Universidad de Murcia

Facultad de Informática

Universidad Politécnica de Cartagena

Escuela Técnica Superior de Ingeniería de Telecomunicación

TÍTULO DE GRADO EN
CIENCIA E INGENIERÍA DE DATOS

Fundamentos de Computadores

Práctica 3: Representación de la información

Boletines de prácticas

CURSO 2022 / 23

Índice general

Boletines de prácticas

B3.1. Boletín 1: Decodificación de ficheros con Okteta (I)

B3.1.1. Objetivos

El objetivo de esta primera sesión de prácticas es conseguir que el alumno se familiarice con los sistemas de codificación más usuales para representar datos en un ordenador. Haciendo uso de la herramienta de Linux denominada *okteta*, el alumno podrá examinar la codificación de distintos tipos de datos y experimentar con ciertos conceptos básicos como el direccionamiento o desplazamiento dentro de un fichero. Esta sesión también se utilizará para reforzar conceptos que se imparten durante las clases teóricas. Para ello se solicitará también a los alumnos que creen mediante okteta ficheros binarios que contengan la codificación correcta de valores enteros, reales y de cadenas de caracteres.

B3.1.2. Plan de trabajo

El plan de trabajo de esta sesión será el siguiente:

1. Lectura del boletín por parte del alumno.
2. Presentación del programa y análisis de un ejemplo.
3. Realización de forma individual de los ejercicios propuestos en el boletín (con supervisión del profesor).

B3.1.3. La herramienta okteta

Okteta es un editor bastante sencillo de archivos binarios. Los datos de los archivos se muestran en dos columnas, una con los valores numéricos de cada byte y otra con los caracteres asociados a dichos valores numéricos. El programa permite editar los valores de los bytes tanto en la columna binaria como en la de los caracteres. Además de esto, otra funcionalidad que utilizaremos a lo largo de este boletín de prácticas es la decodificación de los datos binarios atendiendo a los tipos de datos más comunes, como son los enteros de distinta longitud y los números en coma flotante.

Quizá la mejor forma de familiarizarse con la aplicación sea echando un vistazo a su interfaz, como se muestra en la Figura I.1. Podemos ver que una vez que se ha abierto un determinado fichero (en este caso denominado a233.pdf), se nos empieza a mostrar el contenido del mismo en dos columnas distintas:

- Contenido plano. Se nos muestra el valor de cada byte del fichero en hexadecimal. Esto nos permite conocer cuál es el valor de los bytes almacenados en el fichero sin ningún tipo de interpretación, simplemente su contenido binario.
- Interpretación de caracteres. En esta columna se muestra cuál sería la interpretación de cada byte del fichero si se considerara que en realidad representan caracteres codificados mediante un esquema de codificación que es configurable.

Para saber en qué posición concreta se encuetra cada byte dentro del fichero, hay una columna de direcciones. Los valores que ahí se muestran se corresponden con la posición dentro del fichero del primer byte de cada fila. Para saber la dirección exacta de un determinado byte en el fichero basta con situar el cursor sobre el byte, puesto que se nos mostrará en la parte inferior izquierda de la ventana la dirección en hexadecimal de dicho byte.

Por último, una de las zonas más importantes de la interfaz es la tabla de decodificación. Una vez situado el cursor sobre un determinado byte, okteta realiza una interpretación de dicho byte y de los bytes siguientes como si estuvieran codificando distintos tipos de datos. Por ejemplo, podríamos ver cuál es el valor numérico que representaría

