UNIVERSIDADE DA CORUÑA
Departamento de Tecnoloxías da Información
e as Comunicacións

LABORATORIO DE GESTIÓN DE REDES (I)

1. PRESENTACIÓN

El laboratorio de Gestión de Redes constará de un conjunto de prácticas obligatorias. Estas prácticas se realizarán en el laboratorio 0.3. La evaluación de las prácticas será individual y constará de:

- Examen oral individual en el laboratorio, realizando una demostración de la ejecución de las prácticas y explicando el código implementado.
- Examen en Junio con el resto de teoría de la asignatura.

A continuación se detalla el programa correspondiente a las prácticas de este bloque. Las 2 primeras prácticas se indican a modo de guía para la puesta en práctica de los conceptos teóricos de programación de Sockets en C y no es necesario su entrega y defensa.

Práctica 1: Realización de un cliente de un servicio de eco mediante TCP.
Práctica 2: Realización de un servidor de un servicio de eco mediante TCP.
Práctica 3: Realización de un juego en red orientado a conexión.
Práctica 4: El agente Net-SNMP.
Práctica 5: Creación de Módulos en Net-SNMP.
Práctica 6: Extensión del agente Net-SNMP para gestionar la aplicación creada en la práctica 3.

Esta documentación incluye material introductorio sobre los sistemas informáticos a utilizar en el laboratorio y la descripción de las tres primeras prácticas. El enunciado del resto de prácticas se detallarán en otros documentos que serán entregados a lo largo del curso.

2. COMPILACIÓN DE PROGRAMAS

Por convención el nombre del archivo fuente que contiene el programa debe terminar con ".c" por ejemplo: `miPrograma.c`. El contenido del archivo deberá obedecer la sintaxis de C.Existen muchos compiladores de C. El compilador GNU de C es *gcc*, y está disponible en varias plataformas. Para este laboratorio se va a utilizar la implementación del compilador C, gcc-4.0 (GNU project C and C++ Compiler), del sistema operativo Linux Ubuntu 6.06 (http://www.ubuntu.com).

Para compilar el programa usaremos el comando `gcc`. El comando deberá estar seguido por el nombre del programa en C que se quiere compilar. Es posible indicar diferentes opciones al compilador. El comando básico de compilación es:

```
gcc miPrograma.c
```

dónde `miPrograma.c` es el nombre del archivo que contiene la aplicación escrita en C. Si la compilación ha sido exitosa, se genera un archivo llamado a.out. Es recomendable utilizar la opción `–o` del compilador para especificar el nombre del programa compilado, como se muestra a continuación:

```
gcc -o miPrograma miPrograma.c
```

Para más información sobre el compilador, utilizar las páginas en línea del manual del sistema (*man gcc*).

3. FICHEROS DE CABECERAS

La funcionalidad de C se obtiene a través de librerías de funciones. Una librería es un conjunto de recursos (algoritmos) prefabricados que puede utilizar el programador para realizar determinadas operaciones. Son trozos de código que contienen alguna funcionalidad preconstruida que puede ser utilizada por un ejecutable.

Existen librerías estándares, como las de entrada salida. Las declaraciones de las funciones utilizadas en estas librerías, junto con algunas macros y constantes predefinidas que facilitan su utilización, se agrupan en ficheros de nombres conocidos que suelen encontrarse en sitios predefinidos.

Para utilizar una función de una librería en una aplicación, es necesario incluir en el código fuente los prototipos de dichas funciones, que están definidos en ficheros con extensión ".h", denominados ficheros de cabeceras. La directiva del preprocesador `#include` instruye al compilador para incluir el archivo fuente indicado, en el sitio indicado del código fuente que lo referencia, y de esta forma compilar los dos conjuntamente. El archivo fuente indicado en la directiva #include se debe encerrar entre comillas dobles o paréntesis de ángulo. Por ejemplo:

```
#include <archivo>
#include "archivo"
```

Cuando se indica `<archivo>` se le dice al compilador que localice los archivos donde están los archivos incluidos o "include" del sistema. Usualmente los sistemas UNIX/Linux guardan los archivos en el directorio `/usr/include`. Si se usa la forma "`archivo`", entonces se busca en el directorio actual, es decir, donde se está ejecutando el programa.

