Aplicaciones Internet

Ramon Martí Escalé
Xavier Perramon Tornil

P03/75064/00979

UOC
www.uoc.edu

Índice

Introducción

En este módulo didáctico se describe toda una serie de aplicaciones (programas) que utilizan Internet como medio de comunicación, así como los protocolos de comunicaciones que tienen asociados. Dichas aplicaciones se conocen como **aplicaciones distribuidas**, puesto que están formadas por diferentes partes y cada una se encuentra en máquinas diferentes.

Por norma general, hay una parte llamada **servidor** que se ejecuta en un ordenador, a la que se conectan los diferentes **clientes** (que se encuentran en otros ordenadores remotos) para requerir sus servicios (por lo general, solicitan la ejecución de algún tipo de operación).

En este módulo, veremos en primer lugar algunos conceptos básicos referentes a las aplicaciones distribuidas y después, una serie de **aplicaciones distribuidas en Internet**, de las que estudiaremos las características siguientes:

- El **modelo**: describimos los diferentes elementos que forman la aplicación y sus características.

- El **protocolo**: exponemos las ideas principales del protocolo de comunicaciones de cada aplicación.

- El **modelo de información**: describimos el formato de información de los protocolos que disponen de uno concreto.

- La **funcionalidad**: explicamos la funcionalidad y los comandos que la proporcionan.

Incluimos ejemplos breves del intercambio de comandos para facilitar su comprensión.

Objetivos

En este módulo didáctico, encontraréis las herramientas necesarias para lograr los objetivos siguientes:

1. Comprender el modelo cliente/servidor, que sirve como base de la implementación de aplicaciones distribuidas.

2. Conocer las aplicaciones distribuidas más utilizadas en Internet y, sobre todo, sus protocolos de comunicaciones: servicios básicos, servicio de nombres de dominio, transferencia de ficheros, correo electrónico, servicio de noticias, servicio hipermedia y servicio de directorio.

3. Ver algunos protocolos de comunicaciones para aplicaciones multimedia en tiempo real.

4. Aprender los principios básicos de la seguridad en el ámbito de las comunicaciones.

1. El modelo cliente/servidor

Las redes de computadores han hecho aparecer un concepto nuevo en el mundo de la programación: la **programación distribuida**. Con esta última se pretende aprovechar la potencia y los recursos de los ordenadores interconectados para llevar a cabo una tarea de forma cooperativa.

Una aplicación distribuida está formada por varios programas que se ejecutan en ordenadores diferentes y que se comunican por medio de la red que une a los ordenadores.

Conviene destacar que cada programa, por sí solo, no puede hacer nada. Es necesaria la **colaboración** de todos para que la aplicación en conjunto produzca resultados útiles.

La cooperación de los diferentes trozos de código que forman la aplicación debe seguir un **protocolo**. Este último se puede elaborar a medida para cada aplicación que se desarrolle, o se puede definir un estándar que todo el mundo pueda seguir y así ahorrarse, de este modo, los diseños particulares. Asimismo, la existencia de un protocolo estándar garantiza la posibilidad de interactuar con productos de diferentes fabricantes.

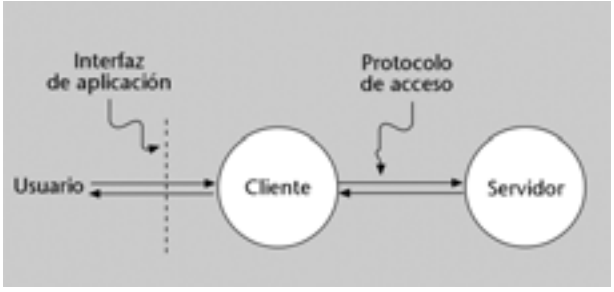
Existen varios estándares de aplicaciones distribuidas; sin embargo, sin lugar a dudas, el que ha tenido más éxito es el que sigue el modelo cliente/servidor.

En el modelo cliente/servidor, la aplicación se divide en dos partes, con dos roles claramente diferenciados:

Servidor: ofrece un servicio que puede ser el acceso a un recurso, la ejecución de operaciones matemáticas complejas, procesamiento de datos, etc

Cliente: realiza una petición al servidor y espera un resultado de la misma.

Por norma general (al menos es lo que dice el modelo), los usuarios acceden a la aplicación por medio del cliente.



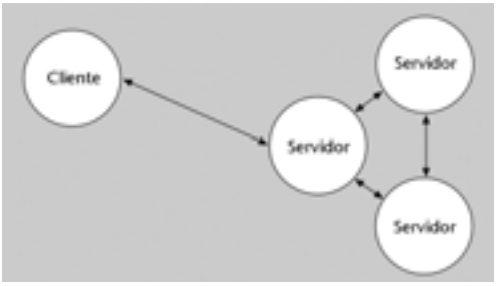
El usuario de las aplicaciones puede ser una persona, pero también otra aplicación. En el primer caso, el cliente debe disponer de una **interfaz de usuario** que permita el diálogo, canalizando las peticiones y mostrándole los resultados. En el segundo, el acceso se suele implementar con llamadas a funciones.

El modelo se puede complicar si se añaden más entidades a la aplicación distribuida. Veamos dos ejemplos:

- Múltiples clientes:

El servidor debe estar preparado para recibir múltiples conexiones, que pueden ser simultáneas.

- Múltiples servidores:



Varios servidores ofrecen servicios complementarios. Si el servidor no puede satisfacer la petición del cliente, pero está en contacto con otro servidor que sí que puede hacerlo, entonces se lo solicita.
Los servidores pueden dialogar entre sí con protocolos propios o siguiendo el modelo cliente/servidor.

Desde el punto de vista del cliente, esta multiplicidad de servidores es completamente transparente: realiza una petición a un servidor y espera una respuesta del mismo.

Cuando diferentes servidores cooperan para ofrecer un servicio, hablamos de **sistema servidor**.

La gran mayoría de las aplicaciones que se utilizan en Internet siguen este modelo cliente/servidor. De aquí los nombres que se emplean con asiduidad: servidor web, cliente de correo, servicio de nombres, etc.

El diseño de una aplicación distribuida que siga el modelo cliente/servidor incluye básicamente dos elementos: **la especificación de los servicios** que ofrece el servidor y **la especificación del protocolo de acceso**, donde se describe cómo se piden estos servicios y cómo debe retornarse el resultado.

2. Servicio de nombres Internet

La red Internet permite el acceso a una ingente cantidad de ordenadores y, en general, de recursos, la mayoría de los cuales se pueden referenciar mediante un nombre. Por motivos de eficiencia y simplicidad (poder saber con facilidad qué recurso corresponde a cada nombre, poder asignar otros nuevos sin peligro de duplicidades o ambigüedades, etc.), todos los nombres de la red están organizados de una manera jerárquica, formando un **sistema de dominios**.

Para obtener información referente a cualquier nombre, se utiliza un servicio que, funcionalmente, es como una base de datos: se le hacen consultas, que pueden incluir una serie de criterios de selección, y responde con la información solicitada. En un inicio, cuando el número de ordenadores conectados a la red era relativamente pequeño, esta base de datos era gestionada por una única autoridad central, el Network Information Center (NIC).

A medida que se expandía la red, este modelo de gestión se iba haciendo cada vez más inviable, tanto por el enorme tráfico que generaban las consultas, como por la dificultad de mantener la información actualizada. Fue entonces cuando nació el **servicio de nombres de dominio** (DNS), que sigue el modelo de una base de datos distribuida descentralizada.

La especificación del DNS está contenida en los documentos RFC 1034, que define los conceptos y la organización del sistema de dominios, y RFC 1035, que define el protocolo de acceso al servicio.

Lecturas complementarias

Si queréis más información sobre el DNS, podéis consultar las obras siguientes:

P.V. Mockapetris (1987, 1 de noviembre). *RFC 1034- Domain names - concepts and facilities*.

P.V. Mockapetris (1987, 1 de noviembre). *RFC 1035- Domain names - implementation and specification*.

2.1. El sistema de nombres de dominio

El sistema de nombres de dominio, en que se basa el DNS, proporciona un espacio de nombres para referenciar recursos, que por norma general son ordenadores conectados a la red, pero que también pueden ser, por ejemplo, buzones de correo electrónico.

En el sistema de nombres de dominio, los nombres están organizados jerárquicamente en forma de árbol. Cada nodo de este último tiene una etiqueta que lo distingue de sus nodos "hermanos".

El **nombre de dominio** correspondiente a un nodo se define como la secuencia formada por las etiquetas existentes en el camino entre este nodo y la raíz.

Cada etiqueta constituye una cadena de caracteres de longitud comprendida entre 0 y 63 caracteres. La etiqueta con longitud 0 está reservada para el nodo raíz: ningún otro nodo puede tener una etiqueta vacía.

El protocolo de acceso al DNS define el formato para representar los nombres de dominio cuando deben enviarse en las consultas y las respuestas, como veremos más adelante. Este formato no impone restricciones en los caracteres que puede haber en una etiqueta; sin embargo, establece una "sintaxis preferida" para facilitar las implementaciones de otros servicios. De acuerdo con esta última, los únicos caracteres válidos en una etiqueta son las letras, mayúsculas o minúsculas, los dígitos decimales y el símbolo "`-`", con la excepción de que el primer carácter sólo puede ser una letra, y el último no puede ser "`-`".

Mayúsculas y minúsculas

A la hora de comparar nombres, las letras mayúsculas y minúsculas se consideran equivalentes. No obstante, para responder las consultas, los servidores DNS deben enviar los nombres tal como están almacenados en la base de datos, respetando las mayúsculas y minúsculas, en previsión del caso de que en alguna versión futura del servicio se consideren diferentes.

Aparte de la representación definida en el protocolo, cuando se desea utilizar una notación textual para expresar un nombre de dominio, se sigue la convención de concatenar las etiquetas que lo forman, separándolas con el carácter "`.`". En el orden utilizado, las etiquetas van desde el más bajo nivel jerárquico más bajo hasta el más alto. Por tanto, la última etiqueta es la del nodo raíz que, como acabamos de observar, está vacía, de manera que los nombres acaban en "`.`", como en el ejemplo siguiente:

```
campus.uoc.edu.
```

Este ejemplo es el que se denomina un nombre de dominio absoluto, pero también pueden utilizarse nombres relativos, referentes a un nodo origen implícito. Así, dentro del dominio `uoc.edu.` podemos utilizar el nombre `tibet` para referirnos al nombre de dominio `tibet.uoc.edu.`. En numerosas ocasiones, el nodo origen de los nombres relativos es la raíz, de manera que el ejemplo anterior también puede escribirse del modo siguiente:

En el Reino Unido...

... antes era usual utilizar el orden inverso en la representación textual de los nombres de dominio, por ejemplo `uk.ac.ic.doc.src` en lugar de `src.doc.ic.ac.uk`; sin embargo, hoy día esta costumbre se encuentra en desuso.

```
campus.uoc.es
```

En un inicio, los dominios de nivel superior* (los subordinados directamente a la raíz) se utilizaban para agrupar los diferentes tipos de organizaciones a que pertenecían recursos como, por ejemplo, los que presentamos en la tabla siguiente:

*** En inglés, *top level domain* (TLD).**

Tipos de organización	Nombre del dominio
Empresas y organizaciones comerciales	com
Universidades y centros educativos	edu
Organismos gubernamentales	gov
Organizaciones militares	mil
Otros tipos	org

Desde entonces hasta la actualidad, se han ido añadiendo al nivel superior del sistema de nombres los elementos siguientes:

a) Por un lado, nuevas clases de organizaciones, como `net` (centros relacionados con la gestión de la red), `int` (organizaciones internacionales) o, más recientemente, `biz`, `info`, `name`, `pro`, `aero`, `coop` y `museum`.

En estos momentos, el organismo encargado de gestionar los dominios es la ICANN. Podéis consultar su web: www.icann.org.

b) Por otro lado, cada país tiene asignado un nombre de dominio que coincide con el código correspondiente de dos letras especificado en el estándar ISO 3166, por ejemplo: `es` (España), `fr` (Francia), `de` (Alemania), etc. Lo que significa que, en realidad, las diferentes ramas del árbol se pueden encabalgar y, de hecho, hay nombres pertenecientes a diferentes dominios de alto nivel que corresponden a una misma organización.

Asimismo, es posible que un determinado recurso (ordenador, buzón de correo, etc.) tenga asignados varios nombres, que pueden estar en la misma rama del árbol o en ramas completamente independientes. En este caso, se considera que uno de los nombres es el denominado *nombre canónico* y los otros se denominan *alias*.

2.2. Modelo del DNS

Los agentes que intervienen en el DNS son los siguientes:

a) Los **servidores**, que reciben las consultas y envían las respuestas correspondientes. Además del acceso directo a una base de datos local, en la que se encuentra una parte de la base de datos total del sistema de nombres, el servidor tiene acceso indirecto al resto de la base de datos por medio de otros servidores. El conjunto de servidores DNS constituye, pues, un sistema servidor.

Consultad la definición de *sistema servidor* en el capítulo "El modelo cliente/servidor" de este módulo didáctico.

b) Los **resolvedores**, que actúan como clientes del servicio. Por norma general, un resolvedor es un programa o una librería que recibe peticiones de las aplicaciones de usuario, las traduce a consultas DNS y extrae de las respuestas la información solicitada. El resolvedor debe tener acceso directo al menos a un servidor DNS.

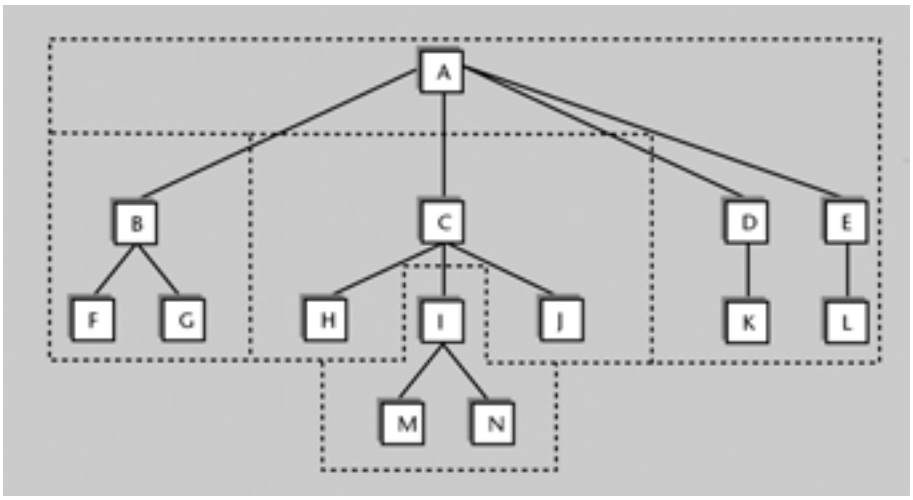
Utilización del resolvedor

Una aplicación que sirva para copiar ficheros almacenados en otros ordenadores puede aceptar que el usuario indique a qué ordenador desea acceder por medio de su nombre. Por consiguiente, para poderse comunicar con él, la aplicación deberá realizar una llamada al resolvedor para obtener su dirección a partir del nombre. Como la aplicación y el resolvedor, por lo general, se encontrarán ubicados en el mismo sistema, no es necesario establecer ningún protocolo de comunicación entre ellos.

Para hacer manejable la gestión y administración del sistema de nombres de dominio, sus nodos están agrupados en zonas. Cada **zona** está formada por un nodo llamado *superior* y todos los que se encuentren en los niveles jerárquica-

mente inferiores hasta llegar a los nodos terminales (las hojas del árbol) o a nodos de otras zonas que sean superiores.

La figura siguiente muestra un ejemplo de división de un árbol en cuatro zonas:



Nombre de las zonas

Como cada zona se suele denominar con el nombre de dominio de su nodo superior, podemos decir que las zonas de la figura son A, B.A, C.A e I.C.A.

El objetivo de agrupar los nodos en zonas consiste en asignar a una autoridad la responsabilidad de gestionar todos los nodos de cada zona. El administrador designado para esta autoridad puede añadir o borrar nodos dentro de su zona, modificar la información de sus nodos o crear nuevas subzonas y delegar su gestión en otras autoridades.

Por ejemplo,...

... una zona puede corresponder a un país y sus subzonas pueden corresponder a organizaciones de este país. Cada organización, a su vez, puede crear subzonas para diferentes departamentos, etc.

La información referente a una zona debe estar almacenada en la base de datos local de un servidor, del que se dice que es un servidor con autoridad sobre esta zona. Este último puede contestar directamente las consultas que reciba sobre los nodos de su zona, sin necesidad de acceder a otros servidores. Es decir, en este caso, el servidor enviará respuestas con autoridad.

Si una consulta se refiere a otra zona, existen dos maneras posibles de generar la respuesta:

- En el **modo no recursivo**, la respuesta únicamente incluye una referencia a algún servidor que puede proporcionar más información. El cliente debe preocuparse de continuar realizando consultas hasta encontrar la respuesta definitiva.

- Si el cliente solicita el **modo recursivo**, es el servidor quien se ocupa de buscar la información donde sea preciso, y sólo puede retornar la respuesta final o un error, pero no referencias a otros servidores.

La especificación DNS establece que todos los servidores deben soportar el modo no recursivo, y que el modo recursivo es opcional*.

*** En la práctica, la mayoría de las consultas de los clientes son en modo recursivo.**

Cuando un servidor debe responder a una consulta en modo recursivo, pide la información a otros servidores y envía la respuesta recibida a quien ha realizado la consulta. Es habitual que el servidor añada la información obtenida a su base de datos, en lo que se denomina *caché*. De este modo, si recibe otra consulta (del mismo cliente o de otro) en la que se solicita la misma información, no será necesario que se la pida de nuevo a otros servidores, sino que puede aprovechar la ya existente en la caché. Sin embargo, es posible que los datos se hayan modificado en el servidor de origen desde que se solicitaron por primera vez. En este caso, pues, el servidor debe avisar al cliente de que le envía una respuesta sin autoridad. Por otro lado, los resolvedores también suelen guardar en una caché propia las respuestas que reciben de los servidores.