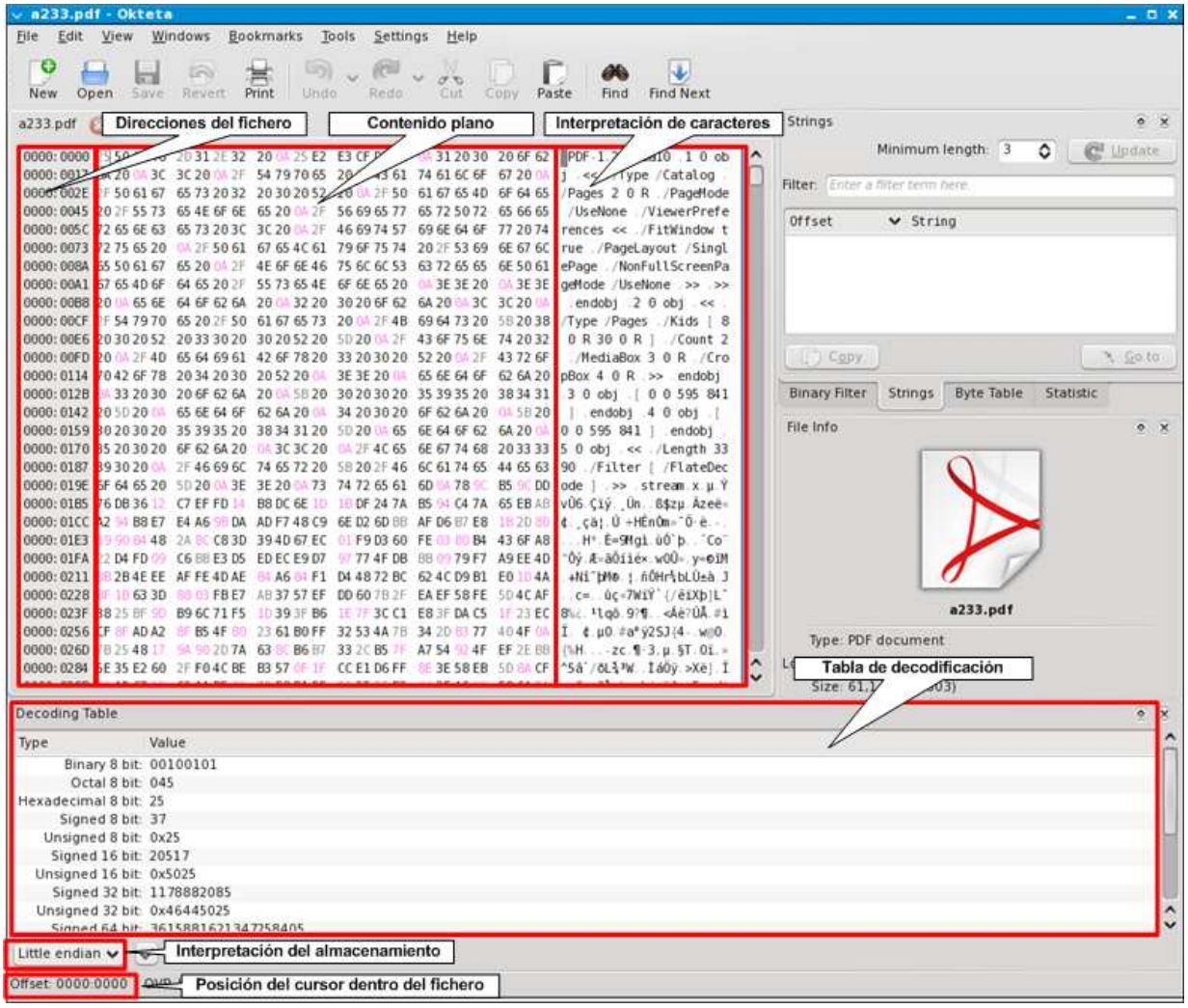

Figura I.1: Interfaz de la herramienta okteta.

dicho byte si se considerara como un entero de 8 bits con signo. En el caso de que el fichero contenga, por ejemplo, un valor numérico representado en coma flotante de simple precisión, okteta interpreta tanto el byte actual como los 3 siguientes como una codificación IEEE-754 y nos indica el número que está representado.

Para casos en los que es necesario más de un byte para representar la información, es importante recalcar que hay dos formas distintas de almacenar la información. El formato *big endian* almacena el byte menos significativo al final, por tanto ocupará una dirección más alta que el resto de bytes de la codificación. Por el contrario, el formato *little endian* almacena el byte menos significativo al principio, lo que implica que ocupará la primera dirección dentro del conjunto de bytes codificados. Es importante saber con qué formato se han almacenado los datos para que okteta pueda decodificarlos correctamente.

B3.1.4. Ejemplo de intepretación de datos con okteta

Vamos a analizar ahora los contenidos concretos de un fichero binario, llamado **salidac.bin**[1], que almacena distintos tipos de datos. Mediante el seguimiento de este ejemplo se pretende que el alumno se familiarice con el uso de la herramienta y refuerce sus conocimientos en lo que respecta a las distintas formas de codificar la información. La Figura I.2 nos muestra el contenido del fichero en cuestión.

El fichero contiene 22 bytes que contienen los siguientes datos:

[1] **Nota:** Este fichero, y los siguientes que vamos a usar, los tenéis disponibles en el Aula Virtual.

