ENTREGA 3
ENVÍO Y RECEPCIÓN DE MENSAJES CON TRAMAS DE DATOS

Prácticas de Fundamentos de Redes y Comunicaciones – Curso 2019/2020

María del Mar Ávila Vegas (mmavila@unex.es)

24 de febrero de 2020

Índice

1. Construcción de tramas de datos.

- Las tramas de datos estarán compuestas por los siguientes campos:

```
struct TramaDatos
 {
   unsigned char S;  → Sincronismo = SYN =22;
   unsigned char D;  → Direccion='T';
   unsigned char C;  → Control = STX = 02;
   unsigned char N;  →NumTrama = (En principio fijo a '0');
   unsigned char L;  →Long (Longitud del campo de datos);
   char Datos[255];  → Datos[255];
   unsigned char BCE;  → (Siempre debe tomar siempre valores entre 1 y 254);
 };
```

22	'T'	02	'0'	Long	Datos	BCE

- En un principio la trama de datos se construirá de la manera expresada anteriormente. En las sesiones sucesivas se explicará como irá cambiando.

2. Cálculo del BCE:

- Para calcular el valor del campo BCE de una trama se debe hacer la operación lógica XOR dos a dos entre todos los caracteres del campo de datos de la trama. Es decir se hará (DATOS[0] XOR DATOS[1]) XOR DATOS[2]) XOR DATOS[3]) ... XOR DATOS[Long-1]). Por ejemplo, para calcular el BCE de una trama de datos que transporta los caracteres "HOLA" se haría:

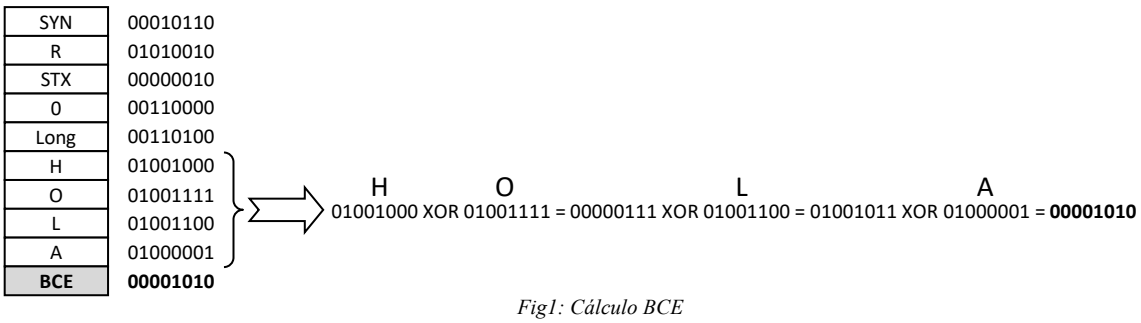

Fig1: Cálculo BCE

- Si al calcular el **BCE** obtenemos un valor ***0*** o un valor ***255***, lo cambiaremos siempre por ***1***.

3. Envío de mensajes con tramas de datos.

- Los mensajes de datos serán como máximo de 800 caracteres.

- Los mensajes se trocearán en bloques de 254 caracteres (ya que la última posición la reservaremos para el '\0'). El bloque de datos se almacenará en el campo ***Datos*** de la trama de datos. El campo ***long*** es el carácter que resulta de expresar la longitud total del campo de datos (valor decimal de un carácter comprendido entre 1 y 254). Se calculará el BCE para cada bloque de datos y se almacenará en el campo ***BCE***. Para cada bloque se enviará una trama de datos completa.

- Se enviarán tantas tramas de datos como número de bloques de datos se hayan extraído del mensaje.

- Al final de la última trama de datos se almacenará un salto de línea ('\n'), de esta forma se imprimirá un salto de línea al final de cada mensaje en el receptor. (Si ya se hizo en la primera entrega con F1 al enviar el mensaje, debe mantenerse).

- ***El envío y la recepción nunca deben excluirse mutuamente.***

4. Recepción de mensajes con tramas de datos.

- La trama de datos se **procesará** solo cuando se reciba completa. En caso de no hacerse así, la práctica será NO APTA.

- *Procesar la trama de datos* consiste en, una vez recibida la trama completa, calcular el BCE del campo Datos de la trama recibida y compararlo con el campo BCE de dicha trama; si tienen el mismo valor, la trama será correcta y, por lo tanto, habrá que mostrar por pantalla el campo **Datos**. Si, por el contrario, no tienen el mismo valor, habrá que mostrar un mensaje por pantalla informando que se ha producido un error

(**"Error al recibir la trama"**) y esa trama se descartaría, por tanto, en este caso no habría que mostrar el campo **Datos**. Si no se hiciera de esta forma, la práctica sería **NO APTA**.