Para ofrecer una alta fiabilidad en el servicio, la especificación DNS requiere que para cada zona haya como mínimo dos servidores con autoridad:

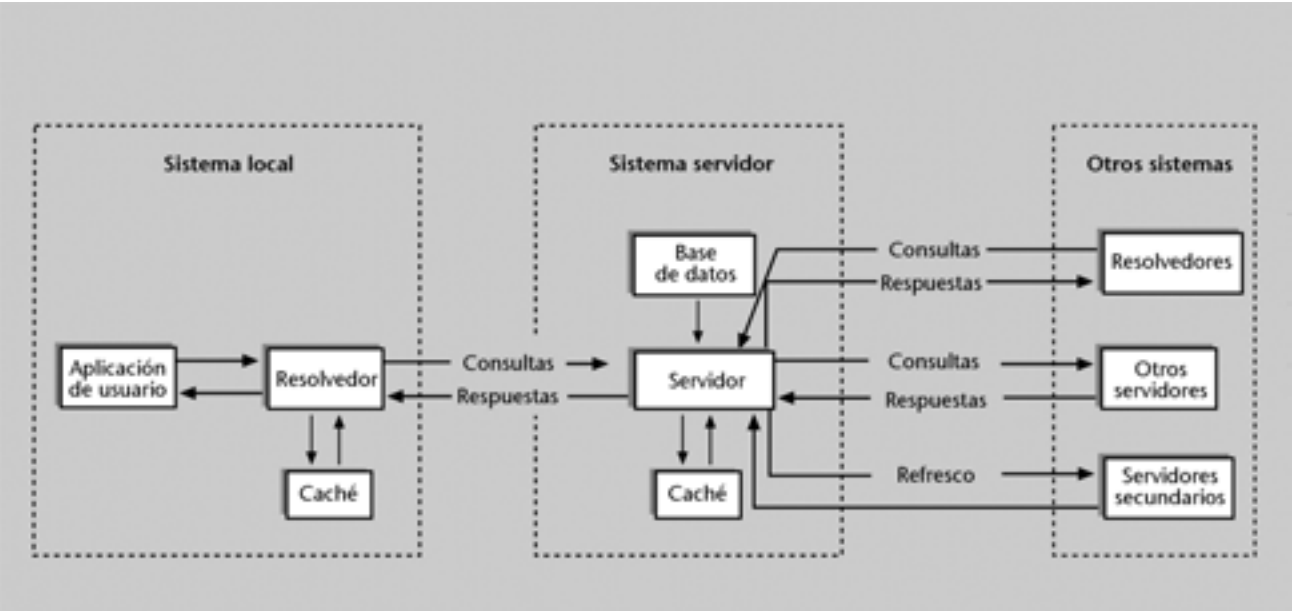
a) Por norma general, hay uno que actúa como **servidor primario** y guarda los ficheros originales de la base de datos correspondiente a la zona; es decir, aquellos que el administrador debe actualizar directamente cada vez que haya una modificación en sus nodos.

b) Los otros actúan como **servidores secundarios** y actualizan automáticamente sus bases de datos a partir de la del primario; por ejemplo, por medio de consultas periódicas para saber si se ha producido algún cambio. De este modo, si un servidor primario está temporalmente inaccesible (por una caída de la red, del mismo servidor, etc.), los clientes pueden enviar sus consultas a uno de los servidores secundarios.

Refrescar

En el argot DNS, ***refrescar los datos*** significa actualizarlos realizando consultas periódicas.

La figura siguiente muestra una posible configuración del servicio DNS en un ordenador:



En la práctica, se pueden encontrar muchas variantes de este esquema; por ejemplo, que el servidor DNS se encuentre en el mismo ordenador que el resolvedor y que ambos compartan la caché, que el resolvedor tenga acceso a diferentes servidores DNS, etc.

2.3. Base de datos DNS: los registros de recurso

Como hemos visto en los subapartados anteriores, un nombre del sistema de dominios identifica un nodo, y para cada nodo puede haber cierta información almacenada en los servidores correspondientes. El objetivo del servicio DNS consiste en permitir la obtención de dicha información a partir del nombre de dominio.

La información asociada a un nodo consta de un conjunto de registros de recurso. Los registros de recurso de todos los nodos forman la **base de datos DNS**.

Cada registro de recurso consta de los campos siguientes, que se explican a continuación:

- Nombre.
- Tipo.
- Clase.
- Tiempo de vida.
- Datos del recurso.

a) Nombre. Contiene el nombre de dominio del nodo al que está asociado el registro.

b) Tipo. Indica qué tipo de información contiene el registro. Los valores que podemos encontrar en este campo y los tipos de información que representan son los siguientes:

- `A`: dirección de un ordenador.
- `CNAME`: nombre canónico de un alias.
- `HINFO`: información sobre el tipo de ordenador.
- `MX` (*Mail eXchanger*): nombre de un servidor de correo electrónico para un dominio.
- `NS`: nombre de un servidor DNS con autoridad para una zona.

- PTR: nombre de un dominio que contiene información relacionada con un nodo.

- SOA (*Start Of Authority*): información sobre el nodo superior de una zona.

Otros valores posibles del campo Tipo son los siguientes:

- MB: nombre de un buzón de correo.
- MG: miembro de un grupo de correo.
- MR: nombre nuevo de un buzón de correo.
- MINFO: información sobre un buzón o una lista de distribución de correo.
- WKS (*Well Known Services*): lista de servicios que proporciona un ordenador.
- TXT: texto descriptivo.
- NULL: registro vacío.

c) Clase. Indica la familia de protocolos utilizada en el espacio de nombres. Por norma general, será la familia de protocolos Internet, pero puede haber otros.

Conjunto de espacios de nombres

El sistema de nombres de dominio proporciona un conjunto de espacios de nombres, uno para cada clase, todos sobre un mismo árbol. Los datos correspondientes a un nodo en una clase pueden ser diferentes de las del mismo nodo en otra clase (un nodo puede no tener ningún registro de recurso de alguna clase), puesto que las bases de datos (como las divisiones en zonas) son separadas e independientes.

d) Tiempo de vida (**TTL**). Indica el tiempo máximo que un servidor o un resolvedor puede guardar el registro en su caché.

e) Datos del recurso (**RDATA**). El valor de este campo depende del tipo de registro:

- Si el registro es de tipo A, en la clase Internet, el valor es un número de 32 bits que corresponde a una dirección IP.

- Si el registro es de tipo CNAME, el valor es un nombre de dominio que corresponde al nombre canónico del alias asociado con el registro. Si un nodo del árbol de nombres es un alias, la única información que puede tener asociada es un registro CNAME.

- Si el registro es de tipo HINFO, el valor es una cadena de caracteres.

- Si el registro es de tipo MX, el valor tiene dos subcampos, el primero es un número que representa una preferencia (cuanto menor es el número, más preferencia) y el segundo es el nombre de un ordenador que está dispuesto a aceptar mensajes destinados al dominio correspondiente al registro.

Uso de los registros MX en el sistema de correo electrónico

El estándar RFC 974 especifica cómo deben utilizarse los registros MX para direccionar un mensaje de correo electrónico dada la dirección del destinatario (por ejemplo, usuari@uoc.edu). De esta dirección se separa la parte correspondiente al dominio de correo (uoc.edu), se consulta el sistema DNS para saber qué servidores aceptan correo para este dominio y se elige uno teniendo en cuenta su preferencia. Si la comunicación con este servidor no tiene éxito, se intenta con los otros por orden de preferencia.

Si no hay ningún registro MX asociado al dominio, se entiende que sólo dispone de un servidor de correo: el ordenador que tenga por nombre el del dominio.

- Si el registro es de tipo NS, el valor es un nombre de ordenador.
- Si el registro es de tipo PTR, el valor es un nombre de dominio.
- Si el registro es de tipo SOA*, el valor del campo RDATA en un registro de este tipo está formado por los siete subcampos siguientes:

– MNAME: nombre del servidor primario de la zona.

– RNAME: nombre de dominio correspondiente al buzón del responsable de la zona.

– SERIAL: número que debe aumentarse cada vez que haya una modificación en los datos de la zona.

– REFRESH: tiempo que debe transcurrir para que los servidores secundarios refresquen sus datos.

– RETRY: tiempo que es preciso esperar para volver a intentar un refrescamiento si no se ha conseguido contactar con el servidor primario.

– EXPIRE: tiempo máximo a partir del cual los datos de un servidor secundario se considerarán sin autoridad si no se han refrescado.

– MINIMUM: valor mínimo del campo TTL en los registros de la zona.

Refrescamiento de una zona en un servidor secundario

El proceso de refrescamiento de una zona en un servidor secundario consiste en pedir al primario su registro SOA y comprobar si ha variado el campo SERIAL. Si ha variado, es preciso llevar a cabo una transferencia de los datos de la zona (por ejemplo, con una petición AXFR del protocolo DNS, por medio del protocolo FTP, etc.) y, si no, no es necesario hacer nada.

*** Cada nodo que sea el superior de una zona debe tener asociado un registro SOA.**

El nombre del buzón...

... puede expresarse como un nombre de dominio siguiendo la sintaxis general de separar los nombres de los nodos con "." (en este caso, la parte correspondiente al usuario sería el nodo inferior).

Por ejemplo, en la bustia usuari@uoc.edu le corresponde el nombre de dominio usuari.uoc.edu.

El protocolo de acceso al DNS, como veremos en el subapartado siguiente, define el formato de representación de los registros de recurso en las respuestas de los servidores. Sin embargo, también es habitual utilizar una representación textual; por ejemplo, en los ficheros de la base de datos que el administrador de una zona puede editar y actualizar. La sintaxis típica de esta representación textual es la siguiente:

```
[nombre] ( [clase] [TTL] | [TTL] [clase] ) tipo RDATA
```

Cada registro se escribe en una línea; sin embargo, si es demasiado largo, se pueden utilizar paréntesis, dentro de los cuales se ignoran los saltos de línea. El símbolo ";" sirve para introducir comentarios en el mismo. El campo Nombre es opcional (por defecto se toma el del registro anterior), como tam-

bién lo son `clase*` y `TTL`, que se pueden escribir en cualquier orden. Los campos que representan dominios pueden ser absolutos (acabados en "`.`") o relativos (respecto a un origen preestablecido), y los que representan tiempo se expresan en segundos.

*** El valor del campo `clase` por norma general es `IN`, que corresponde a la clase Internet.**

El campo `Nombre` puede empezar con la etiqueta "`*`" para indicar que la información del registro se aplica a los dominios que tengan cualquier etiqueta o secuencia de etiquetas en el lugar del "`*`" y no figuren en ningún otro registro.

Líneas del fichero de datos de un servidor `acme.com`

Éstas podrían ser algunas líneas del fichero de datos de un hipotético servidor de la zona `acme.com`:

```
acme.com.   IN   SOA   servidor.acme.com.   admin.acme.com. (
                            38      ; SERIAL
                            7200    ; REFRESH
                            600     ; RETRY
                            3600000 ; EXPIRE
                            60 )    ; MINIMUM
                     NS  servidor.acme.com.
                     NS  servidor2.acme.com.
                     NS  dns.competencia.com.
                     MX  10  correo.acme.com.
                     MX  20  servidor.acme.com.
*.acme.com.          MX  10  correo.acme.com.
servidor.acme.com.   A   128.52.46.32
                     A   128.32.11.99
servidor2.acme.com.  A   128.52.46.33
correo.acme.com.     A   128.52.46.34
```

Un uso habitual de los registros `PTR` es facilitar la obtención del nombre de un ordenador a partir de su dirección por medio de un dominio especial denominado `IN-ADDR.ARPA`. El DNS proporciona una operación para efectuar consultas inversas; es decir, dado un registro, retorna el nombre de dominio a que corresponde. Sin embargo, esta operación puede ser muy costosa para el servidor, y no se puede garantizar que la respuesta sea única o completa, puesto que puede haber otros servidores que contengan información relacionada.

No obstante, como la traducción de direcciones IP a nombres es útil e, incluso, necesaria, en algunos protocolos* se ha adoptado un mecanismo para facilitarla que consiste en introducir registros adicionales que actúan como índice. De este modo, cada vez que se modifica una dirección en la base de datos DNS, o se añade una, es necesario actualizar dos registros:

*** Algunos de los protocolos que necesitan el servicio de traducción de direcciones son rlogin y rsh.**

- El registro `A` que tiene por nombre el del ordenador.
- Un registro `PTR` que tiene un nombre perteneciente al dominio `IN-ADDR.ARPA`.

Los nombres del dominio `IN-ADDR.ARPA` pueden tener hasta cuatro etiquetas*, que representan los valores posibles de los bytes de una dirección IP en

*** Sin tener en cuenta las etiquetas `IN-ADDR` y `ARPA`.**

decimal, entre 0 y 255. De estas etiquetas, la de nivel más alto corresponde al primer byte de la dirección, y la de nivel más bajo, al último byte. De este modo, se puede tener una estructura de zonas y autoridades delegadas similar a la del resto de los dominios. Un nombre del dominio `IN-ADDR.ARPA` que posea las cuatro etiquetas numéricas corresponderá a la dirección de un ordenador, y se le asociará un registro `PTR` que apunte al nombre de este ordenador. Los nombres que dispongan de menos de cuatro etiquetas numéricas serán direcciones de redes y podrán tener asociados registros `PTR` apuntando al nombre de las pasarelas conectadas a estas redes.

Nombres excepcionales

Los nombres del dominio `IN-ADDR.ARPA` son una excepción a la sintaxis preferida para las etiquetas, puesto que deben empezar con un dígito.

Registros de traducción

Siguiendo el ejemplo anterior, los registros del dominio `IN-ADDR.ARPA` siguientes podrían servir para llevar a cabo la traducción de direcciones a nombres:

```
32.46.52.128.IN-ADDR.ARPA.  PTR  servidor.acme.com.
33.46.52.128.IN-ADDR.ARPA.  PTR  servidor2.acme.com.
34.46.52.128.IN-ADDR.ARPA.  PTR  correo.acme.com.
99.11.32.128.IN-ADDR.ARPA.  PTR  servidor.acme.com.
46.52.128.IN-ADDR.ARPA.     NS   servidor.acme.com.
                            NS   servidor2.acme.com.
                            PTR  servidor.acme.com.
11.32.128.IN-ADDR.ARPA.     NS   servidor.acme.com.
                            NS   servidor2.acme.com.
                            PTR  servidor.acme.com.
```

2.4. Protocolo

2.4.1. Mecanismos de transporte

Para acceder al DNS se pueden utilizar los protocolos de transporte UDP o TCP. Por norma general, se utiliza UDP en las consultas de los clientes, por su simplicidad y por los pocos recursos que requiere. Si no llega la respuesta a un datagrama en un tiempo determinado, simplemente se retransmite (hasta que haya transcurrido un tiempo máximo). En cambio, se utiliza TCP cuando conviene asegurar una transmisión fiable; por ejemplo, en las transferencias de datos de una zona de un servidor a otro. Tanto en un caso como en el otro, el número de puerto utilizado es el 53.

Consultad los diferentes protocolos de Internet en el módulo "TCP/IP: los protocolos de la red Internet" de esta asignatura.

Las consultas y las respuestas se intercambian por medio de **mensajes**, cuyo formato depende del protocolo que se utilice:

- En UDP, cada mensaje se envía en un datagrama, con una longitud máxima de 512 bytes.

- En TCP, los mensajes se envían prefijados con 2 bytes, que indican su longitud, para saber dónde acaban, puesto que lo más normal es intercambiar más de uno utilizando una misma conexión.

2.4.2. Mensajes

El esquema siguiente muestra la estructura de un mensaje DNS:

Cada mensaje está formado por una cabecera y cuatro secciones (pregunta, respuesta, autoridad e información adicional)*.

* Algunas de las secciones pueden estar vacías.

La **cabecera** consta de los campos siguientes:

- **ID** es un número de 16 bits que asigna quién realiza la consulta y que se copia en la respuesta para que se sepa a qué consulta corresponde.

- **QR** (*Query/Response*) es un bit que indica si el mensaje es una consulta (0) o una respuesta (1).

- **OPCODE** es un código de operación de 4 bits. En la actualidad, hay definidos los de consulta directa (QUERY, código 0), consulta inversa (IQUERY, código 1) y petición de estatus (STATUS, código 2). Por norma general, el valor de este campo será QUERY, puesto que la operación IQUERY es opcional y poco soportada, y la operación STATUS no está definida en la especificación RFC 1034.

- **AA** (*Authoritative Answer*) es un bit que informa de que el mensaje es una respuesta con autoridad.

- **TC** (*Truncation*) es un bit que avisa que el mensaje se ha truncado porque no cabe en un datagrama. Para saber cuál es su contenido completo, es preciso utilizar el protocolo TCP.

- **RD** (*Recursion Desired*) es un bit que indica que el cliente solicita respuesta en modo recursivo.

- **RA** (*Recursion Available*) es un bit que informa de que el servidor soporta el modo recursivo. Si los bits RD y RA valen 1, el mensaje es una respuesta recursiva.

- **RCODE** es un código de respuesta de 4 bits. Los códigos posibles son: ningún error (0), error de formato (1), error interno del servidor (2), dominio inexistente (3), operación no implementada (4) o consulta rechazada (5).

- **QDCOUNT**, **ANCOUNT**, **NSCOUNT**, **ARCOUNT** son números de 16 bits que indican cuántas entradas hay en cada sección del mensaje: pregunta, respuesta, autoridad e información adicional, respectivamente.

Sección de pregunta

Los mensajes correspondientes a las consultas dispondrán de una o más entradas (por lo general, una) en la sección de pregunta. Cada entrada está formada por tres campos:

a) **QNAME**: nombre de dominio del que el cliente desea obtener información.

b) **QTYPE**: número de 16 bits que indica el tipo de registro de recurso que el cliente quiere obtener como respuesta. El valor de dicho campo puede ser cualquiera de los números que pueden encontrarse en el campo TYPE de un registro, o uno de los siguientes:

- XFR (código 252): se utiliza para solicitar la transferencia de todos los registros de una zona (cuando un servidor secundario quiere refrescar su base de datos).

- MAILB (código 253): sirve para solicitar todos los registros relacionados con buzones (MB, MG y MR).

- * (código 255): sirve para pedir todos los registros de cualquier tipo correspondientes a un nombre.

c) **QCLASS**: número de 16 bits que indica la clase de registros deseados. Su valor puede ser el del campo CLASS de los registros o el valor especial "*" (código 255), que representa todas las clases.