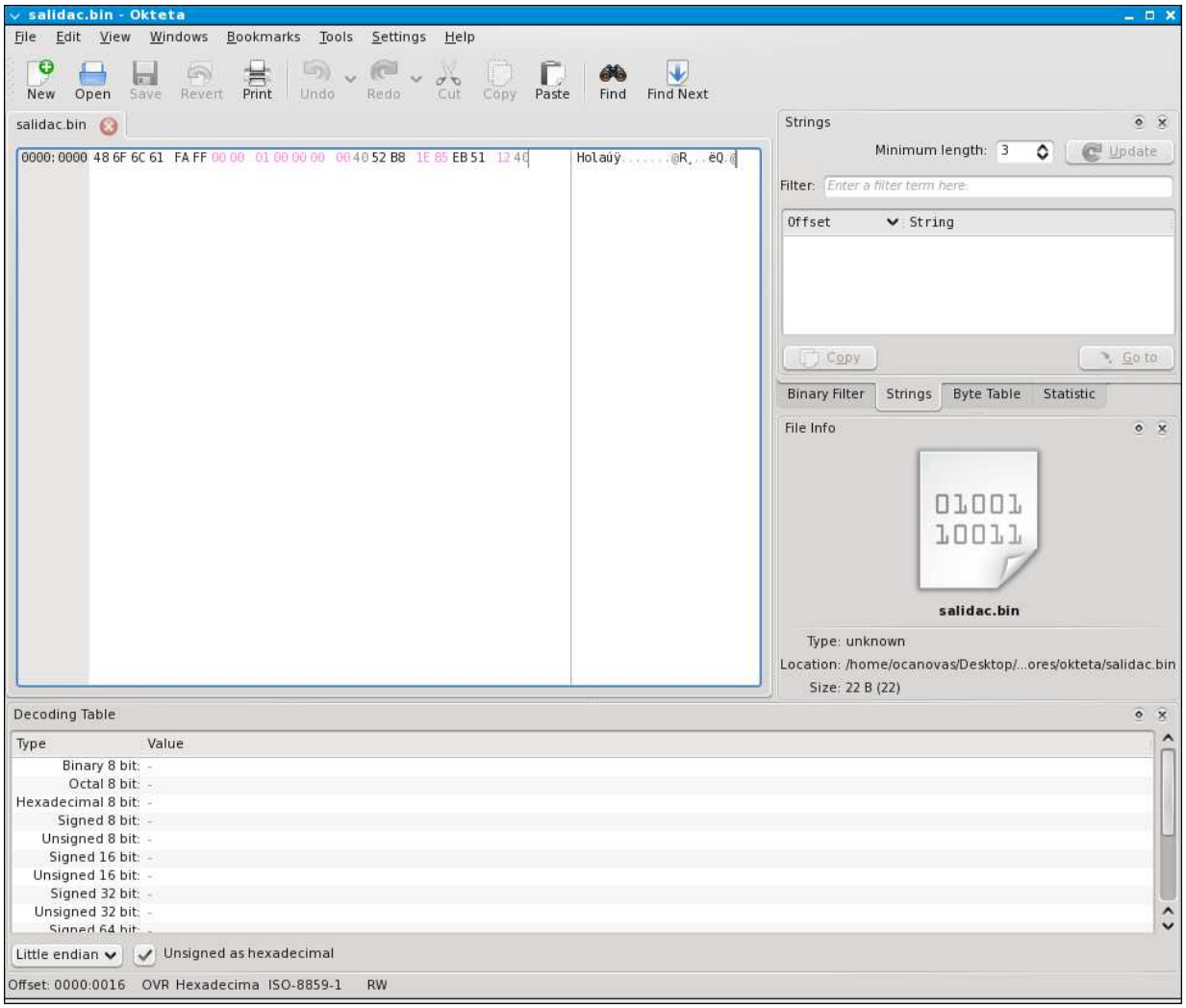

Figura I.2: Fichero binario abierto con okteta.

- Una cadena de 4 caracteres.
- Un entero con signo de 16 bits (en complemento a 2 si es negativo)
- Un entero con signo de 32 bits (en complemento a 2 si es negativo)
- Un número real en coma flotante de simple precisión (IEEE 754 de 32 bits)
- Un número real en coma flotante de doble precisión (IEEE 754 de 64 bits)

Para decodificar la información almacenada podemos proceder de dos formas. La primera es hacerlo de manera manual, anotando los valores almacenados en las posiciones correspondientes y después decodificando a mano. Para ello siempre es necesario saber cuándo empieza un determinado dato (su posición o desplazamiento dentro del fichero), además del formato de almacenamiento que se está usando (que en este caso es little endian). La segunda forma de hacerlo es más sencilla y consiste en aprovechar algunas de las funcionalidades que nos da okteta. Colocando el cursor sobre el primer byte de cada dato, podemos buscar en la tabla de decodificación cuál sería el valor correspondiente a dicho dato una vez decodificado. Procedamos de esta segunda manera para ir interpretando todos los datos almacenados:

1. Para conocer cuáles son los 4 primeros caracteres, nos basta con mirar al principio de la segunda columna, la que

se encarga de interpretar la información binaria como si se tratara de caracteres. En ella podemos contemplar que esos cuatro bytes (48 6F 6C 61) codifican "Hola".

