práctica 1. Sockets

Arquitectura y Tecnología de Computadores. Sistemas Distribuídos

1. Ejercicios a realizar

Los ejercicios a realizar consisten básicamente en los propuestos durante la sesión 1 de prácticas, con alguna variación. Se detallan a continuación.

1.1. Cliente para el servicio *time*

Escribir un cliente que se conecte a un servidor del protocolo hora (RFC 868) e imprima en pantalla una versión textual de la fecha y la hora devueltas por el servidor. El cliente debe admitir el nombre del servidor desde la línea de comandos, de la siguiente manera:

```
$ cliente_hora sirio.edv.uniovi.es
```

Probarlo con los servidores disponibles en `sirio.edv.uniovi.es` y en `hora.uniovi.es`

1.2. Servidor para el protocolo *echo-reverso*, sin concurrencia

Escribir un servidor para una variante del protocolo *echo* (RFC 862), que consiste en que el servidor devuelve cada línea que recibe escrita hacia atrás. El servidor debe admitir en línea de comandos el número de puerto en que escuchará, o terminar con un error si no se especifica. Un ejemplo de invocación sería:

```
$ echo_reverso 7890
Esperando clientes en el puerto 7890...
```

Observar que el servidor, a diferencia del *echo* descrito en RFC 862, no devuelve los datos tan pronto los recibe, sino que debe esperar hasta recibir una línea completa (esto es, un retorno de carro), acumulando lo que va recibiendo hasta entonces[1]. Cuando tiene la línea completa, le "da la vuelta" y envía el resultado al cliente.

Para probarlo usa telnet. Deberías ver algo como lo siguiente (suponiendo que el servidor escucha en el puerto 7890 de sirio):

```
$ telnet sirio.edv.uniovi.es 7890
Hola
aloH
Parece que funciona
anoicnuf euq eceraP
```

El servidor puede asumir que el cliente nunca enviará líneas de más de 256 caracteres. En todo caso, si el cliente enviara más, se comportará como si el caracter 256 hubiera sido un retorno de carro.

[1] Sugerencia: quizás la forma más simple de hacerlo sea que el servidor lea un caracter de cada vez

2. Aspectos formales

Debes crear una carpeta llamada `p1` y dentro de ella dos carpetas llamadas *e1* y *e2*, para cada uno de los ejercicios anteriores. Cada carpeta debe contener el código fuente (con los nombres de ficheros que detallaremos seguidamente) y alguna forma automatizada de compilación, ya sea en forma de *script* o de `Makefile`. El código fuente debe ser C estándar y debe compilar y funcionar al menos en las dos máquinas de prácticas `sirio` y `orion`.

Los nombres de los ficheros fuente serán los siguientes:

Ejercicio 1 `cliente_hora.c`

Ejercicio 2 `echo_reverso.c`

2.1. Entrega

Cuando el código haya sido probado, se borrarán los ejecutables, dejando sólo el código fuente y el *script* o `Makefile` de compilación, y se empaquetarán todas las carpetas en un único archivo `tar` comprimido. Para esto, debes situarte en la carpeta "padre" de la carpeta `p1` y desde ella teclear:

```
$ tar czvf p1.tar.gz p1
```

El resultado será el fichero `p1.tar.gz`, que seguidamente deberás firmar digitalmente con el comando[2]:

```
$ gpg --sign p1.tar.gz
```

El resultado será un archivo `p1.tar.gz.pgp` que entregarás a través del formulario Web en la página de la asignatura. Si lo deseas, además de firmarlo puedes cifrarlo para tu profesor de prácticas, mediante el comando:

```
$ gpg --sign --encrypt -r jldiaz@uniovi.es -r ariasjr@uniovi.es p1.tar.gz
```

[2]Deberás efectuar esta operación en la máquina en la que hayas generado tu par de claves.