Sección de respuesta

Los mensajes correspondientes a las respuestas contienen en la sección de respuesta el conjunto de registros de recurso que satisfacen los criterios dados por

los campos de la consulta. Es decir, el servidor envía en la respuesta los registros correspondientes al nombre solicitado que concuerdan con el tipo o con los tipos requeridos y en la clase o clases requeridas.

Un caso especial es el de los alias. Si el nombre solicitado corresponde a un alias y el campo QTYPE es diferente de CNAME, el servidor debe repetir automáticamente la consulta sustituyendo el nombre original por el nombre canónico. De este modo, el cliente obtendrá directamente los datos deseados tanto si utiliza un alias, como si emplea el nombre canónico.

La respuesta a una petición AXFR consistirá en una secuencia de mensajes. El primero debe contener el registro SOA de la zona; en los siguientes deben encontrarse el resto de los registros, y el último debe ser otra vez el registro SOA, para que el receptor sepa que ya ha acabado la transferencia.

Sección de autoridad

En una respuesta, la sección de autoridad puede contener registros que referencien un servidor con autoridad para responder la consulta (para el caso de las respuestas en modo no recursivo).

Sección de información adicional

La sección de información adicional puede contener otros registros de recurso relacionados con la consulta, pero que no forman parte de la respuesta.

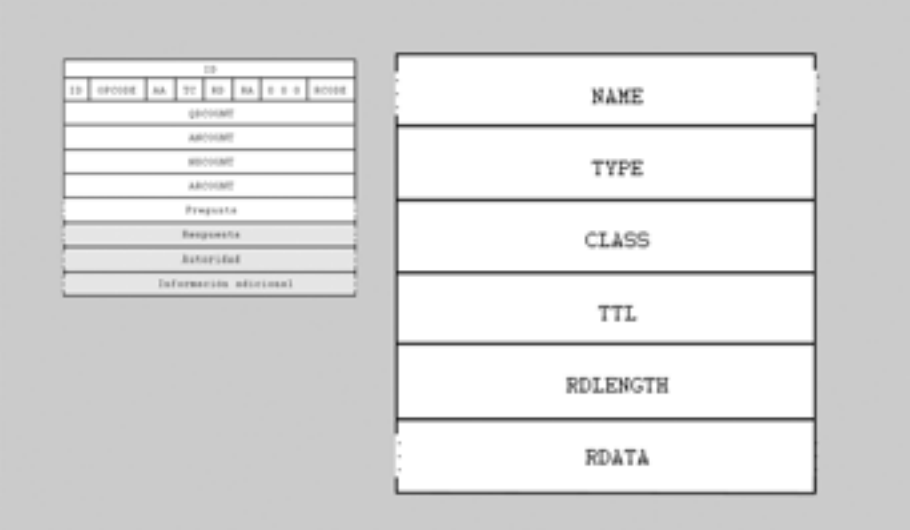
2.4.3. Representación de los registros de recurso

Cuando un mensaje debe contener una cadena de caracteres, se representa por medio de 1 byte, que indica su longitud y, a continuación, los caracteres de la cadena. Un nombre de dominio se representa como una concatenación de cadenas, una para cada una de las etiquetas que lo forman. Como los nombres de dominio acaban con la etiqueta del nodo raíz, una cadena que tenga por longitud 0 (y, por tanto, ningún carácter a continuación) indica el final del nombre. Para simplificar la implementación del servicio, la especificación RFC 1035 requiere que la longitud total de un nombre de dominio no sea superior a 255 bytes.

Representación comprimida de los nombres de dominio

La especificación RFC 1035 también define una representación comprimida de los nombres de dominio cuando hay muchos iguales, o que acaban igual, en un mensaje (como es el caso de los que contienen registros SOA). En lugar de una etiqueta, en un nombre puede haber 1 byte con los 2 bits de más peso a 1 (por tanto, no se puede confundir con la longitud de una etiqueta, que como máximo puede ser 63). Los otros 6 bits y los 8 del byte siguiente indican una posición dentro del mensaje en que se encuentra la continuación del nombre.

La representación de los registros de recurso que pueden aparecer en las secciones de respuesta, autoridad e información adicional sigue el formato siguiente:



Los campos que forman los registros de recurso dentro de los mensajes DNS son los siguientes:

- El **campo `NAME`** es el nombre de dominio correspondiente al registro.

El **campo `TYPE`** es un número de 16 bits que indica el tipo de registro, según la tabla siguiente:

Número	Tipo de registro
1	A
2	NS
5	CNAME
6	SOA
7	MB
8	MG
9	MR
10	NULL
11	WKS
12	PTR
13	HINFO
14	MINFO
15	MX
16	TXT

- El **campo `CLASS`** es un número de 16 bits que indica la clase del registro. Por norma general, su valor es 1, que corresponde a la clase Internet.

- El **campo `TTL`** es un número de 32 bits que indica el tiempo de vida del registro en segundos.

- El **campo `RDLENGTH`** es un número de 16 bits que indica cuántos bytes hay en el campo `RDATA`.

- El **campo `RDATA`** contiene los datos del registro, y su formato depende del valor del campo `TYPE`:

– En los registros `A` (de la clase Internet), es un número de 32 bits.

– En los registros `CNAME`, `NS`, `PTR`, `MB`, `MG` y `MR`, es un nombre de dominio.

– En los registros `SOA`, es una secuencia de dos nombres de dominio (`MNAME` y `RNAME`) seguida de cinco números de 32 bits (`SERIAL`, `REFRESH`, `RETRY`, `EXPIRE` y `MINIMUM`).

– En los registros `MX`, es un entero de 16 bits (la preferencia) seguido de un nombre de dominio.

– En los registros `HINFO`, es la concatenación de dos cadenas de caracteres. La primera indica el tipo de UCP del ordenador, y la segunda, el sistema operativo.

– En los registros `WKS`, es un número de 32 bits, como en los registros `A`, seguido de un número de 8 bits que identifica un protocolo* y una secuencia de bits en la que cada bit indica si el ordenador ofrece un determinado servicio o no.

*** Por ejemplo, *6* = TCP, 17 = UDP.**

– En los registros `MINFO`, es la concatenación de dos nombres de dominio. El primero constituye la dirección del responsable del buzón o lista de distribución, y el segundo, la dirección donde deben enviarse las notificaciones de error.

– En los registros `TXT`, es una cadena de caracteres.

– En los registros `NULL`, puede ser cualquier cosa.

2.5. Implementaciones del DNS

Prácticamente todos los ordenadores conectados a la red Internet incorporan alguna implementación de un cliente DNS (por norma general, una librería

de funciones llamadas por las aplicaciones) para poder acceder a otros ordenadores conociendo los nombres de los mismos. Éste es el caso, por ejemplo, de los sistemas UNIX, que proporcionan funciones como `gethostbyname` o `gethostbyaddr`.

En muchos sistemas UNIX también se encuentra disponible una utilidad denominada `nslookup`, que sirve para efectuar consultas directamente a un servidor DNS. En el modo de funcionamiento básico, acepta comandos como los siguientes:

- *`nombre`*: envía una consulta utilizando el nombre especificado como valor del campo `QNAME` y muestra los resultados obtenidos, indicando si la respuesta es sin autoridad. En caso de serlo, informa de qué registros ha enviado el servidor en la sección de autoridad de la respuesta.

- `server` *`servidor`*: cambia el servidor por defecto al que se envían las consultas.

- `ls` *`dominio`*: muestra una lista de nodos del dominio e información asociada; si la lista es muy larga, se puede guardar en un fichero añadiendo `> fichero`.

- `set querytype=`*`tipo`*: permite cambiar el campo `QTYPE` de las consultas que, por defecto, es `A`, a cualquier otro valor (`NS`, `MX`, `PTR`, `CNAME`, etc.).

- `set q=`*`tipo`*: es equivalente a `set querytype`.

- `set domain=`*`origen`*: cambia el dominio origen que debe utilizarse cuando se especifica un nombre relativo.

- `set` *`opción`*: cambia el valor de alguna opción utilizada por el programa. Si el valor es `recurse`, pide que las consultas se lleven a cabo en modo recursivo. Si el valor es `norecurse`, pide que se realicen en modo no recursivo. Si el valor es `vc`, pide que se utilicen "circuitos virtuales", es decir, el TCP. Si el valor es `novc`, pide que se empleen datagramas, o lo que es lo mismo, el UDP, etc.

- `set all`: muestra el valor actual de los parámetros con que trabaja el programa (servidor por defecto, opciones, lista de nodos origen que debe utilizarse cuando se consultan nombres de dominio relativos, etc.).

- `?`: muestra un mensaje de ayuda.

- `exit`: acaba el programa.

3. Servicios básicos de Internet

3.1. Terminal virtual: el protocolo Telnet

Uno de los servicios básicos para los que se puede utilizar una red de comunicaciones entre computadores es el acceso por medio de un terminal. Cuando se desarrollaron los sistemas multiusuario, la manera habitual de trabajar en los mismos consistía en utilizar un equipo terminal (en un inicio, un teletipo y, más adelante, un dispositivo con teclado y pantalla) conectado directamente al ordenador, en general por medio de una línea serie. El ordenador debía saber qué terminales tenía conectados y cómo podía controlarlos.

Teletipos

El hecho de que en una primera época se utilizaran teletipos es el motivo por el cual en UNIX se utilizó la abreviación `tty` (*teletype*) para designar los terminales (básicamente, impresoras con teclado).

Con la llegada de las redes de comunicaciones, que permitían interconectar diferentes ordenadores, uno de sus usos naturales era mantener una sesión de trabajo interactiva con un ordenador remoto sin necesidad de utilizar los terminales que estuvieran conectados al mismo directamente. Para ello, era preciso establecer un protocolo que, por un lado, hiciera posible que el ordenador remoto y el sistema en que quería trabajar el usuario se pudieran comunicar y se entendieran (es decir, un protocolo en el ámbito de aplicación, puesto que el problema del transporte se supone que ya estaba resuelto) y, por otro lado, facilitara el control del terminal del usuario desde el ordenador remoto. Un ordenador puede permitir que se acceda al mismo desde cualquier terminal de cualquier otro ordenador conectado a la red; sin embargo, no es práctico que deba saber cómo se controlan todos los tipos posibles de terminal. La solución general a estos requisitos se basa en el uso de un protocolo de terminal virtual.

Un **terminal virtual** es un dispositivo imaginario para el que se definen unas funciones de control canónicas, de manera que se puede establecer una correspondencia entre ellas y las de cada tipo de terminal real.

Lectura complementaria

Si queréis más información sobre el protocolo Telnet, consultad la obra siguiente:

J. Postel; J.K. Reynolds (1983, 1 de mayo). *RFC 854 - Telnet Protocol Specification*.

En el entorno Internet, el protocolo de terminal virtual más utilizado es **Telnet**, cuya especificación se publicó en 1983 en el estándar RFC 854.

3.1.1. Principios básicos del protocolo Telnet

El protocolo Telnet está basado en el protocolo de transporte TCP. En una comunicación Telnet, por norma general se sigue el modelo cliente/servidor; es decir, el sistema usuario establece una conexión con el sistema proveedor, que está esperando peticiones de conexión en un puerto determinado. Se puede utilizar cualquier número de puerto para las conexiones y, de hecho, existen

muchas aplicaciones que utilizan el protocolo Telnet para la comunicación, cada una con su propio número. La aplicación básica, sin embargo, consiste en establecer una sesión de trabajo interactiva con el sistema servidor y, en este caso, el número de puerto utilizado es el 23.

Para resolver el problema del control del terminal en la aplicación de sesiones interactivas, en el protocolo Telnet se utiliza el concepto de **terminal virtual de red** o **NVT**. Un NVT es un terminal virtual con una funcionalidad muy básica, definida en la misma especificación del protocolo.

Cuando se establece la conexión entre el cliente y el servidor, en un inicio se supone que la comunicación se produce entre dos NVT. Ello significa que tanto el sistema cliente como el sistema servidor deben mapear sus características en las de un NVT y suponer que en el otro extremo de la conexión hay otro NVT. En el modo de operación normal, cada terminal acepta datos del usuario y los envía por medio de la conexión establecida en el otro terminal, así como acepta los datos que llegan por la conexión y se los presenta al usuario.

Los usuarios

El usuario del terminal puede ser una persona o un proceso. Los datos leídos en el primer caso por norma general serán los caracteres tecleados y, en el segundo, los que vaya generando el proceso. La presentación de los datos recibidos puede consistir en escribirlos en la pantalla, enviarlos a la impresora o pasarlos al proceso usuario.

Las características principales de los NVT son las siguientes:

a) Los datos se intercambian en bytes de 8 bits. Cuando representan caracteres, se utiliza la codificación ASCII (por tanto, se envía un código de 7 bits en cada byte).

b) Existen tres caracteres de control que el NVT siempre los interpreta de la manera siguiente:

- `NUL` (carácter de control nulo, código 0): no es necesario hacer nada.
- `LF` (avance de línea, código 10): mover la posición de escritura actual* una línea adelante manteniendo la posición horizontal.

*** En una impresora, el carro, y en una pantalla, el cursor.**

- `CR` (retorno de carro, código 13): mover la posición de escritura atrás hasta el principio de la línea actual.

  Por tanto, un final de línea (es decir, un salto al principio de la línea siguiente) se podría representar con las secuencias `CR-LF` o `LF-CR`. Sin embargo, para facilitar el mapeo en terminales que no tienen funciones separadas para el avance de línea y el retorno de carro, el protocolo Telnet especifica que los finales de línea deben representarse con la secuencia `CR-LF`, y los retornos de carro sin mover la posición vertical, con la secuencia `CR-NUL`. De este modo, cuando el terminal recibe un carácter `CR`, puede esperar a que llegue el siguiente, que deberá ser `LF` o `NUL`, para saber qué acción debe llevar a cabo.

c) Existen cinco caracteres más que opcionalmente pueden ser interpretados por el NVT y cuyo significado es el siguiente:

- BEL (timbre, código 7): generar una señal que, por lo general, será audible, pero que alternativamente puede ser visible, sin mover la posición actual.
- BS (retroceso, código 8): mover la posición horizontal un espacio atrás.
- HT (tabulación horizontal, código 9): mover la posición horizontal hacia delante hasta la posición de tabulación siguiente.
- VT (tabulación vertical, código 11): mover la posición vertical hacia delante hasta la posición de tabulación vertical siguiente.
- FF (avance de página, código 12): mover la posición vertical hasta el principio de la página siguiente, sin mover la posición horizontal.

Cualquier otro carácter de control se considera equivalente a NUL.

d) El dispositivo NVT es *full duplex*; sin embargo, por norma general trabaja en modo *half duplex*; es decir, sólo envía los datos del usuario cuando ha leído una línea completa o cuando recibe alguna otra señal local que provoque la transmisión.

e) Si el usuario, de momento, no dispone de más datos para enviar, y se han procesado todos los datos recibidos, el terminal puede transmitir al sistema remoto una señal, denominada *Go Ahead* (GA), para indicarle que es su turno de enviar datos.

La señal GA

Sólo debería utilizarse la señal GA cuando el otro sistema no tuviera otra manera de saber que no debe esperar más datos. Una situación típica es que el usuario (cliente) genere líneas de una en una y el sistema remoto (servidor) envíe respuestas de cero, una o más líneas a cada una. En este caso, sólo será necesario que el servidor envíe la señal GA para notificar que el control pasa al usuario del terminal, y no será preciso que el cliente se lo envíe después de cada línea.

f) El NVT también puede generar, a instancias del usuario, dos señales más, denominadas *Break* (BRK) y *Synch*. El significado de la primera depende de la aplicación. La segunda sirve para cancelar los datos que haya enviado el terminal y todavía no hayan sido procesados por el sistema remoto.

g) Asimismo, existen cinco funciones de control que puede generar el NVT a instancias del usuario: *Interrupt Process* (IP), *Abort Output* (AO), *Are You There* (AYT), *Erase Character* (EC) y *Erase Line* (EL).

Veremos el significado de las funciones de control en el subapartado 3.1.2 de este módulo didáctico.

Si el cliente y el servidor soportan una funcionalidad más avanzada que la de un NVT, que es lo más habitual, el protocolo permite que lleven a cabo una negociación para ponerse de acuerdo sobre qué características diferentes de las de un NVT utilizarán en la comunicación.

La **negociación** consiste en intercambiar códigos que indican las opciones del protocolo que cada parte desea o está dispuesta a utilizar. Cuatro son los códigos que se utilizan para implementar la negociación:

- `DO` es una petición que se envía para pedir que se utilice una determinada opción.
- `WILL` es un ofrecimiento que se envía para indicar que el sistema está preparado para utilizar una determinada opción.
- `DON'T` significa que el sistema no quiere que se utilice la opción indicada.
- `WON'T` significa que el sistema no está preparado para utilizar la opción indicada.

El mecanismo de negociación es simétrico, de manera que cualquiera de las dos partes puede enviar un `DO` o un `WILL`. Tras enviar un `DO`, puede recibirse un `WILL` como respuesta positiva o un `WON'T` como respuesta negativa. Desde un punto de vista análogo, después de enviar un `WILL`, se puede recibir un `DO` como respuesta positiva o un `DON'T` como respuesta negativa. Ello significa que un `DO` puede actuar como petición o como confirmación si la otra parte ha enviado un `WILL`, y viceversa.

Por norma general, la negociación de las opciones se produce al inicio de la conexión, pero también es posible efectuarla en cualquier otro momento. Cada sistema debe empezar a utilizar una opción cuando envía o recibe la respuesta positiva correspondiente.

Este mecanismo de negociación está diseñado para que sea fácilmente extensible. Si un sistema no reconoce una determinada opción que se le pide o se le ofrece, simplemente debe contestar con `WON'T` o `DON'T`.

En ocasiones se necesita más información para negociar una opción, como el valor de algún parámetro. En este caso, en primer lugar deben ponerse de acuerdo las dos partes con el mecanismo básico de los `DO/DON'T/WILL/WON'T` para utilizar la opción deseada y, una vez se haya aceptado, debe llevarse a cabo una **subnegociación** de los valores de los parámetros utilizando una sintaxis específica de cada opción.

Muchas opciones definen sus comandos para la subnegociación, como `IS` (para enviar el valor de un parámetro), `SEND` (para pedir que la otra parte lo envíe), etc.

Las **opciones Telnet** posibles no están definidas en la especificación del protocolo, sino en especificaciones independientes. A continuación, se presentan algunas de las opciones que se pueden negociar y los códigos correspondientes:

- `TRANSMIT-BINARY` (RFC 856): código 0. Los datos que envíe la parte que tenga activada esta opción deben interpretarse como bytes de 8 bits.