2. Para saber cuál es el valor del entero con signo de 16 bits hay que situar el cursor sobre el quinto byte (Offset = 4), dado que es el primero de los dos que se utilizan para almacenar el valor (FA FF). Al hacerlo comprobaremos cómo en la tabla de decodificación se nos muestra que el valor correspondiente es -6. La figura I.3 nos resalta aquellos aspectos de la interfaz de okteta que acabamos de comentar.

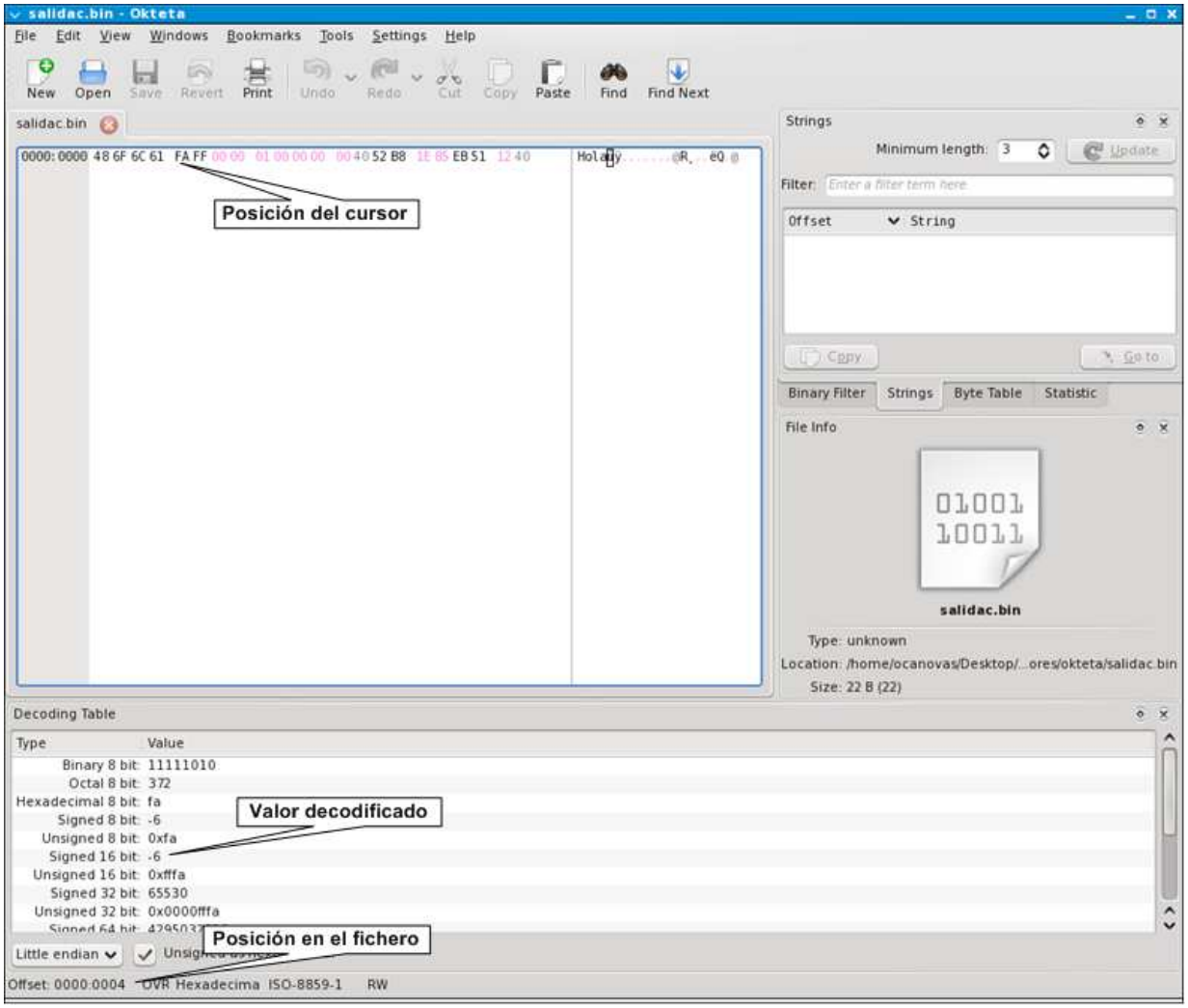

Figura I.3: Decodificación del entero con signo de 16 bits.

3. Procederemos de forma similar para interpretar el entero con signo de 32 bits. Para ello habrá que colocar el cursor sobre el séptimo byte (Offset = 6), puesto que es el primero de los cuatro que almacenan el valor (00 00 01 00). Al hacerlo observaremos que en la tabla de decodificación se nos indica que al interpretarlo como un entero con signo de 32 bits su valor es 65536.

