Práctica 1
Compresor Huffman

Universidad de Zaragoza

Mario Arcega 679192
Pablo Hernández 616923

Decisiones y Funcionamiento

La primera decisión que se tomó fue en que lenguaje realizar la práctica. El lenguaje elegido fue C, debido a tres factores, la velocidad de ejecución que tienen los programas escritos en C, la facilidad de C para trabajar con bits, y la facilidad que se tiene con C para manejar llamadas al sistema.

A la hora de programar se realizó de una forma muy modular para mejorar la lectura del código. Todos los ficheros .c tienen una cabecera con los métodos publicos.

Compresión

A la hora de comprimir se llama a la función 'comprimir()' situada en el fichero compactador.c. En nuestra solución hemos considerado que los caracteres coinciden con los carácteres de C de 8 bits. Así pues la primera tarea que realiza nuestro algoritmo es recorrer el fichero comprobando la frecuencia de aparición de cada palabra utilizando para ello una tabla indexada con los propios carácteres.

Una vez se tienen las frecuencias se pasa a crear un montículo y a poblarlo. Se crea directamente un monticulo de arboles para compatibilizar las operaciones, de este modo primero se insertan arboles que son solo raices. Para la realización del montículo nos basamos en uno encontrado en http://www.thelearningpoint.net/computer-science/data-structures-heaps-with-c-program-source-code http://www.thelearningpoint.net/computer-science/data-structures-heaps-with-c-program-source-code del cual se extrajeron las operaciones básicas y se adaptó al uso que se le iba a dar en la práctica.

Tras obtener el montículo se escribe en el nuevo fichero el numero de elementos del montículo y el propio montículo para poder recuperarlo al descomprimir, así como el numero de carácteres del

fichero original para evitar problemas de bits sueltos. Antes de pasar a escribir se forma el árbol sintactico a partir del montículo, y de él para obtener mayor velocidad al comprimir se extraen en otra tabla indexada por carácter, los códigos comprimidos de cada carácter.

Por último se pasa a comprimir el fichero. Para mejorar las velocidades se utilizan dos bufferes de 256 carácteres uno para leer y otro para escribir, de este modo se realizan menos llamadas al sistema.

Descompresión

A la hora de descomprimir lo primero que se hace es extraer el tamaño del montículo y el del fichero original. Con el tamaño del montículo se recompone y se recalcula el árbol. Con el arbol se pasa a leer del fichero y se comienza a recorrer el árbol y recomponer los carácteres. Hay una decisión de diseño que se reconoce a primera vista y es que, para intentar mejorar la velocidad se ha usado un tipo de dato de C que es la unión, la cual, facilita la lectura de bits. Así pues no utilizamos un bucle para recorrer los bits de un caracter, evitando de ese modo sobre cargas de bucles y permitiendo que el compilador pueda utilizar ejecución predicada lo que aumenta la velocidad enormemente.

Pruebas

Para las pruebas se van a utilizar tres tipos de ficheros, un fichero de texto plano llamado LoremIpsum, una imagen llamada FordGt y el propio pdf de la práctica. Las pruebas se han realizado en un ordenador propio con un procesador intel i5 y el sistema operativo MacOSx. El programa se puede compilar y ejecutar en cualquier sistema linux con compilador gcc. Todos los ficheros se encuentran en la carpeta *Pruebas*.

- Tamaños:

Nombre	Tam. Original	Tam. Comprimido
LoremIpsum	291915	156126
FordGT.jpg	393089	393873
practica1.pdf	31768	31695

- Tiempos de ejecución (segundos):

Nombre	Compresion	Descompresion
LoremIpsum	0.04	0.02

FordGT.jpg	0.03	0.04
practica1.pdf	0.01	0.02

Como se puede ver los tiempos de ejecución están relacionados con el tamaño de los códigos, a más tamaño más tiempo. Otra cosa evidente que se ve es que la imagen no se ha comprimido. Esto puede deberse a que la distribución de los carácteres es tan uniforme que no se comprime nada, pero como al fichero se le añaden datos extras se hace más grande.

Conclusión

Ha sido un trabajo que nos ha servido para comprender como funcionan los algoritmos de compresión, así como la importancia que tienen pequeños cambios en el código que pueden hacer que un programa vaya más o menos rápido. Además dado que ambos compañeros utilizamos C muy a menudo también nos ha servido para ganar confianza y velocidad a la hora de utilizar este lenguaje.