- `ECHO` (RFC 857): código 1. La parte que tenga activada esta opción reenviará a la otra los caracteres que reciba.

- `SUPPRESS-GO-AHEAD` (RFC 858): código 3. La parte que tenga activada esta opción suprimirá la transmisión de la señal `GA`.

- `STATUS` (RFC 859): código 5. La parte que tiene activada esta opción puede pedir a la otra cuál es el estado actual de las opciones, o cuál creéis que es, para evitar bucles en las negociaciones.

- `TERMINAL-TYPE` (RFC 1091): código 24. Esta opción permite informar al sistema remoto del tipo de terminal real que utiliza el usuario.

  Emulaciones

  En caso de que el servidor no reconozca un tipo de terminal que se le envía con la opción `TERMINAL-TYPE`, puede pedir que el cliente le envíe otro, por si soporta la emulación de diferentes tipos. Este proceso se puede repetir hasta que se pongan de acuerdo o hasta que al cliente no le quede ningún tipo más de terminal soportado. Si el servidor reconoce este tipo de terminal, podrá controlarlo directamente sin necesidad de convertir las funciones de control a la representación canónica del NVT.

- `NAWS` (*Negotiate About Window Size*, RFC 1073): código 31. El cliente informa al servidor de los tamaños de la ventana de visualización del terminal con dos números de 16 bits: el número de caracteres en cada fila horizontal y el número total de filas.

- `TERMINAL-SPEED` (RFC 1079): código 32. El cliente indica al servidor la velocidad de transmisión y la de recepción del terminal, en bits por segundo, representadas como una cadena de caracteres ASCII con los dos números en decimal separados por una coma.

- `TOGGLE-FLOW-CONTROL` (RFC 1372): código 33. Permite habilitar o inhabilitar el uso de los caracteres de control "`^S`" y "`^Q`", que sirven para suspender y retomar la transmisión de datos al terminal, respectivamente.

- `LINEMODE` (RFC 1184): código 34. Permite negociar los caracteres de control que deben usarse localmente en el terminal para editar líneas o para generar determinadas señales.

- X-DISPLAY-LOCATION (RFC 1096): código 35. Si el usuario trabaja en el sistema de ventanas X, con esta opción puede informar al servidor del nombre de su dispositivo lógico de visualización.

- AUTHENTICATION (RFC 1416): código 37. Esta opción permite intercambiar información de autenticación del usuario dentro del mismo protocolo Telnet en lugar de usar el método habitual, es decir, que la aplicación requiera un nombre y una contraseña. De este modo, se pueden implementar mecanismos de protección más seguros; por ejemplo, enviar la contraseña cifrada, etc.

- NEW-ENVIRON (RFC 1572): código 39. Esta opción es una versión actualizada de la opción llamada ENVIRON y permite enviar los valores de ciertas variables del entorno del cliente que se pueden utilizar en el sistema remoto.

Todas estas opciones, excepto las tres primeras, utilizan el mecanismo de subnegociación.

3.1.2. Comandos del protocolo Telnet

En los datos intercambiados entre dos sistemas que se comunican con el protocolo Telnet, se pueden intercalar ciertos comandos propios del protocolo, tales como la señal GA o los códigos de negociación. Para distinguir los comandos de los datos normales, los primeros deben ir prefijados con un código especial llamado IAC (Interpret As Command), que se representa con un byte igual a 255.

Por consiguiente, cada comando se representa con una secuencia de dos bytes, en la que el primero es el código IAC, y el segundo, el código propio del comando, excepto los comandos de negociación, que disponen de un tercer byte que sirve para indicar a qué opción se refieren. Para representar un byte normal de datos que sea igual a 255, es preciso prefijarlo con un código IAC. Cualquier otro byte de datos se representa directamente con su código.

Los comandos definidos en el protocolo Telnet son los siguientes:

- NOP (*No Operation*, código 241): operación nula.

- GA (*Go Ahead*, código 249): señal GA.

- BRK (*Break*, código 243): señal BRK.

- DO (código 253) + código opción: código de negociación DO.

- DON'T (código 254) + código opción: código de negociación DON'T.

- `WILL` (código 251) + código opción: código de negociación `WILL`.

- `WON'T` (código 252) + código opción: código de negociación `WON'T`.

- `SB` (*Subnegotiation Begin*, código 250) + código opción: los datos que vengan a continuación corresponden a la subnegociación de una opción.

- `SE` (*Subnegotiation End*, código 240): indica el fin de los datos de subnegociación.

- `DM` (*Data Mark*, código 242): indica en qué punto de la secuencia de datos se ha enviado una señal *Synch*.

- `IP` (*Interrupt Process*, código 244): código de función que puede enviar el usuario para indicar al sistema remoto que interrumpa el proceso que está ejecutando.

- `AO` (*Abort Output*, código 245): código de función que sirve para indicar que se continúe ejecutando el proceso remoto, pero que deje de enviar al usuario los datos que genera.

- `AYT` (*Are You There*, código 246): código de función que pide al sistema que, cuando lo reciba, conteste con algún mensaje que sirva al usuario para verificar que la conexión continúa establecida (por ejemplo, cuando el proceso que ejecuta tarda en generar una respuesta).

- `EC` (*Erase Character*, código 247): código de función que sirve para borrar el último carácter escrito en el terminal.

- `EL` (*Erase Line*, código 248): código de función que sirve para borrar toda la línea actual.

El protocolo Telnet también proporciona un mecanismo para transmitir la señal *Synch*. Esta última no se representa con un comando como los otros, dado que debe tener un efecto inmediato. En este caso, se envían datos urgentes TCP acabados con un código DM y, en la secuencia de datos normales, se inserta otro código DM. Al recibir los datos urgentes, el sistema remoto pasará a procesar los datos normales en modo urgente (que puede consistir en descartar todo lo que haya en estos datos, salvo los comandos especiales) hasta que encuentre el DM.

El terminal puede generar automáticamente una señal *Synch* cuando el usuario envía alguna de las funciones que requieren atención inmediata, como `IP`, `AO` o `AYT` (pero no `EC` o `EL`).

3.1.3. Implementaciones del protocolo Telnet

Hay implementaciones de servidores Telnet prácticamente en todos los sistemas multiusuario que utilizan el protocolo TCP. Los clientes Telnet son más numerosos todavía, porque también hay para sistemas monousuario.

Un ejemplo de implementación de cliente Telnet es la utilidad del sistema operativo UNIX denominada precisamente `telnet`. Si se llama sin argumentos, entra en modo comando. Con un argumento, que puede ser una dirección IP o un nombre de servidor, establece una conexión con el puerto Telnet (el 23) de este servidor y entra en modo conexión. Con un segundo argumento, que puede ser un número de puerto o un nombre de servicio, establece la conexión con este puerto.

Cuando el número de puerto utilizado es el 23, el cliente inicia automáticamente el proceso de negociación enviando los códigos `DO` y `WILL` correspondientes a las opciones que soporta. Con cualquier otro puerto, por norma general no se envía ningún código de negociación, salvo que se reciba alguno del sistema remoto.

Cuando el programa está en modo conexión, envía al sistema remoto cada carácter que teclea el usuario. Desde el modo conexión se puede pasar al modo comando por medio del carácter de escape, que suele ser "`^]`". En este modo, el programa admite, entre otros, los comandos siguientes*:

*** Los nombres se pueden abreviar siempre que no generen ambigüedades.**

- `open`: establece una conexión con el servidor, y opcionalmente con el puerto, indicado en los argumentos de este comando.

- comando nulo (línea vacía): si hay una conexión establecida, sale del modo comando y vuelve al modo conexión.

- `send`: envía al sistema remoto el código Telnet indicado por el argumento, que puede ser `ao`, `ayt`, `brk`, `ec`, `el`, `ga`, `ip`, `synch`, etc.

- "`^]`" (o el carácter que actúe como carácter de escape): envía este último al sistema remoto (equivale a `send escape`).

- `?`: muestra un mensaje de ayuda.

- `quit`: cierra la conexión, en caso de haber conexión establecida, y acaba el programa (que también acaba cuando el sistema remoto cierra la conexión).

3.1.4. Ejemplo de intercambio de datos

El ejemplo siguiente muestra un posible intercambio de datos entre un cliente y un servidor Telnet justo después de establecer la conexión:

```
> FF FD 03          DO SUPPRESS-GO-AHEAD
> FF FB 18          WILL TERMINAL-TYPE
> FF FB 1F          WILL NAWS
> FF FB 20          WILL TERMINAL-SPEED
> FF FB 21          WILL TOGGLE-FLOW-CONTROL
> FF FB 22          WILL LINEMODE
> FF FB 27          WILL NEW-ENVIRON
> FF FD 05          DO STATUS
< FF FD 18          DO TERMINAL-TYPE
< FF FD 20          DO TERMINAL-SPEED
< FF FD 23          DO X-DISPLAY-LOCATION
< FF FD 27          DO NEW-ENVIRON
> FF FC 23          WON'T X-DISPLAY-LOCATION
< FF FB 03          WILL SUPPRESS-GO-AHEAD
< FF FD 1F          DO NAWS
< FF FD 21          DO TOGGLE-FLOW-CONTROL
< FF FE 22          DON'T LINEMODE
< FF FB 05          WILL STATUS
> FF FA 1F          SB NAWS 80 25 SE
> 00 50 00 19 FF F0
< FF FA 20 01 FF F0  SB TERMINAL-SPEED SEND SE
< FF FA 27 01 FF F0  SB NEW-ENVIRON SEND SE
< FF FA 18 01 FF F0  SB TERMINAL-TYPE SEND SE
> FF FA 20 00 33 38  SB TERMINAL-SPEED IS "38400,38400" SE
> 34 30 30 2C 33 38
> 34 30 30 FF F0
> FF FA 27 00 FF F0  SB NEW-ENVIRON IS "" SE
> FF FA 18 00 4C 49  SB TERMINAL-TYPE IS "LINUX" SE
> 4E 55 58 FF F0
< FF FD 01          DO ECHO
> FF FC 01          WON'T ECHO
< FF FB 01          WILL ECHO
< 42 65 6E 76 69 ... "Bienvenido al servidor..."
> FF FD 01           DO ECHO
```

Nota

En las líneas que empiezan con ">", se encuentran los datos, en hexadecimal, que el cliente envía al servidor y, en los que empiezan con "<", los que el servidor envía al cliente. En la derecha indicamos el significado de estos datos.

3.2. Terminal virtual en UNIX: el protocolo rlogin

Cuando se desea establecer una sesión de trabajo interactiva desde un sistema UNIX con otro sistema UNIX, además de utilizar el protocolo Telnet, también

existe la posibilidad de utilizar el protocolo llamado **rlogin**. Este último se desarrolló en un inicio en el entorno de las utilidades de comunicaciones de la variante de UNIX llamada BSD; sin embargo, hoy día hay implementaciones disponibles en todas las variantes, así como en otros sistemas operativos. La especificación del protocolo rlogin está publicada en el documento *RFC 1282*.

Las características principales que diferencian el protocolo rlogin del protocolo Telnet son las siguientes:

Lectura complementaria

Para obtener más información sobre el protocolo rlogin, consultad la obra siguiente:

B. Kantor (1991, diciembre). *RFC 1282 - BSD Rlogin*.

a) El servidor siempre es informado del tipo de terminal con que trabaja el cliente sin necesidad de negociación.

b) El servidor también es informado de la identidad del usuario en el sistema cliente, lo que se puede utilizar para automatizar el proceso de autenticación.

c) Si cambian los tamaños de la ventana de presentación del terminal, se puede enviar automáticamente una señal al servidor para notificárselo.

d) En la transmisión de datos, siempre pueden utilizarse los 8 bits de cada byte.

3.2.1. Conceptos básicos del protocolo rlogin

El protocolo rlogin utiliza conexiones TCP. El nombre oficial del servicio proporcionado por este protocolo es `login` y el número de puerto asignado a este servicio es el 513.

Cuando el cliente establece la conexión con el servidor, en primer lugar le envía cuatro cadenas de caracteres, cada una acabada con un carácter `NUL` (código 0). Son las cadenas siguientes:

- Una cadena vacía (sólo contiene el carácter `NUL`).
- El nombre del usuario en el sistema cliente.
- El nombre de usuario con que se quiere establecer la sesión de trabajo en el sistema servidor.
- El tipo de terminal y, separado con "`/`", su velocidad*.

*** Por ejemplo, vt100/9600.**

Cuando ha recibido estas cuatro cadenas, el servidor envía un carácter `NUL` y empieza la transferencia de datos entre cliente y servidor en modo control de flujo. En este modo, el cliente envía los caracteres que teclea el usuario tal como le llegan, excepto los caracteres de control DC3 ("`^S`") y DC1 ("`^Q`"), que significan 'suspender la transmisión' y 'retomarla', respectivamente.

Por otro lado, el cliente presenta los datos que envía el servidor tal como le llegan. El cliente también puede recibir mensajes de control del servidor, que se representan con un código de un byte enviado en datos urgentes TCP. El cliente debe responder a estos mensajes de manera inmediata y suspender temporalmente el procesado de otros datos que pueda haber recibido.

Los significados de los mensajes de control que puede recibir el cliente son los siguientes:

- Descartar los datos recibidos hasta el momento (código 2).
- Pasar a modo transparente, es decir, transmitir todos los caracteres, incluyendo los DC1 y los DC3 (código 16).
- Volver a modo control de flujo (código 32).
- Enviar los tamaños actuales de la ventana del terminal (código 128).

Si el servidor soporta la notificación de cambios en los tamaños de la ventana del terminal, se lo hace saber al cliente enviándole un mensaje con código 128 justo después de la fase inicial de identificación del usuario. El cliente debe contestar con una secuencia especial de 12 bytes, en la que todos los números (filas y columnas de caracteres, y tamaños horizontal y vertical de la ventana en píxeles) se representan con 2 bytes en el orden de la red. El formato de esta secuencia es el siguiente:

Nota

Recordad que el orden de la red consiste en transmitir primero el byte de más peso.

Formato de la secuencia de respuesta de cambios de los tamaños de la ventana

Durante la conexión, el cliente puede enviar esta misma secuencia cada vez que se produzca un cambio en los tamaños de la ventana.

3.2.2. Implementación del protocolo rlogin

En los sistemas UNIX hay una utilidad llamada `rlogin` que actúa como cliente del protocolo rlogin. Es preciso que se le especifique el nombre del servidor remoto y, de manera optativa, el nombre de usuario que debe utilizar en este servidor, con la opción `-l` (por defecto, se utiliza el mismo nombre de usuario que en el sistema local).

La mayoría de los servidores rlogin implementan la autenticación automática de la manera siguiente:

Si el usuario remoto tiene en su directorio un fichero llamado `.rhosts` con una línea en la que se encuentre el nombre del ordenador cliente y, opcionalmente, el nombre del usuario cliente (en ausencia de este nombre de usuario, se toma el del usuario en el servidor), se inicia directamente la sesión interactiva sin necesidad de pedir ninguna contraseña.

Si no dispone de este fichero, se consulta otro, el `/etc/hosts.equiv`, para ver si existe alguna línea con el nombre del ordenador cliente (en este caso, debe coincidir el nombre del usuario en el cliente y en el servidor) y, si está, también se da al usuario por autenticado. En cualquier otro caso, se sigue el procedimiento normal de autenticación mediante contraseña.

Seguridad en el protocolo rlogin

Por motivos de seguridad, los servidores rlogin suelen requerir que el fichero `.rhosts` tenga permiso de lectura sólo para el usuario a quien pertenece. Por otro lado, los nombres de servidores en los ficheros `.rhosts` y `/etc/hosts.equiv` no pueden ser alias, deben ser los nombres oficiales. Y, como precaución adicional, el fichero `/etc/hosts.equiv` sólo se utiliza cuando el nombre del usuario en el servidor es diferente de `root`.

Antes de consultar los ficheros `.rhosts` y `/etc/hosts.equiv`, el servidor siempre comprueba que el cliente se haya conectado desde un puerto reservado (con número inferior a 1024). En caso contrario, no se permite la autenticación automática, puesto que la conexión podría provenir de otro usuario que estuviera ejecutando un programa que siguiese el protocolo rlogin, pero que enviase información falsa sobre su identidad.

La autenticación automática puede resultar cómoda porque ahorra teclear las contraseñas, pero también puede ser una fuente de problemas si no se utiliza con precaución. 

Además de leer los caracteres tecleados y enviárselos al servidor, el cliente también puede recibir comandos del usuario. Para distinguirlos de los caracteres normales, se representan con un carácter precedido del símbolo "~". Esta secuencia sólo se reconoce al inicio de las líneas (es decir, inmediatamente a continuación de un salto de línea). El carácter del comando puede ser, por ejemplo, el siguiente:

- "^Z" (o el carácter correspondiente a la función SUSP del terminal): suspende la ejecución del programa cliente, con la posibilidad de retomarla más adelante (la manera de hacerlo puede depender del *shell*, pero por lo general es con el comando `fg`).

- "^Y" (o el carácter correspondiente a la función DSUSP del terminal): suspende el envío de datos al servidor y, por consiguiente, el teclado queda

disponible para otros procesos; sin embargo, la recepción continúa (si se reciben nuevos datos, se mostrarán cuando lleguen).

- "~": envía el carácter ~.

- " . ": cierra la conexión.

Si el carácter no corresponde a ningún comando reconocido por el cliente, se envía al servidor tal como ha llegado, precedido por el carácter "~".

3.3. Otros servicios

3.3.1. Ejecución remota con autenticación automática: rsh

Paralelamente al servicio de terminal virtual, que permite mantener una sesión de trabajo en un sistema remoto, en los sistemas UNIX BSD se desarrolló otro servicio que permite ejecutar un comando en el sistema remoto. El programa que actúa como cliente de este servicio se llama **rsh***.

*** rsh corresponde a la expresión inglesa *remote shell*.**

El nombre oficial de este servicio es `shell`, el protocolo de transporte utilizado es el TCP y el número de puerto asignado es el 514.

A continuación presentamos los pasos que se siguen en este protocolo:

1) El servidor comprueba que el cliente se haya conectado desde un puerto reservado. Si no es el caso, cierra la conexión.

