Teoría de Códigos. Curso 2008-2009

Prácticas. 1ª Parte. Fecha de entrega 5 de noviembre.

A. 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:

1. 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. Se puede omitir el nombre del archivo de salida (<fileout>), en ese caso el resultado se escribirá por la salida estándar.

2. Los ficheros filein_1, filein_2 y fileout son ficheros de texto donde el carácter "0" representa el bit 0 y el carácter "1" representa el bit 1. Los espacios en blanco, tabulaciones y saltos de línea que aparezcan en los ficheros originales se ignoraran; pero si aparece cualquier otro carácter se interrumpirá la ejecución del programa visualizando el mensaje de error: "**`sumabi: ERROR: Carácter no admitido en archivo <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. Estas características son extensivas al resto de programas.

3. 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.

4. 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.

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

6. Para obtener cada bit del archivo resultado sólo se han de tener en cuenta los bits que ocupan la misma posición en los dos archivos que se *suman* (0 y 0 ó 1 y 1 darán 0; 0 y 1 ó 1 y 0 darán 1); es decir, *esta suma se realiza sin acarreos*. Dicho de otra manera, el archivo resultado ha de ser exactamente igual que el primer archivo que se *suma*, salvo en aquellas posiciones (bits) 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.

7. El parámetro n es opcional, siendo un numero entero entre 0 y 255. Si el valor de n es nulo indica que el archivo de salida se escribirá en una sola línea; en otro caso se escribirá dividido en líneas de n caracteres. En ambos casos, sin espacios en blanco ni tabulaciones. El valor de n por omisión es 80.

8. Se debe entregar el fichero fuente, el fichero ejecutable y las instrucciones para su complilación. Esto es extensible a todos los programas.

B. Se trata de elaborar un programa **`numbit`**, que cuente los bits con un determinado valor, dentro de un archivo. El formato con que se utilizará este programa es

`numbit [0|1] <file>`

Han de tenerse en cuenta los siguientes aspectos:

1. El primer parámetro indica el valor de los bits que hay que contar. El segundo parámetro corresponde al nombre del archivo que queremos analizar y tiene las mismas características que los archivos del apartado anterior.

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

3. 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.

4. Se debe entregar el fichero fuente, el fichero ejecutable y las instrucciones para su complilación.

Teoría de Códigos. Curso 2008-2009

Prácticas. 2ª Parte. Fecha de entrega 15 de diciembre.

Se trata de elaborar un programa **`codifica`** que codifique un archivo ascii utilizando un código lineal binario. El programa se invocará con el siguiente formato:

`codifica [ | -p | -f] <codigo> <filein> <fileout>`

Han de tenerse en cuenta los siguientes aspectos:

1. El código lineal binario que se utilizará en la codificación estará descrito en un archivo de texto que tendrá siempre la misma estructura: Las tres primeras líneas del archivo contendrán, respectivamente, la longitud (n), la dimensión (k) y la distancia mínima (d) del código. ($0 < n, k, d < 100$). 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 generadora del código. Por ejemplo, la descripción de un código de longitud 15, dimensión 5 y distancia mínima 7, podría tener el siguiente aspecto:

```
15
5
7
111011001010000
011101100101000
001110110010100
000111011001010
000011101100101
```

2. El nombre de este archivo se le indicará al programa como primer parámetro al invocarlo desde la línea de comando.

3. El nombre del archivo que hay que codificar y el que deberá tener el archivo que contenga el resultado de la codificación se indicarán, respectivamente, como segundo y tercer parámetros en la línea de comando al invocar el programa.

4. Si el número de bits del archivo a codificar no fuese múltiplo de la dimensión del código, se completará (sólo a efectos de su codificación) añadiendo al final el número necesario de ceros.

5. Si no existiese el fichero con la descripción del código o si su estructura no fuese correcta, el programa se limitará a emitir el siguiente mensaje: "CODIFICA: ERROR: código incorrecto".

6. El fichero a codificar no podrá ser modificado en modo alguno por el programa. Si no existiese (o no se pudiese acceder a él), el programa se limitará a emitir el siguiente mensaje: "CODIFICA: ERROR: no se encuentra el archivo a codificar".

7. Si ya existiese un archivo con el mismo nombre que el archivo resultado, el programa lo sobrescribirá con el resultado de la codificación.

8. Si se invoca el programa con la opción **`-p`** , antes del nombre del archivo que contiene la descripción del código, se entenderá que el archivo a codificar debe procesarse "en modo palabras": en ese caso, el archivo codificado se grabará con un salto de línea después de cada "palabra código". Si se invoca al programa con la opción **`-f`** (la opción por defecto) las palabras código se grabarán una detrás de otra sin separación entre ellas.

9. Se debe entregar el fichero fuente, el fichero ejecutable y las instrucciones para su complilación.

Teoría de Códigos. Curso 2008-2009

Prácticas. 3ª Parte. Fecha de entrega 20 de enero.

A. Se trata de elaborar un programa **`decod`** que decodifique (corrigiendo errores) un archivo codificado con un *código lineal binario* dado. El programa se invocará con el siguiente formato:

`Decod <p> <n> <k> <filein> <fileout>`

Han de tenerse en cuenta los siguientes aspectos:

1. El primer parámetro es obligatorio, tiene cuatro posibles valores que indica el código utilizado para codificar el archivo. 1 significa que se utilizó un código Hamming (puede ser truncado), 2 que se utilizó un código Hamming ampliado (nuevamente puede ser truncado), 3 un código Reed-Muller y 4 un código de Golay.

2. El segundo y tercer parámetro son obligatorios e indica, respectivamente, la longitud y dimensión del código utilizado.

3. Los dos últimos argumentos pasados al programa desde la línea de comando indicarán, respectivamente, el nombre del archivo codificado y el que deberá asignarse al resultado de la decodificación.

4. El archivo codificado tendrá el mismo formato que los generados por el programa **`codifica`** y puede contener *errores de transmisión*. Si el número de bits del archivo a decodificar no fuese múltiplo de la longitud del código, se completará (sólo a efectos de su decodificación) añadiendo al final el número necesario de ceros.

5. Si el programa no pudiese acceder al archivo codificado, se interrumpirá la ejecución y se mostrará el mensaje "**`DECOD: ERROR: archivo codificado inexistente o inaccesible`**".

6. Si existiese algún error cuya corrección es *dudosa*, se indicará al final de la ejecución con un mensaje de la forma "**`DECOD: <n> posibles errores`**", donde <n> será el número de *correcciones dudosas* que hubo que realizar. En esos casos, se debe decodificar la palabra recibida aunque posiblemente de forma incorrecta.

7. Recuerda entregar el fichero fuente, el fichero ejecutable y las instrucciones para su complilación. El valor de esta parte de la práctica es el doble del valor de cada una de las partes anteriores.