Es posible indicar una lista de directorios en los cuales se buscarán los archivos de cabecera `#include` con nombres relativos (es decir, los que no empiezan con diagonal /), utilizando la opción `-I` del compilador. Por lo tanto, si se quiere incluir archivos de cabecera guardados en `/usr/include/cab_386` se tendrá que hacer:

```
gcc miPrograma.c -I /usr/include/cab_386 -o miPrograma
```

Para la realización de las prácticas primeras 3 prácticas será necesario utilizar, al menos las siguientes librerías:

<*sys/types.h*>	Contiene las definiciones de tipos del estándar POSIX.
<*sys/socket.h*>	Definiciones de tipos para el uso de sockets y prototipos de las funciones necesarias para su manejo.
<*netinet/in.h*>	Constantes definidas para sistemas Internet.
<*arpa/inet.h*>	Rutinas de translación de direcciones IP.

4. EVALUACIÓN DE PRÁCTICAS

Las prácticas deberán finalizarse la semana del 19 al 23 de Mayo. El código fuente de la práctica deberá ser copiado al repositorio de entrega de prácticas durante esa semana. En caso contrario, se considerará que la práctica NO ha sido presentada.

La defensa de las prácticas se realizará durante las dos últimas semanas de clase (del 26 de mayo al 8 de junio) en el horario habitual de prácticas del alumno, es decir, al horario en el que el alumno se haya apuntado al inicio de las prácticas. En caso de hacerlo en el horario de otro grupo, sin autorización previa del profesor de prácticas de la asignatura, se podrá considerar que la práctica ha sido entregada tarde.

Para la defensa de cada práctica, cada alumno deberá compilar el código en el laboratorio y mostrar su funcionamiento. El profesor podrá solicitar la explicación de alguna parte específica del código implementado o la ejecución de algún escenario de prueba en concreto

5. PRÁCTICA 1 (Opcional): REALIZACIÓN DE UN CLIENTE DE UN SERVICIO DE ECO MEDIANTE TCP

5.1. Objetivos

Familiarización con aplicaciones elementales de red y conocimiento de las librerías de sockets, para servicios orientados a conexión.

5.2. Requisitos

Lectura de la documentación aportada para la práctica.

5.3. Enunciado

Implementación de un cliente que solicite un servicio muy sencillo por medio de TCP (orientado a conexión). El cliente objetivo de esta primera práctica recibirá como parámetros la dirección IP de la máquina en donde se encuentre el servidor, un número de puerto de TCP y un mensaje. Para esta primera práctica se utilizará el servidor eco de UNIX, que se encuentra en el puerto 7 de todas las maquinas. Este servidor se encuentra esperando en el puerto TCP número 7 la llegada de peticiones de clientes en forma de mensajes TCP, previamente establecida la conexión, a los que contestará con el mismo mensaje que se le ha enviado por medio del cliente.

5.4. Realización

Se pide implementar el cliente de eco. Para dicha implementación se sugieren unos pasos a realizar que se dan a continuación, aunque estos pasos no son obligatorios siempre que se llegue a la consecución del cliente pedido.

5.5. Esqueleto del programa

```
/*****************************************************************************************/
/*                                                                                       */
/*     PRÁCTICAS DE GESTIÓN DE REDES                                                     */
/*                                                                                       */
/*     Práctica  Construcción de un cliente de un servicio TCP.                          */
/*                                                                                       */
/*     Programa: clienteTCP.c                                                            */
/*     Autor:                                                                            */
/*     Fecha:                                                                            */
/*                                                                                       */
/*****************************************************************************************/

/* #include (librerías de sockets necesarias y librerías estándar ) */

/* Función principal del cliente */
        /* Comprueba que clienteTCP tiene 4 argumentos (clienteTCP, dirección, puerto,
                mensaje) */
        /* Creación del socket TCP */

        /*****************************************************************************************/
        /* Preparar un nombre local en un puerto cualquiera: El nombre local       */
        /* se prepara con la propia dirección de Internet y el puerto se deja a    */
        /* la elección de la máquina.                                              */
        /*****************************************************************************************/

        /* Asigna nombre local al socket: Asignación de una dirección local        */

        /*****************************************************************************************/
        /* Preparar el nombre de la máquina remota: La dirección remota hay        */
        /* que convertirla a binario con la función inet.addr y el número de       */
        /* puerto remoto hay que ponerlo en formato de red con htons.              */
        /*****************************************************************************************/

        /* Establecer la conexión con el servidor */

        /* Enviar el mensaje */

        /* Recibir la respuesta */

        /* Cerrar el socket */
```

6. PRÁCTICA 2 (Opcional): REALIZACIÓN DE UN SERVIDOR DE UN SERVICIO DE ECO MEDIANTE TCP

6.1. Objetivos

Familiarización con aplicaciones elementales de red y conocimiento de las librerías de sockets, para servicios orientados a conexión.