2) El cliente envía al servidor una cadena acabada en `NUL` que contiene un número decimal:

- Si es diferente de cero, se interpreta como un número de puerto y el servidor establece en este puerto una conexión secundaria (esta segunda conexión se establece también desde un puerto reservado).

- Si es cero, la misma conexión principal actúa también como secundaria.

3) El cliente envía tres cadenas más acabadas en `NUL` con el nombre del usuario en el sistema servidor, el nombre en el sistema cliente y el comando a ejecutar.

4) El servidor comprueba en los ficheros `.rhosts` y/o `/etc/hosts.equiv` que el usuario esté autorizado para la autenticación automática.

5) Si lo está, el servidor envía un carácter `NUL` por la conexión secundaria; en caso contrario, cierra las conexiones.

6) El servidor ejecuta el comando indicado con la identidad del usuario indicado. Los datos que escriba este comando por la salida estándar (`stdout` en la nomenclatura del lenguaje C) se enviarán por la conexión principal, y los que escriba por la salida de error (`stderr`), por la conexión secundaria. Las que envíe el cliente serán accesibles por la entrada estándar (`stdin`).

7) Cuando acabe la ejecución del comando, el servidor cerrará la conexión.

3.3.2. Ejecución remota: rexec

Con rsh, el usuario debe estar autorizado por medio de los ficheros `.rhosts` o `/etc/hosts.equiv` del sistema remoto para poder ejecutar un comando en el mismo. Existe otro servicio muy parecido en que no se utiliza la autenticación automática, sino que siempre requiere que el usuario proporcione su contraseña en el sistema remoto. En algunos sistemas UNIX, se encuentra disponible un programa llamado ***rexec*** que actúa como cliente de este servicio. En otros, sólo hay una función de la librería estándar, *rexec*, que implementa el protocolo.

El nombre oficial de este servicio es `exec`, el protocolo de transporte utilizado es el TCP y el número de puerto asignado es el 512.

Este protocolo es muy parecido al anterior: las únicas diferencias son que, en lugar de enviar el nombre del usuario, en el sistema cliente se envía la contraseña del usuario en el sistema servidor (los ficheros `.rhosts` y `/etc/hosts.equiv` no se consultan) y que no es preciso verificar que las conexiones provengan de puertos reservados.

3.3.3. Servicios triviales

La mayoría de los sistemas UNIX proporcionan una serie de servicios denominados *triviales*, que se pueden utilizar para llevar a cabo pruebas de funcionamiento de los módulos que implementan los protocolos de transporte. Todos son accesibles por medio de TCP y UDP, y el número de puerto utilizado en ambos casos es el mismo. Algunos ejemplos de servicios triviales son los siguientes:

- `echo` (puerto 7): retorna todos los bytes (en TCP) o datagramas (en UDP) que recibe (RFC 862).

- `discard` (puerto 9): lee datos (bytes o datagramas), pero no hace nada más (RFC 863).

- `chargen` (puerto 19): en TCP, cuando se establece la conexión, el servidor empieza a enviar una secuencia de caracteres hasta que el cliente la cierra y, en UDP, cuando el servidor recibe un datagrama, responde con otro que contiene un nombre arbitrario de caracteres (RFC 864).

- `daytime` (puerto 13): cuando se establece la conexión, o se recibe un datagrama, el servidor envía la fecha y la hora actual en formato legible para los humanos (RFC 867).

- `time` (puerto 37): cuando se establece la conexión, o se recibe un datagrama, el servidor envía un número de 4 bytes que representa el número de segundos transcurridos desde el 1 de enero de 1970 a las 0 horas GMT (RFC 868).

Otro ejemplo de servicio sencillo es el que permite obtener información de un usuario en un sistema remoto. La especificación del protocolo para este servicio, llamado `Name/Finger`, está recogida en el documento *RFC 742*.

El nombre oficial de este servicio es `finger`, el protocolo de transporte utilizado es el TCP y el número de puerto asignado es el 79.

El programa cliente en UNIX se llama *finger* y, por norma general, admite argumentos de la forma `usuario`, `usuario@host` o `@host`:

- En el primer caso, proporciona información sobre un usuario en el sistema local sin necesidad de utilizar ningún protocolo.

- En el segundo, informa sobre un usuario de un sistema remoto.

- En el tercero, normalmente ofrece una información resumida de los usuarios que están actualmente conectados con sesiones interactivas al sistema remoto.

Cuando se establece la conexión, el cliente simplemente envía una línea al servidor acabada con `CR-LF`. En esta última se encuentra el nombre del usuario de quien se quiere obtener información. Si no hay ningún nombre, se entiende que la solicitud se refiere a todos los usuarios que estén conectados.

El formato de la respuesta depende de la implementación del servidor y puede incluir información como el nombre del usuario, su nombre real, cuándo se conectó por última vez, cuándo recibió o leyó el correo por última vez, etc., y, si está conectado actualmente, desde dónde, en qué terminal, cuánto tiempo hace que no hay actividad en su terminal, etc. La información retornada normalmente es un subconjunto de todos estos campos; sin embargo, si la línea de petición empieza con los caracteres `"/W"`, el servidor puede retornar información más completa.

4. Transferencia de ficheros

4.1. FTP: protocolo de transferencia de ficheros

Una de las primeras aplicaciones básicas desarrolladas en el entorno de lo que después sería la red Internet fue la transferencia de ficheros entre diferentes sistemas. Al principio de los años setenta ya se elaboraron las primeras especificaciones del protocolo más utilizado para esta finalidad, el **FTP***. Después de algunas modificaciones y mejoras, la especificación oficial del protocolo se publicó en 1985 en el documento *RFC 959*.

*** FTP es la sigla de *file transfer protocol*.**

El FTP se basa en el modelo cliente/servidor y permite la transferencia de ficheros tanto del servidor al cliente, como del cliente al servidor. Asimismo, permite que un cliente efectúe transferencias directas de un servidor a otro, con lo que se ahorra la necesidad de copiar los ficheros del primer servidor al cliente y pasarlos después del cliente al segundo servidor.

Lectura complementaria

Para más información sobre el FTP, consultad la obra siguiente:

J. Postel; J. Reynolds (1985, octubre). *RFC 959 - File Transfer Protocol*.

El protocolo proporciona también operaciones para que el cliente pueda manipular el sistema de ficheros del servidor: borrar ficheros o cambiarles el nombre, crear y borrar directorios, listar sus contenidos, etc.

Uno de los objetivos principales de este protocolo consiste en permitir la interoperabilidad entre sistemas muy distintos, escondiendo los detalles de la estructura interna de los sistemas de ficheros locales y de la organización de los contenidos de los ficheros.

Características obsoletas del FTP

Algunos de los sistemas utilizados en la época de desarrollo del FTP actualmente están obsoletos; por ejemplo, los que utilizan palabras de 36 bits o códigos de caracteres incompatibles con el estándar ASCII o los que almacenan los ficheros en "páginas" no necesariamente contiguas. De hecho, muchas de las implementaciones actuales del protocolo no soportan estas características especiales previstas en la especificación.

4.1.1. El modelo del FTP

En el modelo general descrito en la especificación del FTP, tanto en el servidor como en el cliente hay dos entidades que intervienen en la transferencia de ficheros:

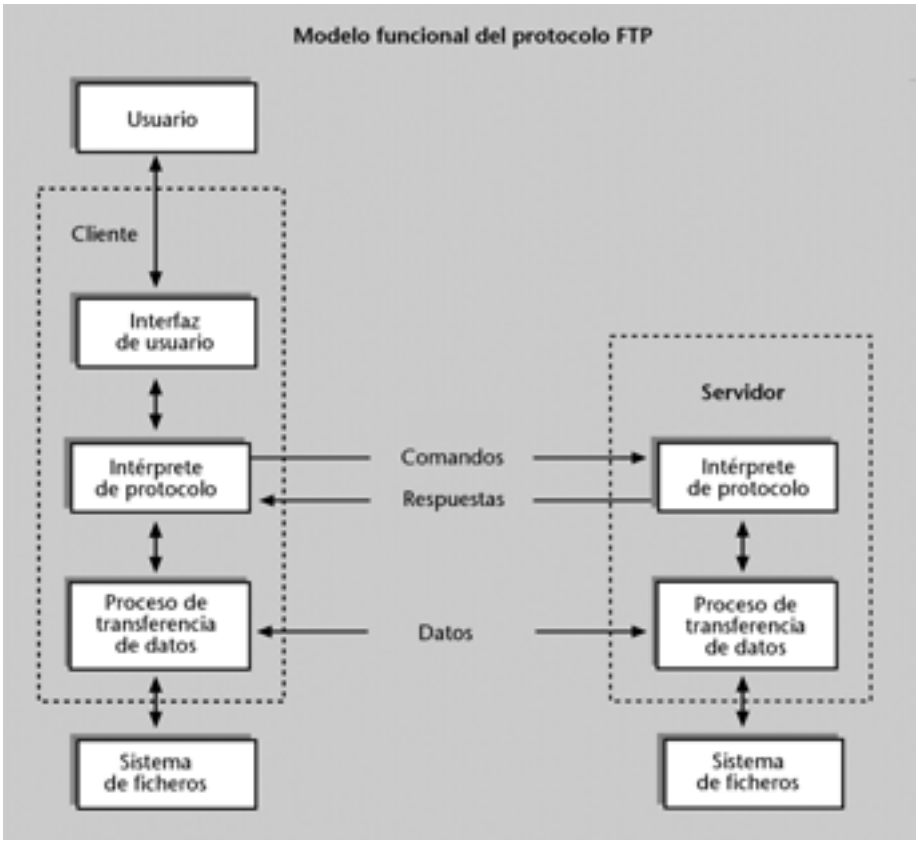
a) El **intérprete de protocolo** se encarga del intercambio de los comandos del protocolo. En la parte del cliente, las operaciones que el usuario solicita por

medio de la interfaz de usuario, el intérprete las convierte en una secuencia adecuada de comandos FTP y se envían al servidor.

En la parte del servidor se interpretan los comandos recibidos, se generan las respuestas correspondientes y se envían al cliente. Por tanto, el intérprete cliente debe analizar estas respuestas, por ejemplo, para informar al usuario del resultado de la operación o para proseguir la secuencia de comandos FTP.

b) El **proceso de transferencia de datos**, que está bajo el control del intérprete de protocolo, se encarga de intercambiar los datos que deben transferirse, es decir, los contenidos de los ficheros o bien los listados de los directorios.

Tanto en la parte del cliente, como en la del servidor este proceso interactúa directamente con el sistema de ficheros locales para leer sus datos o para almacenarlos en los mismos.

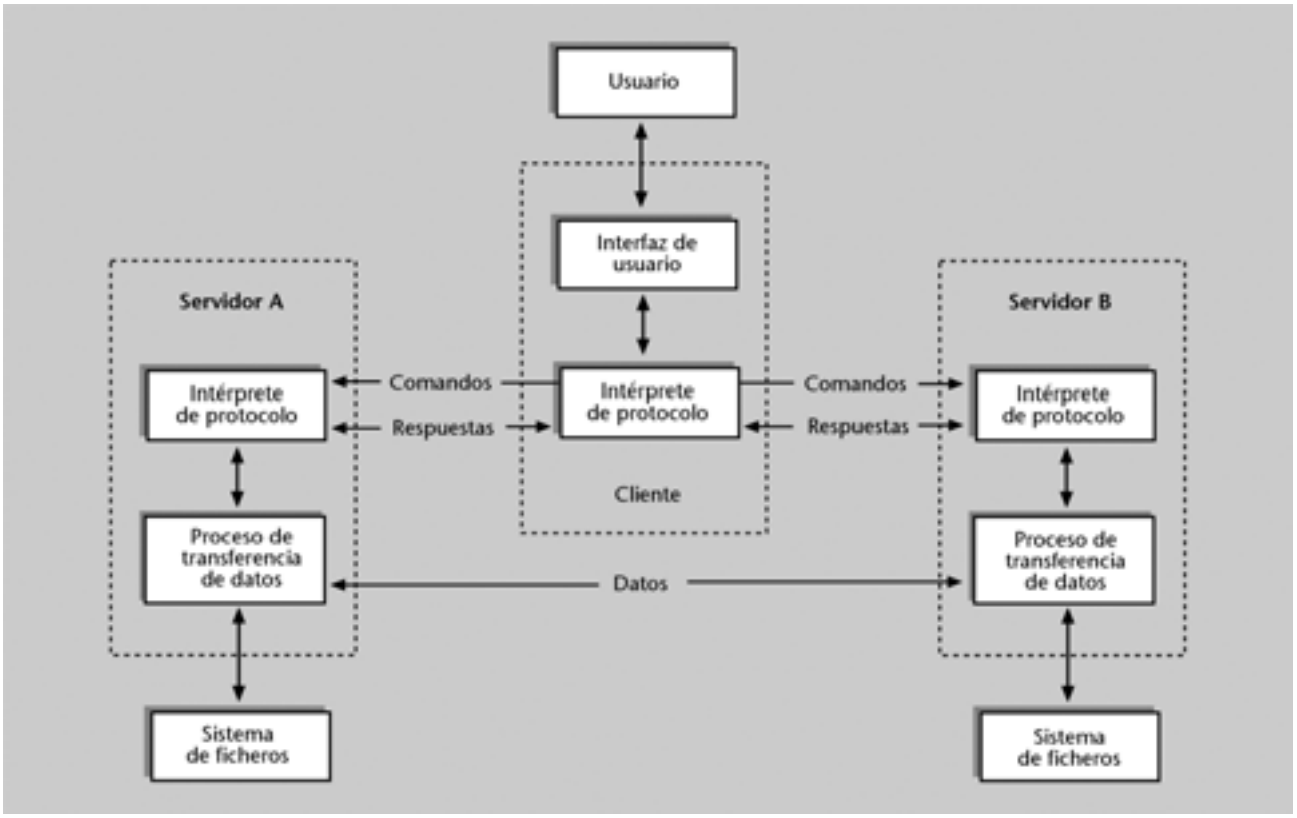


Los dos intérpretes de protocolo se comunican mediante una conexión TCP llamada *conexión de control*. Cuando deben transferirse datos de un sistema al otro, se establece otra conexión TCP entre los dos procesos de transferencia de datos denominada *conexión de datos*. Generalmente, la parte activa de esta conexión (la que la inicia) constituye el proceso de transferencia del servidor, y la parte pasiva, el del cliente.

Realización de diferentes transferencias por proceso

El FTP permite efectuar más de una transferencia en una única sesión. Según el modo de transmisión, es posible que se abran y se cierren diferentes conexiones de datos durante una misma conexión de control, o bien que se utilice la misma conexión de datos para las diferentes transferencias.

La figura siguiente muestra la variación del modelo general para el caso en que un cliente controle una transferencia entre dos servidores (uno de los cuales actuará como servidor activo y el otro, como servidor pasivo):



4.1.2. Conceptos básicos del FTP

El FTP está basado en conexiones TCP. El intérprete de protocolo del servidor debe estar preparado para recibir peticiones de conexión en un puerto TCP que, por defecto, es el asignado oficialmente al servicio FTP: el número 21.

El intérprete de protocolo del cliente establece una conexión de control con el puerto del intérprete servidor. En esta conexión se utilizan las reglas especificadas en el protocolo Telnet. Ello significa que, por defecto, los intérpretes de protocolo se intercambian mensajes codificados con bytes de 8 bits, según el juego de caracteres ASCII, y representan los finales de línea con la secuencia <CRLF>.

Consultad la descripción de las reglas especificadas por el protocolo Telnet (RFC 854) en el subapartado 3.1 de este módulo didáctico.

Los **comandos FTP** constituyen los mensajes que envía el intérprete cliente, y los que envía el intérprete servidor son respuestas a dichos comandos. Las respuestas se generan siguiendo el orden en que el cliente envía los comandos, puesto que en general el servidor efectúa las operaciones de manera secuencial (no empieza una nueva operación hasta que no ha acabado

Las excepciones...

... a la regla de la respuesta secuencial son los comandos obtener el estado actual de una transferencia, abortar una operación y cerrar la sesión.

la anterior). A continuación, analizamos por separado los comandos y las respuestas:

1) Comandos FTP: un comando FTP se representa por medio de un código de comando de hasta cuatro letras (que pueden ser indistintamente mayúsculas o minúsculas), seguido de una lista de cero o más argumentos, separados por espacios, acabada con un final de línea.

La especificación RFC 959 define treinta y tres comandos agrupados en tres categorías diferentes (representamos entre paréntesis el código correspondiente a cada comando):

Comandos no estándar

Algunas implementaciones de servidores FTP soportan otros códigos de comandos no estándares, que pueden tener más de cuatro letras.

a) Comandos de control de acceso: nombre de usuario (`USER`), contraseña (`PASS`), cuenta (`ACCT`), cambiar de directorio (`CWD`), cambiar al directorio padre (`CDUP`), montar un sistema de ficheros (`SMNT`), reinicializar (`REIN`), acabar la sesión (`QUIT`).

b) Comandos de parámetros de transferencia: estructura de fichero (`STRU`), modo de transmisión (`MODE`), tipo de representación (`TYPE`), puerto de datos (`PORT`), puerto pasivo (`PASV`).

c) Comandos de servicio FTP: obtener (`RETR`), almacenar (`STOR`), almacenar con nombre único (`STOU`), añadir (`APPE`), listar (`LIST`), listar nombres (`NLST`), mostrar el directorio actual (`PWD`), reservar espacio (`ALLO`), abortar (`ABOR`), retomar (`REST`), borrar (`DELE`), nombre antiguo (`RNFR`), nombre nuevo (`RNTO`), crear un directorio (`MKD`), borrar un directorio (`RMD`), estatus (`STAT`), sistema (`SYST`), servicios adicionales (`SITE`), ayuda (`HELP`) y operación nula (`NOOP`).

Comandos de transferencia

De este grupo de comandos de servicio, los seis siguientes son los comandos de transferencia; es decir, los que utilizan la conexión de datos.

- `RETR`
- `STOR`
- `STOU`
- `APPE`
- `LIST`
- `NLST`

2) Respuestas FTP: las respuestas constan de un código numérico de tres dígitos decimales, que permite al intérprete del protocolo cliente saber cuál es el resultado de la operación, seguido de un texto explicativo destinado al usuario humano.

El texto explicativo del mensaje de respuesta puede tener una línea o más:

- Si tiene una línea, la respuesta se representa con el código numérico seguido del carácter espacio y, a continuación, el texto y un final de línea.