4. Por último, los valores reales codificados mediante IEEE 754 de simple y doble precisión se decodificarían siguiendo el mismo procedimiento, colocando esta vez el cursor en la posición 0A y 0E respectivamente. Los bytes correspondientes en hexadecimal son 00 y 52, con valores decodificados de 2 y 4.58 respectivamente.

B3.1.5. Ejercicios a realizar durante la sesión

1. Interpretar los contenidos del fichero *datos.bin* con la herramienta okteta. Dicho fichero contiene, por este orden, los siguientes datos[2]:
   - una cadena de seis caracteres
   - un número real representado mediante IEEE 754 de doble precisión
   - un entero con signo de 32 bits
   - un número real representado mediante IEEE 754 de simple precisión
   - un entero con signo de 16 bits
2. Modificar los contenidos del fichero anterior de manera que ahora se almacenen los siguientes valores mediante las codificaciones correspondientes:
   - una cadena de seis caracteres que representen la palabra "PRUEBA"
   - un número real representado mediante IEEE 754 de doble precisión, con valor 8.75
   - un entero con signo de 32 bits, con valor -140506
   - un número real representado mediante IEEE 754 de simple precisión, con valor 5
   - un entero con signo de 16 bits, con valor 36

[2]**Nota:** Todos los ejercicios propuestos en este boletín utilizan convenio de almacenamiento *little-endian*, allí donde sea necesario (almacenamiento de distintos tipos de valores que ocupen más de un byte: enteros con/sin signo, o números reales en simple o en doble precisión).

B3.2. Boletín 2. Decodificación de ficheros con Okteta (II)

B3.2.1. Objetivos

Este segundo boletín del tema 2 tiene como objetivo fundamental acercar al alumno los detalles de la codificación de datos de más alto nivel. Una vez que en la sesión anterior se analizó la codificación de los números enteros, reales y las cadenas de caracteres, esta sesión va un paso más allá para abarcar la representación de información de más alto nivel, como son las imágenes o las páginas Web. Como se verá, en muchas ocasiones la representación de esta información de alto nivel se realiza componiendo distintos datos de tipo más básico, por lo que algunos aspectos analizados durante esta sesión tomarán como base lo visto en el anterior boletín.

Para el análisis de las imágenes haremos uso de nuevo de la herramienta okteta, ya que nos permitirá analizar perfectamente el formato de representación PPM, elegido en esta práctica para ilustrar la codificación de imágenes. Una vez presentadas las caracterísiticas de dicho formato, se realizará tanto la interpretación de imágenes ya codificadas como la generación de imágenes utizando dicho formato.

Por otro lado, esta sesión introduce los fundamentos de la representación de documentos digitales como las páginas Web o los textos y hojas de cálculo generados mediante herramientas ofimáticas. Se trata de dar un vistazo general a la representación de la información con el fin de conocer los detalles más básicos que caracterizan a este tipo de codificaciones. Uno de los objetivos es que los alumnos puedan entender y crear páginas Web sencillas, que podrán ser visualizadas posteriormente en un navegador. Además, también se pretende que conozcan las diferentes posibilidades que hay a la hora de codificar la información generada por los procesadores de texto o las hojas de cálculo.

B3.2.2. Plan de trabajo

El plan de trabajo de esta sesión será el siguiente:

1. Lectura del boletín por parte del alumno.
2. Presentación de los formatos de codificación y análisis de ejemplos.
3. Realización de forma individual de los ejercicios propuestos en el boletín (con supervisión del profesor).

B3.2.3. El formato de representación de imágenes PPM

El nombre PPM es un acrónimo del inglés "*Portable Pixel Map*". Se trata de un formato de representación obviamente ineficiente por varias razones. Contiene más información de la que el ser humano es capaz de distinguir. Además, la información está representada sin ningún tipo de compresión, lo que hace que sus fichero ocupen más espacio del que es habitual para imágenes del mismo tamaño. Sin embargo, se trata de un formato muy sencillo de interpretar y que es perfecto para afianzar los principios básicos de la representación digital de imágenes. En consecuencia, será el formato elegido para realizar algunos ejercicios de esta sesión de prácticas.

Cada imagen almacenada mediante PPM está formada por los siguientes elementos:

1. Un número "mágico"que identifica el tipo de fichero. El número mágico de una imágen PPM son los caracteres **P6**.
2. Caracteres que se utilizan como delimitadores (espacios, tabuladores, retorno de carro, alimentación de línea). Suele utilizarse el carácter 0A para delimitar los distintos elementos.
3. Una anchura, especificada en decimal mediante caracteres ASCII.
4. Una altura, especificada de la misma manera.
5. El valor máximo que puede tomar un color. Debe ser menor a 65536 y superior a 0. También se codifica en decimal mediante caracteres ASCII.

