Teoría de Códigos. Curso 2006-2007

Prácticas: **1ª Parte**

1. Se trata de elaborar un programa **`sumabi`** que realice la suma binaria de dos archivos. El programa se invocará con el siguiente formato:

`sumabi [n] <filein_1> <filein_2> <fileout>`

Han de tenerse en cuenta los siguientes aspectos:

Los nombres de los archivos que hay que *sumar* (<filein_1> y <filein_2>) y del archivo donde se guardará el resultado (<fileout>), se indicarán como parámetros en la línea de comandos al invocar el programa para su ejecución.

Los tres archivos están en formato ASCII, la suma se efectuará carácter a carácter, teniendo en cuenta que el carácter '0' representa el bit 0 y que el carácter '1' representa el bit 1. En este caso se ignorarán los espacios en blanco, tabulaciones y saltos de línea que aparezcan en los ficheros originales; pero si aparece cualquier otro carácter se interrumpirá la ejecución del programa visualizando el mensaje de error: "**`sumabi: ERROR: Carácter no admitido en archivo ASCII <file>: <c>`**", dónde <file> es el nombre del archivo de entrada donde se ha encontrado el carácter ilegal, y <c> es el carácter en cuestión.

El archivo resultado ha de tener exactamente la misma longitud que el primero de los dos archivos que se *suman*. Si el segundo archivo fuese más corto que el primero, el resultado de la *suma* sería el mismo que se obtendría si se completase el segundo archivo con ceros hasta alcanzar la longitud del primero. Si el segundo archivo fuese más largo que el primero, el resultado de la suma sería el mismo que se obtendría si se truncase (por el final) el segundo archivo para que tuviese la misma longitud que el primero.

Si alguno de los archivos que hay que *sumar* no existiese, el programa terminará inmediatamente mostrando el siguiente mensaje de error: "**`sumabi: ERROR: No existe el archivo <file>`**", donde <file> es el nombre del archivo que no existe.

Si ya existiese un archivo con el nombre del archivo resultado, simplemente se sobrescribirá.

Dicho de otra manera, el archivo resultado ha de ser exactamente igual que el primer archivo que se *suma*, salvo en aquellas posiciones en que en el segundo archivo haya un 1, en cuyo caso el valor del archivo resultado será el complementario del que haya en el primero.

El valor de **n** (un número entero entre 1 y 255, el valor por defecto será 80) indica que el archivo de salida se escribirá dividido en líneas de **n** caracteres y sin espacios en blanco ni tabulaciones. Si **n** es 0, la salida se escribirá en una sola línea.

Si se omite el nombre del archivo de salida (<fileout>), el resultado se escribirá por la salida estándar.

2. Elaborar un programa **`numbit`**, que cuente los bits con un determinado valor, dentro de un archivo. El formato con que se utilizará este comando es

`numbit [0 | 1] <file>`

El primer parámetro indica el valor de los bits que hay que contar.

El último parámetro corresponde al nombre del archivo que queremos analizar.
Sobre los caracteres del archivo **<file>** se hacen las mismas consideraciones que en el programa anterior.

El resultado se mostrará por pantalla en una única línea con el siguiente formato:
<xxx> **de** <yyy> **(**<pp.pppp> **%)** donde:

a) <xxx> será el número de bits con el valor indicado que aparecen en el archivo e <yyy> será el número total de bits. Ambos valores se escribirán sin ceros a la izquierda. (***Nota: para este apartado en concreto, se considerará que el archivo a procesar nunca tendrá más de 100 MB***)
b) <pp.pppp> será el porcentaje de *bits del valor indicados* que hay en el archivo, indicado siempre con dos cifra enteras y cuatro decimales .

Si el archivo no existiera se mostrará una única línea con el siguiente mensaje: **numbit: ERROR: no existe el archivo <file>** donde <file> será el nombre del archivo.

3. Se trata de elaborar un programa **mmult** que calcule el producto en binario de dos matrices

mmult [| -u | -e] <matriz_1> <matriz_2> <matriz_res>

Han de tenerse en cuenta los siguientes aspectos:

Cada matriz binaria estará descrita en un archivo de texto que tendrá siempre la misma estructura: Las tres primeras líneas tendrán tres valores enteros *n, k, d* ($0 < n, k, d < 100$); *n* es el número de columnas de la matriz, *k* es el número de filas de la matriz, para este programa el significado de *d* es irrelevante. Las siguientes *k* líneas contendrán cada una de ellas una secuencia de *n* caracteres ('0' o '1'), correspondiendo a las *k* filas de la matriz.

El primer parámetro indica como se debe realizar el producto: **-u** indica que el producto se realizará según la definición usual del producto de matrices (es la opción por defecto); **-e** indica que el producto se realizará multiplicando filas por filas; es decir, sería el producto usual de matrices de la primera matriz por la traspuesta de la segunda matriz. En ambos casos, si las matrices no tienen los tamaños apropiados para multiplicarse según la opción deseada, el programa únicamente emitirá el siguiente mensaje: "**mmult: ERROR: tamaño de matrices inapropiados**".

El fichero resultado será el resultado de multiplicar la matiz_1 por la matriz_2 según la opción elegida y se escribirá en el mismo formato que los ficheros originales escribiendo el valor 1 para *d*.

Si no existiese algún fichero origen, el programa únicamente emitirá el siguiente mensaje: "**mmult: ERROR: archivo no encontrado**".

Si ya existiese un archivo con el mismo nombre que el archivo resultado, se sobrescribirá.

Si se omite el nombre del archivo de salida, el resultado se escribirá en la salida estándar.