- Si tiene más de una línea, la primera se representa con el código numérico seguido del carácter "`-`" y, a continuación, las líneas de texto, acabadas con finales de línea, la última de las cuales debe empezar con el mismo código numérico de la primera línea seguido del carácter espacio.

Ejemplo de respuesta

Ésta podría ser una respuesta de una sola línea:

```
220 Sistema preparado. Introducid nombre de usuario y contraseña.
```

Y esta otra, una respuesta con múltiples líneas:

```
220-Sistema preparado.
   Introducid el nombre de usuario y la contraseña para acceder
   a los ficheros restringidos, o utilizad el nombre "anonymous"
220 para acceder al directorio público.
```

La especificación RFC 959 define los códigos de respuesta posibles para cada uno de los comandos y su significado.

Códigos de respuesta	
Código	**Significado**
110	Marca de reanudación.
120	Esperar hasta que el servidor esté preparado.
125	El servidor transfiere datos (la conexión de datos ya está abierta).
150	El servidor establece la conexión de datos e inicia la transferencia.
200	Operación efectuada.
202	Comando innecesario en este servidor.
211	Información de estado del sistema o mensaje de ayuda del sistema.
212	Información de estado del directorio.
213	Información de estado del fichero.
214	Mensaje de ayuda (destinado al usuario humano).
215	Tipo de sistema.
220	Servidor preparado.
221	El servidor cierra la conexión de control.
226	Operación efectuada y conexión de datos cerrada.
227	El servidor está en modo pasivo (*h1,h2,h3,h4,p1,p2*).
230	Proceso de autenticación completado satisfactoriamente.
250	Operación de fichero efectuada.
257	El directorio es el resultado de la operación.
331	Enviar contraseña.
332	Enviar cuenta.
350	Enviar el comando siguiente para completar la operación.
421	Servicio no disponible (por ejemplo, por *time out*...), conexión de control cerrada.
425	No se puede establecer la conexión de datos.
426	Transferencia abortada y conexión de datos cerrada.
450	No se puede efectuar la operación sobre el fichero (por ejemplo, fichero ocupado).
452	Espacio de disco insuficiente: transferencia no iniciada.
500	Error de sintaxis en el comando.
501	Error en los argumentos.
502	Comando no implementado.

La estructura de los códigos de respuesta y el significado de cada dígito son comunes a todos los protocolos. Podéis encontrar una explicación de los mismos en el anexo 2 de este módulo didáctico.

Códigos de respuesta	
Código	**Significado**
503	Error en la secuencia de comandos (este comando no corresponde).
504	Argumento no soportado.
530	El usuario no se ha autenticado correctamente.
532	Se necesita cuenta para esta operación.
550	No se puede acceder al fichero o directorio (no existe, acceso denegado, etc.).
552	Espacio de disco insuficiente: transferencia abortada.
553	Nombre de fichero no permitido.

Cuando ha establecido la conexión de control con el intérprete de protocolo del servidor y antes de enviar ningún comando, el cliente debe esperar a recibir una respuesta, que puede tener los códigos 220, 421 ó 120. Si el código es 120, el cliente debe esperar hasta que el servidor envíe una respuesta 220. Cuando el servidor indique que está preparado, puede empezar el intercambio de comandos y respuestas. Por norma general, el servidor cerrará la conexión de control cuando se lo solicite el cliente (con el comando de acabar sesión).

Formato

El texto explicativo de las respuestas es de formato libre, excepto en el caso de las respuestas 110 (comando RETR), 227 (comando PASV) y 257 (comandos PWD y MKD).

Si el cliente desea realizar **operaciones de transferencia**, su proceso de transferencia de datos debería estar preparado para recibir peticiones de conexión en un puerto TCP, que por defecto es el mismo desde el que el intérprete ha iniciado la conexión de control.

El encargado de establecer las conexiones de datos es el proceso de transferencia de datos del servidor, desde su puerto de datos. Por defecto, este puerto es $L - 1$ (donde L es el número de puerto correspondiente a la conexión de control*). El cliente puede solicitar el uso de puertos diferentes de los predeterminados por medio de los comandos adecuados.

*** Es decir, si el servidor ha recibido la conexión de control en el puerto 21, su puerto de datos por defecto será el 20.**

Generalmente, las conexiones de datos las cierra el servidor, excepto cuando sea la otra parte (el cliente o un servidor pasivo) quien envíe los datos y el modo de transmisión utilizado requiera el cierre de la conexión para indicar el final del fichero.

4.1.3. Funcionalidad del FTP

Las acciones que lleva a cabo el servidor para cada uno de los comandos definidos en la especificación RFC 959 son los siguientes:

1) Nombre de usuario (`USER`)

El argumento de este comando es el nombre que es preciso suministrar al sistema servidor para identificar al usuario con el objetivo de acceder a los ficheros. Éste suele ser el primer comando que envía el cliente. Asimismo, es posible enviarlo

en medio de una sesión; entonces el servidor se olvida de la identidad del usuario anterior y realiza las nuevas operaciones bajo el nombre del nuevo usuario.

Si el servidor envía la respuesta 230, significa que el proceso de autenticación ya se ha completado; si envía la 530, significa que este usuario no es admisible (por ejemplo, porque no hay ningún ususario con este nombre) y, si envía la 331 o la 332*, entonces se necesita una contraseña o una cuenta, respectivamente.

*** En muchos sistemas, un intento de conexión con un nombre de usuario inexistente dará como resultado una respuesta 331, para no proporcionar información a los usuarios ajenos al sistema sobre qué nombres son válidos y cuáles no.**

El código 332 indica que la operación solicitada se llevará a cabo tan pronto como se reciba el comando ACCT.

2) Contraseña (PASS)

El argumento de este comando es la contraseña que necesita el sistema servidor para verificar la identidad del usuario. Si no se necesita ninguna, la respuesta será 202; si se necesita pero es incorrecta, 530, y si es correcta, puede ser 230 (autenticación completada) o 332 (se necesita una cuenta).

3) Cuenta (ACCT)

Algunos sistemas pueden requerir que el usuario proporcione una identificación de cuenta, bien en el proceso de autenticación inicial, bien para efectuar determinadas operaciones, como almacenar ficheros. El servidor hará saber que necesita esta información enviando una respuesta 332 en el proceso de autenticación, o una respuesta 332 o 532 después de una determinada operación.

4) Estructura de fichero (STRU)

Este comando sirve para especificar cómo estarán estructurados los ficheros que deben transferirse. El tipo de estructura afecta al modo de transmisión, como veremos a continuación. Los valores posibles del argumento son los tres siguientes:

- R: el fichero consta de una secuencia de registros.
- P: la estructura se basa en páginas indexadas*.
- F: el fichero se considera simplemente una secuencia de bytes (en este caso se considera que sólo hay estructura de fichero).

*** La opción P era útil en los sistemas de la época en que se desarrolló el FTP; hoy día está prácticamente en desuso.**

Si no se utiliza el comando STRU, el tipo de estructura por defecto es F.

5) Modo de transmisión (MODE)

El argumento indica cómo se transmitirán los ficheros. Puede tener los valores siguientes:

- B: la transmisión se lleva a cabo en bloques de datos, cada uno precedido de una cabecera con la longitud del bloque (2 bytes) y un código descrip-

tor (1 byte). Este último sirve para indicar, por ejemplo, si el bloque es el último de un registro o del fichero.

- C: la transmisión se realiza en bloques más compactos, con cabeceras de un solo byte, y permiten codificar una secuencia de hasta 64 bytes repetidos en un bloque de 2 bytes.

- S: la transmisión se efectúa en modo *stream*; es decir, como una simple secuencia de bytes. Si se utiliza con el tipo de estructura en registros, se insertan códigos de control en la secuencia para señalar los finales de registro y de fichero. Si el tipo de estructura es de fichero y la transmisión es en modo *stream*, la única manera de indicar el final de fichero es cerrando la conexión de datos.

¿Cuándo se cierra la conexión de datos?

La combinación modo *stream* y estructura de fichero es la única que requiere cerrar la conexión de datos después de cada transferencia. En los otros casos, el servidor puede optar entre mantenerla abierta o cerrarla, e informa de la misma al cliente con una respuesta 250 o 226, respectivamente.

Si no se utiliza el comando MODE, el modo de transmisión por defecto es S.

6) Tipo de representación (`TYPE`)

Con este comando, se especifica cómo se representarán los datos de los ficheros que deben transferirse. El proceso que los envía debe convertir el contenido de los ficheros en la representación especificada, y el proceso que los recibe, debe convertirlos en su representación local. Los valores posibles del argumento son los siguientes:

- A: se utiliza la especificación NVT definida en el protocolo Telnet (caracteres de 8 bits codificados en ASCII y finales de línea representados con la secuencia <CRLF>).

- E: se utilizan caracteres de 8 bits codificados en EBCDIC.

Argumentos opcionales a los tipos A y E

En los tipos de representación A y E, un segundo argumento opcional permite indicar cómo se especifica la información de control para el formato vertical del texto (separación entre párrafos, cambios de página, etc.) y puede valer N (no hay información de formato), T (hay caracteres de control Telnet: ASCII\EBCDIC) o C (se utiliza el formato del estándar ASA, RFC 740).

- I: se envía una imagen del fichero consistente en una secuencia arbitraria de bits, codificados en bytes (8 bits), que el receptor debe almacenar tal como le llega (o añadiendo ceros al final si el sistema local necesita que la longitud total sea múltiplo de alguna cantidad).

- L: se envía una secuencia de bytes lógicos de *n* bits, siendo *n* el valor del segundo argumento (obligatorio en este caso), con todos los bits consecutivos agrupados en bytes de 8 bits. Según el valor de *n*, es posible que el receptor necesite aplicar alguna transformación (reversible) para almacenar los datos.

Observad...

.. que, sea cual sea el tipo de representación, la transferencia siempre se realiza en bytes de 8 bits.

Si no se utiliza el comando TYPE, el tipo de representación por defecto es A (y sin información de formato vertical).

7) Cambiar de directorio (**CWD**)

Normalmente, cada usuario tiene asignado un directorio por defecto en el sistema de ficheros del servidor. Este comando sirve para cambiar el directorio de trabajo por el directorio que indica el argumento.

8) Cambiar al directorio padre (**CDUP**)

Este comando no recibe ningún argumento. Se trata de un caso particular del anterior que el protocolo proporciona para facilitar el acceso al directorio padre del actual con independencia de la sintaxis utilizada en el sistema de ficheros para referenciarlo.

9) Montar un sistema de ficheros (**SMNT**)

El argumento es un nombre que, en la sintaxis del sistema de ficheros del servidor, permite seleccionar un nuevo grupo de ficheros (por ejemplo, otro sistema de ficheros, otra partición del disco, otro disco, etc.).

10) Mostrar el directorio actual (**PWD**)

En la respuesta a este comando (código 257), el servidor informa del directorio actual. Para facilitar la interpretación de la respuesta, el nombre del directorio debe ser la primera palabra que venga después del código numérico, y debe ir delimitado por el carácter "".

11) Puerto de datos (**PORT**)

Con este comando, el cliente indica cuál es su puerto de datos, en caso de que sea diferente del puerto por defecto. Para efectuar las transferencias, el servidor deberá establecer la conexión de datos en el puerto especificado. Si ya había una conexión de datos establecida con otro puerto cuando se recibe este comando, el servidor deberá cerrarla.

Recordad...

... que el puerto de datos por defecto del cliente es el mismo desde el que ha establecido la conexión de control.

El argumento debe tener esta forma: *h1*,*h2*,*h3*,*h4*,*p1*,*p2* (cada uno de los elementos es un número decimal entre 0 y 255). Los cuatro primeros números forman la dirección IP, y los dos últimos, el número de puerto (que se representa de más a menos peso).

Cambio de puerto

Cuando el modo de transmisión requiere cerrar la conexión después de cada transferencia, se suele utilizar este comando para ir variando el número de puerto y evitar así las demoras que puede haber cuando se intenta reutilizar un puerto TCP que se acaba de cerrar.

El cliente puede especificar una dirección IP diferente de la suya; de esta manera, se puede efectuar una transferencia entre dos servidores, como ilustra la figura del modelo FTP, en que un cliente controla la transferencia de datos entre dos servidores.

Consultad la figura mencionada en el subapartado 4.1.1 de este módulo didáctico.

12) Puerto pasivo (PASV)

Este comando sirve para indicar al servidor que, cuando se le envíe un comando de transferencia, en lugar de establecer de manera activa la conexión de datos, debe quedar preparado para recibirla de manera pasiva. En la respuesta (código 227), el servidor devuelve la dirección en que esperará las peticiones de conexión, y utilizará la misma notación del argumento del comando PORT.

El comando PASV se utiliza en las transferencias entre servidores. El cliente debe establecer conexiones de control con los dos servidores, enviar un comando PASV a uno de los mismos y pasar la dirección devuelta al otro con un comando PORT. Entonces debe enviar el comando de transferencia correspondiente (leer o almacenar) al servidor pasivo, y el comando complementario al activo.

13) Reservar espacio (ALLO)

Algunos sistemas pueden requerir que se especifique la longitud de un fichero antes de almacenarlo. El argumento constituye el número de bytes lógicos a reservar. Si es necesario, el primer argumento puede ir seguido de la cadena R *n*, donde n indica la longitud máxima de los registros o páginas (para ficheros con tipo de estructura R o P).

Longitud del byte lógico

Recordad que cada byte lógico tiene *n* bits, donde *n* es el argumento del comando TYPE L o, por defecto, 8.

14) Obtener (RETR)

Ésta es la operación de transferencia de ficheros del servidor hacia el cliente (o hacia el servidor pasivo). El argumento es el nombre del fichero que debe transferirse.

Tanto en esta operación como en las de almacenar y añadir, si el modo de transmisión es B o C, el proceso que envía los datos puede insertar un tipo especial de bloque denominado *marca de reanudación* (su contenido es un identificador de la posición actual del fichero), que deberá utilizarse en caso de error de la transferencia. Cuando encuentra la marca, el receptor asocia a la posición actual un identificador propio y se lo notifica al usuario. Si quien actúa de receptor es el servidor, activo o pasivo, la notificación se lleva a cabo por medio de una respuesta con el código 110 y el texto MARK *c* = *s**.

*** c y s son los identificadores de posición del cliente y servidor, respectivamente.**

15) Almacenar (STOR)

Ésta es la operación de transferencia de ficheros del cliente hacia el servidor. El argumento es el nombre del fichero en que el servidor debe almacenar los

datos. Si este fichero no existe, se crea; si ya existe, se descarta su contenido y se sustituye por los datos recibidos.

16) Almacenar con nombre único (`STOU`)

Esta operación es como la anterior, pero sin argumento: el servidor elegirá un nombre para el fichero (en el directorio actual) que no coincida con ninguno de los ya existentes. En la respuesta debe notificarse el nombre elegido.

17) Añadir (`APPE`)

Esta operación también es como la de almacenar, excepto que, si el fichero ya existe, los datos recibidos se añadirán al final de su contenido.

18) Listar (`LIST`)

El argumento de este comando es opcional. Si no hay argumento, el servidor envía por la conexión de datos una lista de los ficheros del directorio actual y sus atributos, por lo general en un formato propio del sistema. Si hay argumento, la lista se refiere al fichero o grupo de ficheros especificado, o a los ficheros contenidos en el directorio especificado.

19) Listar nombres (`NLST`)

Este comando es como el anterior, pero con el formato de la lista normalizado, lo que significa que consta sólo de los nombres de los ficheros, separados por finales de línea, que permite que la lista devuelta sea procesada, por ejemplo, para que el cliente pueda implementar una operación para obtener un grupo de ficheros.

Como precaución,...

... el cliente se debería asegurar que el tipo de representación es `A` o `E` antes de enviar el comando `LIST` o `NLST`.

20) Abortar (`ABOR`)

Este comando hace que el servidor interrumpa la transferencia en curso (si hay) y, después, cierre la conexión de datos. Si no, simplemente cierra la conexión de datos. En el primer caso, el servidor envía un código 426 como respuesta al comando de transferencia y, a continuación, un código 226 en respuesta al comando `ABOR`. En el segundo caso, sólo envía la respuesta 226.

Para indicar que el servidor debe atender un comando (`ABOR`, `STAT`, `QUIT`) mientras se esté llevando a cabo una transferencia, la especificación RFC 959 sugiere que el cliente envíe para la conexión de control el código de función IP del protocolo Telnet, seguido de una señal *synch* (es decir, un envío de datos urgentes TCP combinado con el código DM), y a continuación el comando FTP.

21) Retomar (`REST`)

Recordad que...

... sólo puede haber marcas de reanudación en los modos de transmisión `B` o `C`.

Si una transferencia ha sido interrumpida por cualquier causa (caída del servidor, del cliente, de la red, etc.) es posible retomarla a partir de un punto indi-

cado por una marca de reanudación. El cliente debe enviar el comando REST con un argumento igual que el identificador de posición del servidor correspondiente a la marca deseada e, inmediatamente a continuación, el comando de transferencia que desea retomar.

22) Borrar (DELE)

El efecto de este comando es borrar el fichero especificado en el argumento.

23) Nombre antiguo (RNFR)

Para cambiar el nombre de un fichero, en primer lugar el cliente debe enviar este comando, con el nombre actual en el argumento e, inmediatamente, el comando RNTO.

24) Nombre nuevo (RNTO)

El argumento de este comando es el nombre nuevo que se debe conferir al fichero. Sólo se puede utilizar inmediatamente a continuación de un comando RNFR.

25) Crear un directorio (MKD)

El servidor crea el directorio indicado por el argumento. Si la operación tiene éxito, el formato del código de respuesta (código 257) es idéntico al del comando PWD.

Formato de la respuesta 257

Según la especificación RFC 959, el nombre retornado en la respuesta 257 debe ser apto para poderlo utilizar como argumento del comando CWD (cambiar directorio). Como en algunos sistemas (obsoletos) este argumento no podía ser un nombre relativo al directorio actual, sino que debía ser un nombre absoluto (y el del comando MKD, en cambio, sí que podía ser relativo), por norma general los servidores responden al comando MKD con el nombre completo del directorio creado.

26) Borrar un directorio (RMD)

El servidor borra el directorio indicado por el argumento.