6. Una ristra de bytes que representan los píxeles de la imagen por filas, de arriba a abajo y de izquierda a derecha. Cada píxel es una tripleta de valores que representan la intensidad de los colores rojo, verde y azul, en ese orden. Cada valor de color está representado mediante binario puro. En el caso de que el valor de color se represente mediante dos bytes (debido a que el valor máximo de color sea superior a 255) el byte más significativo es el primero.

7. Los ficheros PPM también pueden contener comentarios que empiezan por el carácter # y terminan con un delimitador.

Hay otra versión del formato PPM que almacena los valores de los píxeles mediante valores decimales codificados en ASCII. Sin embargo, nosotros no haremos uso de dicho formato durante el desarrollo de estas prácticas.

La Figura I.4 muestra un ejemplo muy sencillo de los valores de pixel (en decimal) de una imagen de dimensión 4x2 que contiene 8 pixeles de colores diferentes (los retornos de carro introducidos no son parte de la representación, sólo están ahí para facilitar la interpretación). El valor máximo que puede tomar cada color es 255.

```
P3
4 2
255
255   0   0   0 255   0   0   0 255   0   0   0
255 255   0   0 255 255 255   0 255 255 255 255
```

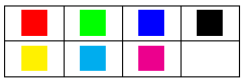

Figura I.4: Ejemplo de codificación de imagén en formato PPM.

B3.2.4. Ejemplo de interpretación de una imagen PPM mediante okteta

Veamos mediante un ejemplo cómo es posible interpretar una imagen mediante okteta. Al abrir el fichero **blanco.ppm**, okteta nos muestra la información que aparce en la Figura I.5.

Comprobamos que mediante okteta resulta sencillo llevar a cabo la interpretación de los parámetros básicos de la imagen. Tal y como se muestra en la figura, podemos averiguar los siguientes datos:

- Se trata de una imagen PPM puesto que el número mágico es P6.
- Se trata de una imagen de 33x33 píxeles
- El valor máximo de color es 255, lo que implicará que cada píxel estará formado por tres bytes representando el rojo, verde y azul respectivamente.
- Comprobamos que los píxeles que se muestran en pantalla son píxeles blancos, puesto que todos tienen valor FFFFFF (cuando todos los colores tienen el valor máximo, el píxel resultante es blanco).

B3.2.5. Representación de documentos digitales

El segundo tipo de documento que vamos a analizar durante esta sesión son los documentos digitales, aquellos que se generan mediante la ayuda de procesadores de texto avanzados, software para hojas de cálculo, o aquellos que se redactan para ser visualizados mediante un navegador Web. Haremos un repaso muy introductorio a las principales características de varios de los formatos de representación más comunes con el fin de conocer de qué manera se almacenan los documentos. Concretamente, nos centraremos en el lenguaje HTML y en las recomendaciones de Open Document Format.