- En ningún momento deberán mostrarse los campos de la trama que no se pidan expresamente.

```
carR = RecibirCaracter(PuertoCOM)
Si (carR != 0) //Comprobamos si hay datos que recibir
    Inicio
        Caso de (campo) //Campo es una variable utilizada para indicar el campo que se va recibiendo
            1://Carácter de sincronismo
                Si (caracter == 22)
                    Inicializar_Trama (TDR);
                    Almacenarlo en TDR
                    Campo++
                    Break
                Else                    //Estas dos líneas se eliminarán ya del código, ya que el
                    printf("%c',carR)   //mensaje se recibe en forma de tramas de datos
            2://Carácter de dirección
                Almacenarlo en TDR
                Campo++
                Break
            3://Carácter de control
                Almacenarlo en TDR
                Campo++
                Break
            4://Número de trama
                Almacenarlo en TDR
                Si (Carácter_control es distinto de 02, es decir, que no es trama de datos)
                    Campo = 1 //Esta trama es de control y el siguiente campo será de una trama nueva
                    Procesar_Trama_Control (TDR)
                Si no (Sería una trama de datos)
                    Campo++  //Seguiremos recibiendo el resto de campos de una trama de datos
                Break
            5://Longitud
                Almacenarlo en TDR
                Campo++;
                ¡Cuidado, no ponemos Break para que entre en el siguiente campo!
            6://Datos
                //Recibimos los datos completos de una vez y los almacenamos en el campo datos de la
                //trama
                RecibirCadena (PuertoCOM, Campo_Datos, Campo_Longitud);
                Campo_Datos[Campo_Longitud] = '\0'; //Añadimos el final de cadena al campo datos
                                                    //de la trama
                Campo++
                Break
            7://BCE
                Almacenarlo en TDR
                Campo = 1
                Procesar_Trama_Datos (TDR)
                Break
    Fin
```

5. Utilización de diferentes colores para diferenciar mensajes enviados y recibidos.

- Para diferenciar los mensajes que se envíen de los que se reciban, es **obligatorio** hacer uso de colores. Para ello deberemos utilizar Windows.h.

- Tendremos que declarar un manejador (Pantalla) para controlar el color.

```
HANDLE Pantalla  //HANDLE está declarado en Windows.h
```

- Mediante la función `GetStdHandle()` se podrá manipular el dispositivo estándar de salida para mostrar el color.

```
Pantalla = GetStdHandle(STD_OUTPUT_HANDLE)
```

- Tendremos que usar una variable color que contendrá el valor con el cual indicaremos el color de texto y color de fondo deseado.

```
int color
```

- El color se calculará a través de la siguiente fórmula:
  color de texto + color de fondo*16

- Los colores que se podrán utilizar son 15 y se identifican con los siguientes valores enteros:

0 = Negro
1 = Azul
2 = Verde
3 = Azul verdoso
4 = Marrón
5 = Morado
6 = Verde mostaza
7 = Gris claro
8 = Gris oscuro
9 = Azul eléctrico
10 = Verde fluorescente
11 = Cian
12 = Rojo
13 = Fucsia
14 = Amarillo
15 = Blanco

Si se quisiera imprimir texto rojo (12) con fondo blanco (15), la variable `color` tendría que almacenar el siguiente valor:

```
color = 12 + 15*16   //(Valdría 252).
```

Cuando la variable `color` tome el valor de 252, se mostrará esa combinación.

- Ejemplo de código para uso de color: *Muestra de un mensaje en gris claro con fondo azul verdoso y de otro mensaje en azul celeste con fondo negro.*

```
#include <iostream>
#include <conio.h>
#include <windows.h>

using namespace std;

int main()
{
  HANDLE  Pantalla;
  int color;

  Pantalla = GetStdHandle (STD_OUTPUT_HANDLE);
  color = 7 + 3*16;  //Letras en gris claro y fondo azul verdoso. (El fondo hay que
                     //multiplicarlo por 16).
  SetConsoleTextAttribute (Pantalla, color);
  printf ("Prueba de color 1\n");
  color = 11 + 0*16; //Letras en azul celeste y fondo negro. (El fondo hay que
                     //multiplicarlo por 16).
  SetConsoleTextAttribute (Pantalla, color);
  printf ("Prueba de color 2\n");
  getch();
}
```

- Se deberán usar colores que no impidan una visión cómoda de la información.

6. Control de fin de aplicación (ESC).

En todo momento, el usuario puede poner fin a la aplicación mediante la pulsación de la tecla **ESCAPE**. En el caso de estar realizando alguna acción mientras se pulsa esta tecla, se cancelará dicha acción.

7. Entrega de la sesión práctica.

A través de la herramienta AVUEx (exclusivamente), según las instrucciones dadas en clase a este respecto, debe entregarse un archivo comprimido en formato ZIP o RAR que contenga lo siguiente:

- Un archivo AUTORES.TXT, que incluya nombre y apellidos, por este orden, de los autores de la práctica, así como el grupo al que pertenecen ambos.
- Los archivos fuente de la práctica. **Cada fichero fuente** debe incluir el nombre, apellidos y grupo de los autores de la práctica.
- El fichero ejecutable (compilado a partir de los ficheros fuentes entregados) de la práctica.
- Debe incorporarse documentación interna adecuada y suficiente como para seguir adecuadamente el código.
- La fecha tope para subir esta ***entrega 3*** será el día antes de dentro de dos sesiones correspondiente a cada grupo de alumnos. **Cualquier práctica entregada con posterioridad a la fecha y hora indicada se considerará no válida a todos los efectos**.