27) Estatus (STAT)

Si se envía este comando durante una transferencia, el servidor incluye en la respuesta información sobre el estado actual de la transferencia. Si no, se le puede pasar un nombre de fichero como argumento para que el servidor envíe

información similar a la del comando `LIST`, pero por la conexión de control, o bien se puede utilizar sin argumento para obtener información general sobre el proceso FTP.

28) Sistema (`SYST`)

El servidor envía una respuesta informando sobre qué tipo de sistema es. La primera palabra del texto debe ser uno de los nombres de sistema normalizados de la lista oficial de números asignados.

WEB

La IANA (Internet Assigned Numbers Authority) es la encargada de publicar la lista de números asignados. En el momento de elaborar este módulo, la última versión de la lista se puede encontrar en la especificación RFC 1700.

ftp://ftp.isi.edu/in-notes/iana/assignments/

Ejemplo

Un cliente UNIX puede utilizar este comando para saber si el servidor también es UNIX y, si lo es, invocar automáticamente el comando `TYPE I` para mejorar la eficiencia de las transmisiones.

29) Servicios adicionales (`SITE`)

Si el servidor ofrece servicios adicionales además de las operaciones estándar del protocolo, el cliente puede invocarlos por medio de los argumentos del comando `SITE`, cuya sintaxis depende del mismo servidor.

30) Ayuda (`HELP`)

Este comando, mediante el cual el servidor envía información general sobre la implementación del protocolo*, se puede utilizar sin argumentos. No obstante, el servidor también puede proporcionar información más específica si se le pasan argumentos**.

*** Por ejemplo, qué comandos soporta y cuáles no.**

**** Por ejemplo, un nombre de comando.**

31) Operación nula (`NOOP`)

El servidor simplemente envía una respuesta 200.

32) Reinicializar (`REIN`)

El servidor reinicializa la sesión olvidando la identidad del usuario y volviendo a dar a los parámetros de transmisión los valores por defecto. La sesión queda en el mismo estado en que se encontraba justo después de establecer la conexión de control.

33) Acabar la sesión (`QUIT`)

El servidor da la sesión por finalizada y cierra la conexión de control (si hay una transferencia en curso, después de enviar la respuesta correspondiente).

La tabla siguiente resume la sintaxis de los comandos y los posibles códigos de respuesta a cada uno de los mismos según la especificación RFC 959:

Sintaxis de los comandos y códigos de respuesta	
Comando	**Respuesta**
Establecimiento de conexión	220, 120, 421
USER *usuario*	230, 331, 332, 530
PASS *contraseña*	230, 202, 332, 530
ACCT *cuenta*	230, 202, 530
STRU F|R|P	200, 504
MODE S|B|C	200, 504
TYPE A|E|I|L [N|T|C|*n*]	200, 504
CWD *directorio*	250, 550
CDUP	250, 550
SMNT *sist-ficheros*	250, 202, 550
PWD	257, 550
PORT *h1*,*h2*,*h3*,*h4*,*p1*,*p2*	200
PASV	227
ALLO *long-1* [R *long-2*]	200, 202, 504
RETR *fichero*	226, 250, 110, 125, 150, 550
STOR *fichero*	226, 250, 110, 125, 150, 452, 532, 552, 553
STOU	226, 250, 110, 125, 150, 452, 532, 552, 553
APPE *fichero*	226, 250, 110, 125, 150, 452, 532, 550, 552, 553
LIST [*nombre*]	226, 250, 125, 150
NLST [*nombre*]	226, 250, 125, 150
ABOR	226
REST *marca*	350
DELE *fichero*	250, 550
RNFR *fichero*	350, 450, 550
RNTO *fichero*	250, 503, 532, 553
MKD *directorio*	257, 550
RMD *directorio*	250, 550
STAT [*nombre*]	211, 212, 213
SYST	215
SITE *cadena*...	200, 202
HELP [*cadena*]	211, 214
NOOP	200
REIN	220, 120, 421
QUIT	221

Además de los códigos de respuesta específicos contenidos en esta tabla, el cliente también puede recibir códigos de respuesta generales a cada comando, como 500, 501, 502*, 530 y 421, o bien, en el caso de los comandos de transferencia, 425, 426 y 450.

*** Esta respuesta sólo es válida para los comandos no básicos; es decir, los que no pertenecen al grupo de la "implementación mínima".**

4.1.4. Implementaciones del FTP

A la hora de implementar el protocolo FTP, debemos distinguir servidor de cliente:

1) Servidores FTP

Según la especificación RFC 959, la implementación mínima de un servidor FTP debe soportar los comandos siguientes:

- `USER`
- `RETR`
- `TYPE A N`
- `NOOP`
- `STOR`
- `MODE S`
- `QUIT`
- `PORT`
- `STRU F`
- `STRU R`

Muchos servidores soportan el acceso público (posiblemente restringido) a una parte de su sistema de ficheros sin necesidad de seguir el procedimiento normal de autenticación.

Este tipo de servicio se conoce como *FTP anónimo*. Por norma general, se accede al mismo utilizando el nombre de usuario `anonymous` como argumento del comando `USER`. Si el servidor pide una contraseña, suele aceptar cualquier cadena. En este caso, es costumbre proporcionar la dirección de correo electrónico del usuario como contraseña.

2) Clientes FTP

En la actualidad, existen muchas implementaciones de clientes FTP para una gran variedad de sistemas diferentes (incluyendo los navegadores WWW); sin embargo, el cliente más utilizado durante mucho tiempo ha sido la utilidad del sistema operativo UNIX denominada precisamente `ftp`.

Este cliente presenta al usuario la mayoría de las respuestas del servidor, incluyendo los códigos numéricos. Los principales comandos que ofrece son los siguientes:

- `open`: permite especificar el nombre del servidor al que es preciso conectarse, si no se ha pasado como argumento del programa, y entonces pide automáticamente el nombre de usuario y, si procede, la contraseña.
- `cd`, `pwd`, `dir`: envían los comandos `CWD`, `PWD` y `LIST` al servidor.

Recordad

Los nombres de los comandos se pueden abreviar siempre que no generen ambigüedades.

- `ascii`, `binary`: envían los comandos `TYPE A` y `TYPE I` al servidor.
- `get`: efectúa la secuencia `PORT-RETR` para copiar un fichero del servidor.
- `put`: efectúa la secuencia `PORT-STOR` para copiar un fichero en el servidor.
- `^C` (o el carácter que genere la señal de interrupción): envía el comando `ABOR`.
- `delete`, `mkdir`, `rmdir`: envían los comandos `DELE`, `MKD` y `RMD`.
- `rename`: envía la secuencia `RNFR-RNTO` necesaria para cambiar el nombre de un fichero.
- `mget`: envía el comando `NLST` para saber qué ficheros concuerdan con un patrón, y después una serie de comandos `RETR` para poderlos copiar.
- `mput`: envía una serie de comandos `STOR`.
- `mdelete`: envía el comando `NLST` y, a continuación, una serie de comandos `DELE`.
- `quote`: permite enviar un comando directamente al servidor (por ejemplo, `quote syst`).
- `?`: muestra un mensaje de ayuda.
- `quit` o `bye`: envía el comando `QUIT` y cierra el programa.

4.1.5. Ejemplo de sesión FTP

A continuación, mostramos los mensajes intercambiados entre un cliente y un servidor en una hipotética sesión de transferencia por medio de la utilidad `ftp`.

Después de cada comando de usuario, se muestran los mensajes enviados, y se indica su origen y el destino (`C` para el cliente, `S` para el servidor), y el número de puerto:

```
open ftp.uoc.es
C:4695 → S:21  (establecimiento de conexión)
S:21 → C:4695  220 tibet FTP server (5.6) ready.<CRLF>
(usuari) anonymous
C:4695 → S:21  USER anonymous<CRLF>
S:21 → C:4695  331 Guest login ok, send ident
               as password.<CRLF>
(contraseña) usuari@acme.com
```

```
C:4695 → S:21  PASS usuari@acme.com<CRLF>
S:21 → C:4695  230 Guest login ok, access restrictions
                   apply.<CRLF>
cd pub
C:4695 → S:21  CWD pub<CRLF>
S:21 → C:4695  250 CWD command successful.<CRLF>
dir
C:4695 → S:21  PORT 193,146,196,254,18,88<CRLF>
S:21 → C:4695  200 PORT command successful.<CRLF>
C:4695 → S:21  LIST<CRLF>
S:20 → C:4696  (establecimiento de conexión)
S:21 → C:4695  150 ASCII data connection for /bin/ls
               (193.146.196.254,4696) (0 bytes).<CRLF>
S:20 → C:4696  total 20<CRLF>
               drwxr-xr-x 7 0 other 512 Aug 3 07:10
               .<CRLF>
               ...
S:20 → C:4696  (cierre de conexión)
S:21 → C:4695  226 ASCII Transfer complete.<CRLF>

cd doc
C:4695 → S:21  CWD doc<CRLF>
S:21 → C:4695  250 CWD command successful.<CRLF>

dir
C:4695 → S:21  PORT 193,146,196,254,18,89<CRLF>
S:21 → C:4695  200 PORT command successful.<CRLF>
C:4695 → S:21  LIST<CRLF>
S:20 → C:4697  (establecimiento de conexión)
S:21 → C:4695  150 ASCII data connection for /bin/ls
               (193.146.196.254,4697) (0 bytes).<CRLF>
S:20 → C:4697  total 886<CRLF>
               drwxr-xr-x 2 0 other 512 Oct 24 1996 .<CRLF>
               ...
S:20 → C:4697  (cierre de conexión)
S:21 → C:4695  226 ASCII Transfer complete.<CRLF>
get README
C:4695 → S:21  PORT 193,146,196,254,18,91<CRLF>
S:21 → C:4695  200 PORT command successful.<CRLF>
C:4695 → S:21  RETR README<CRLF>
S:20 → C:4699  (establecimiento de conexión)
S:21 → C:4695  150 ASCII data connection for README
               (193.146.196.254,4699) (95 bytes).<CRLF>
S:20 → C:4699  (contenido del fichero)
```

```
S:20 → C:4699  (cierre de conexión)
S:21 → C:4695  226 ASCII Transfer complete.<CRLF>

bye
C:4695 → S:21  QUIT<CRLF>
S:21 → C:4695  221 Goodbye.<CRLF>
S:21 → C:4695  (cierre de conexión)
```

4.2. El TFTP

Hay situaciones en las que se necesita transferir ficheros de un ordenador a otro y el FTP no es apropiado, principalmente a causa de su complejidad.

Un ejemplo típico en el que se observa la necesidad de transferir ficheros de un ordenador a otro es el de las estaciones de trabajo sin disco, que cargan el sistema operativo por medio de la red. En este caso, debe haber un ordenador que funcione como "servidor de arranque" de la estación, proporcionándole el fichero o ficheros en que se encuentra el código ejecutable del sistema operativo. Un pequeño programa residente en la memoria ROM de la estación debe controlar la transferencia de los ficheros.

Para esta operación el FTP no es adecuado, puesto que requiere una implementación del protocolo de transporte TCP, establecer en el mismo diferentes conexiones simultáneas, etc.

Para satisfacer las necesidades de transmisiones simplificadas, se ha definido el TFTP*, cuya última versión está especificada en el estándar RFC 1350.

Este protocolo está basado en datagramas, sólo proporciona dos operaciones (leer y escribir ficheros) y no hay ningún tipo de identificación ni autenticación de usuario.

*** TFTP es la sigla de *trivial file transfer protocol.***

4.2.1. Conceptos básicos del TFTP

Como el TFTP se basa en datagramas, generalmente se utiliza con el protocolo de transporte UDP. El número de puerto al que el cliente debe enviar las peticiones de servicio es el 69.

Lectura complementaria

Si queréis más información sobre el TFTP, consultad la obra siguiente:

K. Sollins (1992, julio). *RFC 1350 - The TFTP protocol.*

Una **transferencia TFTP** se inicia con un datagrama enviado por el cliente en el que se solicita la operación deseada: leer o escribir un fichero. A partir de entonces, se establece un diálogo en el que cada parte envía un datagrama de manera alternada. Cada uno de estos datagramas sirve de confirmación de recepción del anterior. Ello significa que en cada momento sólo puede haber un datagrama pendiente de ser confirmado.

De este modo, la recuperación de los errores de transmisión es muy simple: si pasa un tiempo sin que llegue la respuesta a un datagrama, se reenvía y si se recibe un datagrama que ya se había recibido con anterioridad, se ignora. Por tanto, basta con guardar el último datagrama enviado por si debe retransmitirse.

El cliente debe enviar su primer datagrama desde un puerto `C` al puerto 69 del servidor. Cuando lo recibe, el servidor elige un número de puerto `S` que debería ir cambiando en cada transferencia. Todos los datagramas que envíe el servidor tendrán como puerto de origen el número `S` y como puerto de destino el número `C`; todos los datagramas que envíe el cliente, excepto el primero, tendrán como puerto de origen el número `C` y como puerto de destino el número `S`. Ello permite detectar una posible duplicación del primer datagrama: si el servidor lo recibe dos veces o más, debe enviar ambas respuestas desde puertos diferentes; el cliente aceptará la primera y enviará mensajes de error a los puertos de los que provengan las otras.

En el transcurso de la transferencia, una de las partes envía los datos del fichero y la otra sólo envía confirmaciones. Los datos se envían en bloques de longitud fija, 512 bytes, excepto el último bloque, que tendrá entre 0 y 511 bytes. Cada bloque se envía en un datagrama.

Bloque de cero bytes

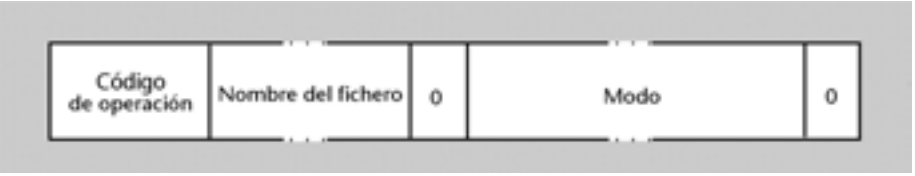
Habrá un último bloque de cero bytes sólo cuando la longitud del fichero que deba transmitirse sea múltiplo de 512.

La transferencia se acaba cuando el receptor recibe el último bloque de datos y envía la confirmación correspondiente. En este momento, puede dar por finalizada la comunicación. Opcionalmente, puede esperar por si vuelve a recibir el último bloque, lo que significaría que la última confirmación no ha llegado al emisor. Si sucede esto, sólo es preciso reenviar esta confirmación.

Por su parte, el emisor dará por acabada la transferencia cuando reciba la última confirmación o cuando haya transcurrido cierto tiempo retransmitiendo el último bloque de datos y no reciba respuesta. En este último caso, podría ser que la transferencia se hubiera efectuado correctamente y que el único problema estuviera en la transmisión de la última confirmación.

4.2.2. Funcionalidad del TFTP

El TFTP ofrece dos operaciones básicas: leer ficheros del servidor y escribir ficheros en el servidor. El datagrama inicial indica la operación que el cliente quiere llevar a cabo y tiene el formato siguiente:

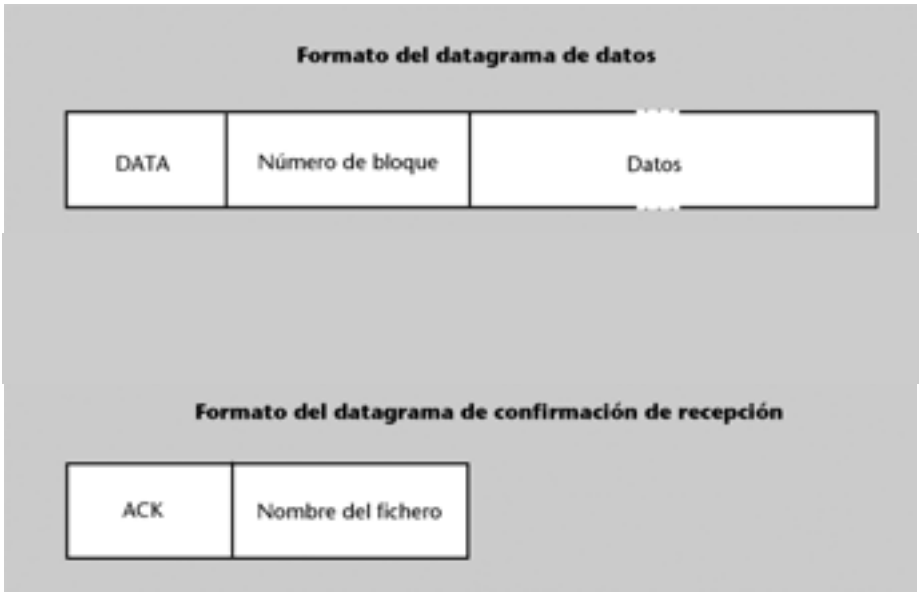


El código de operación es un número de dos bytes, que puede ser `RRQ` o `WRQ` si el cliente solicita leer o escribir un fichero, respectivamente. A continuación, existen dos cadenas de caracteres, acabadas con un byte igual a cero: la primera es el nombre del fichero y la segunda es el modo de transferencia (el equivalente del tipo de representación en FTP). Esta segunda cadena puede ser `netascii` u `octet` (en caracteres ASCII, y en cualquier combinación de mayúsculas y minúsculas). El primer valor indica que los datos son caracteres ASCII tal como se usan en el protocolo Telnet, y el segundo indica que los datos son bytes arbitrarios de 8 bits.

Modo mail

En versiones anteriores del protocolo, había un tercer modo de transferencia llamado `mail`, sólo aplicable a las operaciones de escritura, en el que el nombre del fichero era sustituido por el nombre de un usuario que debía recibir los datos por correo.

Los datagramas que contienen los datos y las confirmaciones de recepción tienen los formatos siguientes:



El primer campo es el código de operación y el segundo es el número de bloque que se envía o se confirma (ambos campos son de dos bytes). Cada bloque del fichero tiene un número correlativo, empezando por 1, que sirve para distinguir las confirmaciones duplicadas.

Si el cliente envía un datagrama inicial `RRQ`, el servidor contesta con un datagrama `DATA` con número de bloque igual a 1 y, si el cliente envía un datagrama inicial `WRQ`, el servidor contesta con un datagrama `ACK` con número de bloque igual a 0 y, a continuación, el cliente envía el primer datagrama de datos. Entonces, cliente y servidor se intercambian datagramas `ACK` y `DATA` de manera alternada, con las retransmisiones necesarias si no llega el datagrama que corresponde en cada momento.