6.2. Requisitos

Lectura de la documentación aportada para la práctica y haber realizado la práctica 3.

6.3. Enunciado

Implementación de un servidor que implemente un servicio muy sencillo por medio de mensajes TCP. El servidor objetivo de esta primera práctica recibirá como parámetros el número de puerto TCP en donde se pondrá a la espera de mensajes de múltiples clientes, a los que contestará con el mismo mensaje recibido, reenviando una copia del mensaje que ha recibido.

6.4. Realización

Se pide implementar el servidor de eco. Para dicha implementación se sugieren unos pasos a realizar que se dan a continuación, aunque estos pasos no son obligatorios siempre que se llegue a la consecución del servidor pedido.

6.5. Esqueleto del programa

```
/*****************************************************************************************/
/*                                                                                       */
/*     PRÁCTICAS DE GESTIÓN DE REDES                                                     */
/*                                                                                       */
/*     Práctica: Construcción de un servidor de un servicio TCP.                         */
/*                                                                                       */
/*     Programa: servidorTCP.c                                                           */
/*     Autor:                                                                            */
/*     Fecha:                                                                            */
/*                                                                                       */
/*****************************************************************************************/

/* #include (librerías de sockets necesarias y librerías estándar ) */

/* Función principal del servidor */
        /* Comprueba que servidorTCP tiene 2 argumentos (servidorTCP, puerto)*/
        /* Creación del socket TCP */

        /*****************************************************************************************/
        /* Preparar un nombre local en el puerto especificado: El nombre local     */
        /* se prepara con la propia dirección de Internet que la sabe el sistema,  */
        /* y el puerto se obtiene del parámetro recibido                           */
        /*****************************************************************************************/

        /* Asigna nombre local al socket: Asignación de una dirección local        */

        /* Esperar el establecimiento de alguna conexión */

                /* Recibir el mensaje  */

                /* Enviar la copia del mensaje (el '\0' incluido) */

        /* Cerrar el socket */
```

7. PRÁCTICA 3: REALIZACIÓN DE UN JUEGO EN RED ORIENTADO A CONEXIÓN

7.1. Objetivos

Diseñar e implementar un protocolo simple a nivel de aplicación utilizando un servicio orientado a conexión.

7.2. Requisitos

Lectura de la documentación aportada para la práctica. Se recomienda haber realizado las prácticas anteriores referentes a servicios orientados a conexión.

7.3. Enunciado

Se pide implementar el siguiente juego en red, utilizando para ello las primitivas del protocolo orientado a conexión, TCP.

El juego consta de dos elementos: un servidor y un cliente. El servidor, al inicio del juego, generará un número al azar (por ejemplo, entre 0 y 100). En el caso del cliente, se encargará de enviar mediante mensajes los números tecleados por el usuario al servidor, con el objetivo de adivinar el número del servidor.

Los mensajes a enviar tanto por el servidor como por el cliente estarán formados básicamente por dos campos de tipo entero:

- **Operación**: campo que indica el tipo de operación que realiza ese mensaje (enviar un número, enviar una respuesta, etc.).
- **Dato**: campo que se utilizará para el envío de los datos asociados al mensaje.

Por ejemplo, cuando el cliente envíe un número podrá utilizar el código de operación 1 y en el campo dato introducir el número a enviar; en el caso de que el servidor envíe la respuesta, se indicará con el código de operación 2 y en el campo dato incluirá el valor de igual, menor o mayor.

En el mensaje de respuesta del servidor se podrán recibir tres valores:

- **0**: El número enviado coincide con el número generado en el servidor.
- **-1**: El número enviado es menor que el número generado por el servidor.
- **+1**: El número enviado es mayor que el número generado por el servidor.
- **-2**: Se ha alcanzado el número máximo de intentos permitido por el servidor.

Un servidor soportará una partida simultáneamente, cada partida consta únicamente de dos jugadores: el servidor y el cliente. El inicio de una partida estará marcado por el envío de un número por parte de un cliente, y una partida finalizará en el mismo momento en que el cliente acierte el número o exceda un número máximo de intentos.

El servidor será configurable a través de un fichero de texto en el que se podrán configurar las siguientes propiedades:

- El puerto que ocupará el servidor (a través de una propiedad denominada *port*).
- El número máximo de intentos (a través de una propiedad denominada *maxAttempts*).

A continuación se muestra el contenido de un fichero de configuración del servidor de ejemplo:

```
port 1234
maxAttempts 10
```

El cliente recibirá como único argumento, el número de puerto del servidor al que conectarse.

7.4. Realización

Se pide implementar el cliente y el servidor.