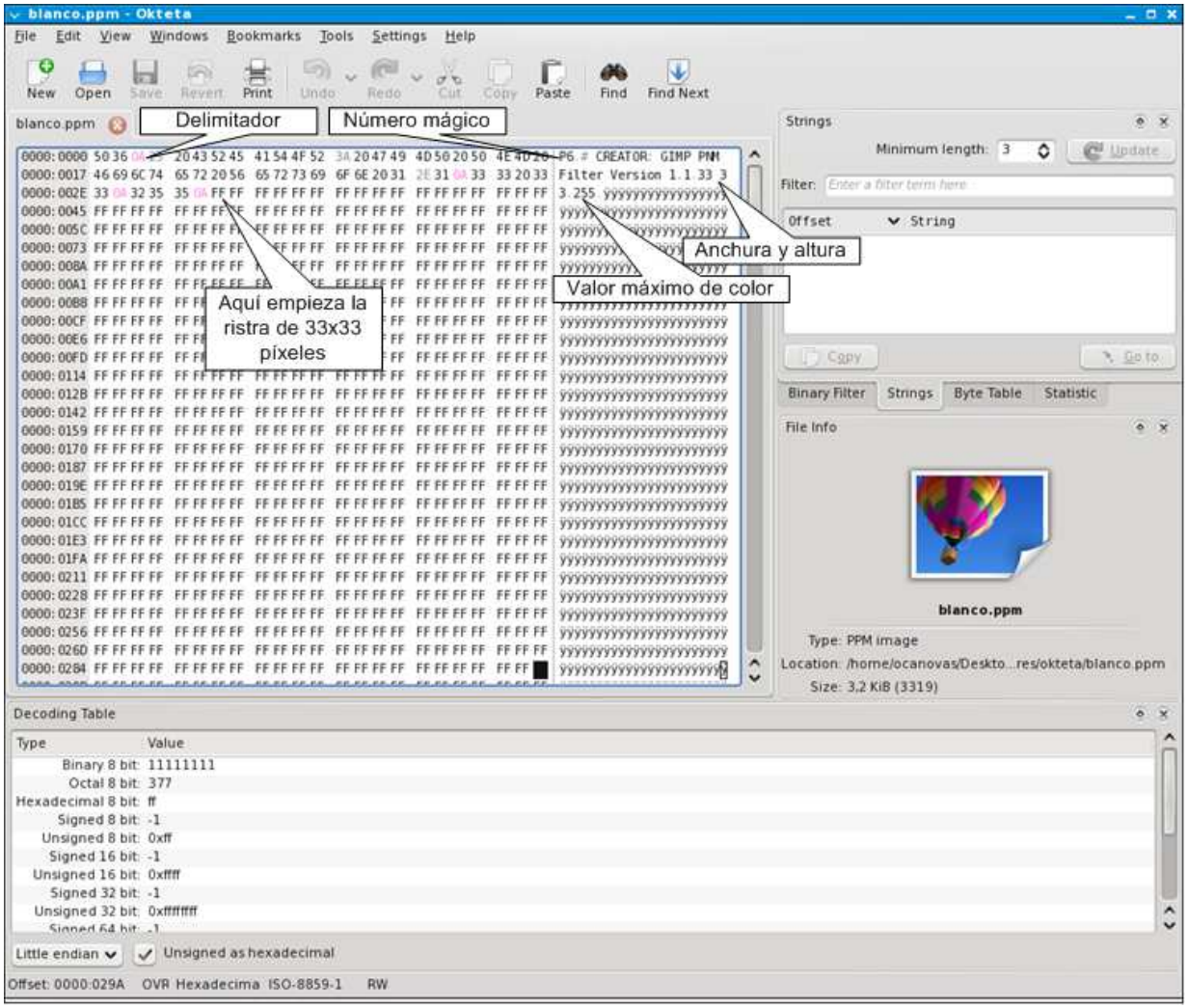

Figura I.5: Fichero PPM en Okteta.

HTML (HiperText Markup Language)

El lenguaje HTML es un lenguaje para crear páginas Web. Los datos para una página Web pueden ser formateados para su interpretación por un navegador. Por ejemplo, para que un texto se muestre en negrita con HTML se debe incluir una etiqueta de comienzo `<b>` y de final `</b>` que indique que el texto debe mostrarse en negrita. Es decir, `<b>Esto está en negrita</b>` se mostraría como **Esto está en negrita**. Las etiquetas son instrucciones para el navegador que se especifican mediante un nombre seguido de una lista opcional de atributos, todos ellos encerrados entre los símbolos menor y mayor.

Aunque, en general, HTML permite usar un repertorio de miles de caracteres, basándose en Unicode (para soportar los distintos idiomas utilizados a nivel mundial), en estas prácticas, por simplicidad, usaremos el reducido conjunto de caracters ASCII, para codificar tanto el texto principal como para las instrucciones de formato.

Una página Web puede constar de dos partes: la cabecera y el cuerpo. La cabecera es la primera parte de la página Web. Contiene el título de la página y otros parámetros últiles para el navegador. El contenido real de la página se encuentra en el cuerpo, que incluye el texto y las etiquetas.

Quizá la mejor forma de familiarizarse un poco con HTML es examinar un ejemplo concreto. A continuación se muestra el fichero **prueba.html** que es un ejemplo de página Web sencilla.

```
1. <HTML>
2. <HEAD>
```

```
3.  <TITLE>Ejemplo</TITLE>
4.  <META NAME="AUTHOR" CONTENT="Fundamentos de Computadores">
5. </HEAD>
6. <BODY>
7. <P>Esta palabra est&aacute; en <B>negrita</B>.</P>
8. <P>Tambi&eacute;n la puedo poner en <I>cursiva</I>.</P>
9. <BR>
10. <P>Un <A HREF="http://www.um.es/">enlace</A> al Web de la UMU.</P>
11. </BODY>
12. </HTML>
```

Analicemos varios aspectos del ejemplo anterior:

- En primer lugar aparece la cabecera del documento, comprendida entre la línea 2 y la línea 5, tal y como delimitan las etiquetas `HEAD`.
- Se trata de un documento, como se indica en la línea 3, titulado "Ejemplo". Este será el título que se muestre en la ventana de nuestro navegador al visualizarlo.
- Se incluye como metainformación el nombre del autor del documento, en la línea 4.
- En el cuerpo de la página aparecen distintos formatos de presentación. En la línea 7 se muestra como poner una palabra en negrita. En la línea 8 cómo ponerla en cursiva.
- Tanto en la línea 7 como en la 8 podemos ver que las tildes tienen su propia codificación. Por ejemplo, *á* se utiliza para indicar que la letra a debe tener un acento agudo. No es estrictamente necesario introducir este tipo de códigos para las tildes, dado que se puede especificar en la cabecera de la página el tipo de codificación de caracteres utilizado. Sin embargo, es útil para garantizar la visualización correcta independientemente del sistema de codificación que se utilice.
- Las etiquetas `P` se utilizan para indicar cuándo comienza y termina un párrafo.
- Las etiquetas `BR` se utilizan para introducir líneas en blanco.
- En la línea 10 aparece una característica clave de HTML, la posibilidad de enlazar las páginas Web entre sí mediante la etiqueta `A`. Al pulsar sobre esta palabra con el navegador se nos redirigirá a la página Web indicada mediante el atributo `HREF`.
- Finalmente, se indica en la línea 12 que el documento HTML ha terminado y que no hay más contenido que mostrar.

Open Document Format

El formato OpenDocument (ODF) es un formato de fichero estándar para el almacenamiento de documentos ofimáticos tales como hojas de cálculo, informes, gráficos y presentaciones. El estándar fue desarrollado públicamente por un grupo de organizaciones, es de acceso libre, y puede ser implementado por cualquiera sin restricción.

Un fichero OpenDocument es un archivo comprimido en ZIP y que contiene varios ficheros y directorios. El formato OpenDocument ofrece una clara separación entre el contenido, la disposición de éste en el documento y los metadatos. Los componentes más notables del formato son los siguientes:

- content.xml: Este el el fichero más importante. Almacena el contenido real del documento (excepto los datos binarios como las imágenes). El formato de base utilizado está inspirado por el HTML, aunque es bastante más complejo que éste, y es razonablemente legible para un humano. Las imágenes se almacenan en un directorio aparte en su formato original, normalmente.

- styles.xml: OpenDocument hace un uso intensivo de los estilos para el formato y disposición del contenido. La mayor parte de la información de estilo se almacena en este fichero (aunque hay parte que aparece en el fichero content.xml). Hay diferentes tipos de estilo, como los de carácter, párrafo, etc.
- meta.xml: Contiene los metadatos del documento. Por ejemplo, el autor, la identificación de la última persona que lo modificó, la fecha de última modificación, etc.
- settings.xml: Este fichero incluye propiedades como el factor de zoom o la posición del cursor que afectan a la apertura inicial del documento, pero no son contenido ni afectan a la disposición de éste en el documento.

B3.2.6. Ejercicios a realizar durante la sesión

1. Dada una imagen PPM cualquiera, y sabiendo el desplazamiento $desp$ dentro del fichero donde comienza la secuencia de valores RGB del primer píxel (es decir, justo a continuación de donde termina la cabecera), determina cuál es la fórmula que permite averiguar la posición exacta dentro del fichero en la que comienzan los tres bytes R, G y B correspondientes a un píxel genérico de coordenadas (x, y).
2. A partir del fichero blanco.ppm, el cual contiene una imagen con todos los píxeles blancos, utiliza okteta para modificar dicho fichero de forma que ahora contenga un cuadrado azul de 3x3 píxeles en la esquina superior izquierda de la imagen.
3. Modifica el fichero resultante del ejercicio anterior para que además tenga un cuadrado negro de 3x3 justo en el centro de la imagen.
4. Genera un documento HTML en el que aparezca la siguiente información, cada una en una línea, formateada según se especifica. Comprueba que se visualiza correctamente mediante un navegador.
   a) Nombre y apellidos de los miembros del grupo de prácticas, en negrita.
   b) DNI de los miembros del grupo, en cursiva.
   c) Una línea en blanco.
   d) El nombre de un país y el correspondiente enlace a la página de la wikipedia en la que se proporciona información sobre dicho país.
   e) La página deberá titularse "Mi primera Web en Fundamentos de Computadores".
5. Genera los siguientes documentos mediante LibreOffice:
   a) Un documento de texto que contenga la dirección postal completa de los miembros del grupo. Escribe en negrita la ciudad de residencia.
   b) Una hoja de cálculo cuyo contenido esté formado por dos columnas de números y una tercera columna cuyo valor sea la suma de los números que hay en las dos columnas previas.
6. Usando los documentos generados en el ejercicio anterior, vamos a proceder al análisis de la codificación de dichos documentos. Para ello debes descomprimir el contenido de los archivos .odt y .ods. Una vez descomprimidos, busca en los ficheros los contenidos que introdujiste mediante LibreOffice así como los formatos que hayas aplicado al contenido.