El TFTP también prevé la terminación de la transferencia si se produce algún error. Cuando se detecta el error, se envía un datagrama con el formato siguiente:

Los dos primeros campos son el código de operación y el código de error (cada uno de dos bytes). A continuación, hay una cadena de caracteres, acabada en 0, que puede servir para describir a un usuario humano la causa del error.

Número de puerto incorrecto

Si llega un datagrama del servidor con un número de puerto de origen incorrecto (probablemente a causa de un datagrama inicial duplicado), la transferencia con este puerto queda interrumpida, pero la que utiliza el puerto correcto debe continuar con normalidad.

Un datagrama de error indica que se da por acabada la transferencia y no debe confirmarse ni, por tanto, retransmitirse. Ahora bien, si por alguna razón se pierde este datagrama, la otra parte interpretará que la transferencia ha acabado prematuramente cuando haya transcurrido cierto tiempo retransmitiendo sin recibir nada.

En las tablas siguientes se presenta una relación de los códigos numéricos asignados a cada operación y a cada tipo de error TFTP, respectivamente:

Código	Operación
1	`RRQ`
2	`WRQ`
3	`DATA`
4	`ACK`
5	`ERROR`

Código	Error
0	Error indefinido (veáse el mensaje)
1	No se ha encontrado el fichero
2	Acceso denegado
3	Disco lleno
4	Operación no válida
5	Número de puerto incorrecto
6	Ya existe el fichero
7	Usuario incorrecto (en modo `mail`)

4.2.3. Implementaciones del TFTP

En muchos sistemas UNIX hay un servidor del TFTP llamado `tftpd`. Como no existe ningún tipo de identificación de usuario, los clientes pueden acceder en principio a cualquier fichero con permisos de acceso público. En algunos sistemas, sin embargo, puede restringirse el acceso a un directorio o unos directorios determinados. No obstante, el servicio TFTP suele estar inhabilitado, y sólo lo ofrecen los sistemas en que se necesita por algún motivo concreto (por ejemplo, los servidores de arranque de las estaciones sin disco).

Por norma general...

... hay un directorio llamado `/tftpboot` en el que se guardan las imágenes de los sistemas operativos de los clientes sin disco, y sólo se permite a este directorio el acceso por medio del TFTP.

Asimismo, hay un programa cliente llamado `tftp` que funciona de manera similar a la utilidad `ftp`: admite comandos como `get`, `put`, `ascii`, `binary` o `quit` (pero no `cd`, `dir`, `delete`, `rename`, etc., porque el protocolo no los soporta).

5. Correo electrónico Internet

El correo electrónico es la aplicación distribuida que permite enviar mensajes electrónicos por medio de sistemas informáticos.

Al especificar esta aplicación, se consideró adecuado que todas sus características siguieran las ideas básicas del correo postal:

- Las **operaciones** permiten básicamente enviar mensajes y recibirlos.
- Los **elementos** son equivalentes a los mensajes, los usuarios, las oficinas de correo y los buzones.
- La **funcionalidad** está basada en la filosofía del almacenamiento y el reenvío: cuando un mensaje llega a una oficina de correos, esta última lo almacena y no lo reenvía hasta el momento en que lo considera oportuno. De este modo, los mensajes van de oficina de correos en oficina de correos hasta que llegan al destinatario.

Almacenamiento y reenvío

La filosofía del almacenamiento y el reenvío permite superar problemas como las caídas de la red; en este caso, los mensajes no se pierden, puesto que en cada momento hay una oficina responsable del mensaje.

Para llevar a cabo esta funcionalidad, se definieron los protocolos siguientes:

- **SMTP***, para la **transferencia de mensajes**.
- **POP3****, para el **acceso simple a buzones de correo**.
- **IMAP4rev1*****, para el **acceso complejo a buzones de correo**.

* **SMTP:** *simple mail transfer protocol*
** **POP3:** *post office protocol*
*** **IMAP4rev1:** *Internet message acces protocol*

También fue necesario definir un **formato de mensaje**, el **RFC 822**, que con posterioridad se amplió y dio lugar al formato **MIME**.

5.1. Formato de los mensajes: el RFC 822

Lectura complementaria

Si deseáis más información sobre el formato de los mensajes de correo Internet, el RFC 822, consultad la obra siguiente:

D. Crocker (1982, 13 de agosto). *RFC 822 - Standard for the format of ARPA Internet text messages*.

Antes de describir los diferentes protocolos relacionados con el correo electrónico, es preciso ver cuál es el formato o la norma de los mensajes que se utiliza en el mismo. Este formato recibe el nombre del estándar en que se especifica, **RFC 822**, y se basa en los elementos típicos de las cartas postales; es decir, el sobre, que contiene la información del destinatario y del remitente de la carta, el contenido, que es el mensaje en sí.

El estándar especifica que los mensajes de correo electrónico están formados por las dos partes siguientes:

- La **cabecera**, que recoge la información general del mensaje. Equivale al sobre de la carta postal y está formada por una serie de **campos de cabecera**, cada uno de los cuales incluye un tipo concreto de información.

- El **cuerpo del mensaje**, que contiene el mensaje en sí. Corresponde al contenido de la carta postal. Esta parte es opcional.

Consultad en el anexo 1 de este módulo la notación que se utiliza para describir los campos de los mensajes.

Cada campo de cabecera consta de un **nombre del campo** (que identifica el tipo de información) seguido por el carácter ":", opcionalmente acompañado por un **cuerpo del campo** (que incluye la información del campo), y acaba con un carácter <CRLF>. El formato de los campos es el siguiente:

```
NombredelCampo:
   [CuerpodelCampo]<CRLF>
```

Por norma general, cada campo de cabecera se representa en una sola línea, empezando desde el primer carácter de la misma. En el caso de que se necesite más de una, es preciso codificar estas líneas adicionales empezando por un carácter, o más, de espacio o tabulador.

El cuerpo del mensaje es simplemente una secuencia de líneas con caracteres ASCII. El cuerpo está separado de la cabecera por una línea nula (es decir, por la secuencia <CRLF><CRLF>).

Algunos sistemas de correo electrónico permiten reenviar mensajes a los usuarios. Por este motivo, se incluyen en el mismo unos campos de cabecera con información sobre el reenvío. Estos campos son los que llevan el prefijo Resent- y tienen la misma semántica que los campos correspondientes que no lo llevan. No obstante, siempre convendrá tener presente que, dentro de un mensaje, los campos que llevan prefijo son los más recientes.

5.1.1. Información de la cabecera

Los campos de cabecera del mensaje deben proporcionar información general del mensaje, incluyendo la información equivalente a la de un sobre postal; de este modo, encontramos los campos siguientes:

Consultad el formato de las direcciones para identificar los buzones de usuario en el subapartado 5.2.2 de este módulo didáctico.

1) Originador (**From**/**Resent-From**)

La identidad y la dirección del buzón de la persona o las personas que originan el mensaje se puede incluir en el campo From o Resent-From. En este campo, se puede introducir la dirección del buzón de un originador o más.

```
From: 1#dirección
Resent-From: 1#dirección
```

2) Destinatario (`To`/`Resent-To`)

El RFC 822 identifica al destinatario o destinatarios principales del mensaje por medio del campo `To` o `Resent-To`. En este campo, se puede introducir la dirección de un destinatario o más.

```
To: 1#dirección
Resent-To: 1#dirección
```

3) Destinatario de copia (`Cc`/`Resent-cc`)

Asimismo, existe la posibilidad de enviar copias de un mensaje. En este caso, la identidad del receptor o receptores secundarios del mensaje se especifica con el campo `Cc` o `Resent-cc`. En este último, se puede introducir la dirección de un destinatario de copia o más.

```
Cc: 1#dirección
Resent-cc: 1#dirección
```

4) Destinatario adicional (o de copia ciega) (`Bcc`/`Resent-bcc`)

Cuando se desea enviar una copia del mensaje a destinatarios adicionales, sin que ninguno de ellos (los principales, los de copia y los de copia ciega) sepan que otros destinatarios la han recibido, se utiliza el campo `Bcc` o `Resent-bcc` que no lo ve ninguno de ellos.

```
Bcc: 1#dirección
Resent-bcc: 1#dirección
```

5) Destinatario de respuesta (`Reply-To`/`Resent-Reply-To`)

La identificación del destinatario o destinatarios a los que deben enviarse las respuestas se puede llevar a cabo por medio del campo `Reply-To` o `Resent-Reply-To`; en este último se puede introducir la dirección de un destinatario de la respuesta o más.

```
Reply-To: 1#dirección
Resent-Reply-To:1# dirección
```

6) Asunto (`Subject`)

Otra posibilidad que ofrece el RFC 822 es enviar un texto explicativo del asunto del mensaje con el campo `Subject`.

```
Subject: texto
```

7) Data (`Date`/`Resent-Date`)

Dentro de un mensaje también puede incluirse la hora y la fecha en que se envía por medio del campo `Date` o `Resent-Date`. El primer sistema de correo que recibe el mensaje genera automáticamente este campo.

```
Date: fecha-hora
Resent-Date: fecha-hora
```

8) Sistema remitente (`Sender`/`Resent-Sender`)

Algunas veces el usuario que envía el mensaje no es el mismo que el autor. En estos casos, la identidad del agente (persona, sistema o proceso) que envía en realidad el mensaje se identifica con el campo `Sender` o `Resent-Sender`.

```
Sender: buzón
Resent-Sender: buzón
```

9) Camino de retorno hacia el originador (`Return-path`)

El mensaje puede incluir la identificación del camino de retorno hacia el originador del mensaje con el campo `Return-path`. Este campo lo añade el sistema de transporte final que entrega el mensaje al receptor.

```
Return-path: addr-ruta
```

10) Información de sistemas intermedios (`Received`)

Existe la posibilidad de que cada sistema de transporte intermedio por el cual pasa el mensaje incluya información de los sistemas emisor (`from`) y receptor (`by`), del mecanismo físico (`via`), del identificador del mensaje (`id`), de las especificaciones originales (`for`) y de la hora de recepción por medio del campo `Received`. Cada sistema intermedio añade una copia de este campo al mensaje.

```
Received:
 [from dominio]
 [by dominio]
 [via atom]
 *(with atom)
[id id-msg]
 [for addr-spec]
 ; fecha-hora
```

11) Identificador del mensaje (`Message-ID`/`Resent-Message-ID`)

Cada mensaje debe incluir un identificador único del mensaje, para que pueda ser contestado o referenciado con posterioridad, en el campo `Message-ID` o `Resent-Message-ID`. El sistema que genera el identificador es el encargado de asegurar que el identificador sea único.

```
Message-ID: id-msg
Resent-Message-ID: id-msg
```

12) Identificador del mensaje contestado (`In-Reply-To`)

Cuando un mensaje constituye la respuesta de un mensaje anterior, el mensaje original se puede referenciar por medio de su identificador dentro del campo `In-Reply-To`.

```
In-Reply-To:
  *(frase | id-msg)
```

13) Identificador de mensajes referenciados (References)

Si se desea enviar un mensaje que se refiere a otros mensajes, el identificador del mensaje referenciado se puede incluir dentro del campo `References`.

```
References:
  *(frase | id-msg)
```

14) Palabras clave (`Keywords`)

Las palabras clave o frases referentes al mensaje se pueden incluir, separadas por comas, dentro del campo `Keywords`.

```
Keywords: #frase
```

15) Comentarios (`Comments`)

Se puede incluir un texto de comentario en el mensaje, sin interferir su contenido, por medio del campo `Comments`.

```
Comments: texto
```

16) Identificación del mecanismo de cifrado (`Encrypted`)

El RFC 822 permite cifrar el cuerpo del mensaje. En este caso, es conveniente enviar una o dos palabras que identifiquen el mecanismo de cifrado que se ha aplicado utilizando el campo `Encrypted`.

```
Encrypted: 1#2palabras
```

El cifrado en el RFC 822

El RFC 822 sólo define la sintaxis para especificar un mecanismo de cifrado; sin embargo, no especifica el mecanismo ni la manera de utilizarlo.

17) Campos de uso personal

El estándar permite también que los usuarios definan campos para uso personal. El nombre de los campos creados por un usuario debe empezar obligatoriamente por la cadena `X-`.

X-*campo-uso-personal*

18) Campos de extensión

En el futuro se pueden estandarizar nuevos campos de cabecera. Para que no haya conflictos con los campos de uso personal, el nombre de los campos nunca empezará por la cadena `X-`.

campo-extensión

El estándar RFC 822 establece que los únicos campos de cabecera que son obligatorios en un mensaje son los siguientes: fecha (`Date`), originadores (`From`), y destinatario (`To`) o destinatario de copia ciega (`Bcc`).

5.1.2. Ejemplo

En este apartado se presenta un ejemplo de mensaje RFC 822 en el que se pueden apreciar los campos de cabecera más típicos, así como un cuerpo breve del mensaje. Conviene recordar que es un mensaje en ASCII de 7 bits.

```
Date: 25 Jun 2003 0932 PDT
From: Jordi Inyigo <jinyigo@uoc.edu>
Subject: Ejemplo mensaje RFC 822
Sender: jmmarques@uoc.edu
Reply-To: jinyigo@uoc.edu
To: Ramon Marti <rmarti@uoc.edu>,
    xperramon@uoc.edu
Cc: Llorenc Cerda <lcerda@uoc.edu>,
     Jose Barcelo <jbarcelo@uoc.edu>, epeig@uoc.edu
Comment: os envío esta información que os puede
     interesar
In-Reply-To: <1234321.567898765@uoc.edu>
Received: from uoc.edu by peru.uoc.es
     (8.8.5/8.8.5) with ESMTP id SAA14826
     for <rmarti@uoc.edu >; Fri, 20 Jun 2003
     18:35:52 +0200 (MET DST)
Received: from rectorat.uoc.edu(147.83.35.35)
     by uoc.es via smap (V2.0) id xma020193;
     Mon, 20 Jun 2003 18:38:50 +0200 for
     rmarti@uoc.edu
Message-Id: <199809211639.SAA20364@uoc.edu>

Este mensaje tiene formato RFC 822 e incluye
algunos de los campos de cabecera mas utilizados.
```

5.2. El SMTP

El SMTP* es el protocolo más utilizado en Internet para transferir mensajes de correo electrónico. Proporciona la funcionalidad necesaria para conseguir una transferencia fiable y eficiente de mensajes de correo entre ordenadores que actúan como oficina de correos. Siguiendo las ideas del correo postal, el SMTP se basa en el almacenamiento y el reenvío. Es decir, cuando un mensaje llega a una oficina, queda almacenado en la misma cierto tiempo antes de ser entregado a otra oficina o al destinatario final. Conviene señalar, asimismo, que cada usuario debe disponer de un buzón para recibir mensajes, el cual siempre debe estar asociado a una oficina determinada.

*** Recordad que SMTP es la sigla de *simple mail transfer protocol*.**

Lectura complementaria

Si queréis más información sobre el SMTP, consultad la obra siguiente:

J. Postel (1982, 1 de agosto). *RFC 821 - Simple Mail Transfer Protocol*.

5.2.1. Modelo del SMTP

Desde el punto de vista del modelo, el SMTP debe proporcionar los elementos necesarios para la transferencia de mensajes. Por ello, se definen los elementos siguientes:

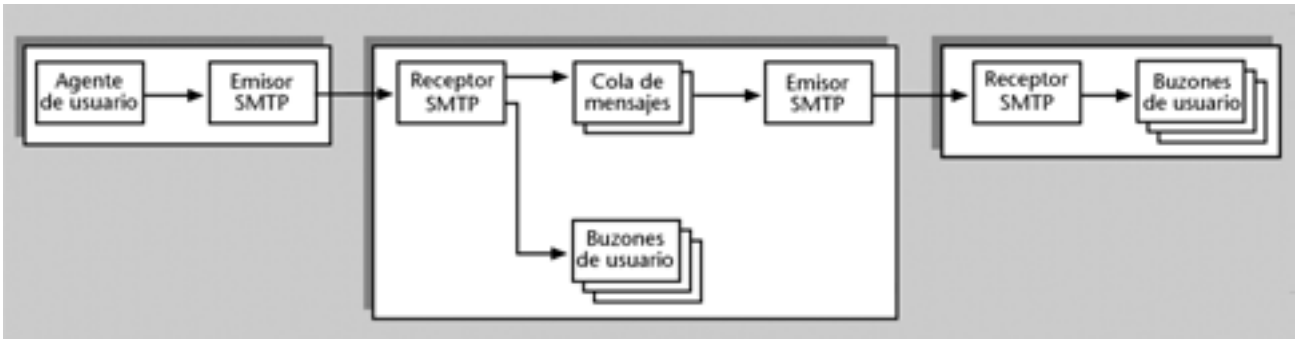
- **Agente de usuario**: se encarga de introducir los mensajes en el sistema de correo SMTP.
- **Emisor SMTP**: se ocupa de realizar las conexiones y de enviar mensajes a receptores SMTP a partir de peticiones de los usuarios. Generalmente, cada emisor SMTP tiene asociada una **cola de mensajes**, en la que se almacenan los mensajes antes de ser reenviados (siguiendo la filosofía del almacenamiento y el reenvío).
- **Receptor SMTP**: se encarga de recibir los mensajes. Si el mensaje va destinado a un usuario asociado al mismo sistema en que se encuentra el receptor SMTP, éste deposita el mensaje en el buzón del destinatario. En caso contrario, el receptor SMTP deposita el mensaje en la cola de mensajes del emisor SMTP asociado, que lo recupera y lo reenvía hacia el receptor SMTP de otra oficina más próxima al destinatario.

Agente de usuario

El agente de usuario, definido en la mayoría de los estándares, es equivalente al elemento usuario especificado en el modelo cliente/servidor. Puede ser tanto otra aplicación, como una persona por medio de una interfaz de usuario.

Conviene destacar que los sistemas que actúan como oficina deben disponer al mismo tiempo de un receptor SMTP (para recibir los mensajes), de buzones de los usuarios y de un emisor SMTP (para poder reenviar los mensajes) con una cola de mensajes.

La figura siguiente nos muestra el modelo de un sistema SMTP:



Notad que este modelo, así como los otros que se estudiarán en este módulo, sigue el modelo cliente/servidor definido con anterioridad. Asimismo, conviene remarcar que cada estándar proporciona un nombre diferente a los elementos del modelo cliente/servidor. En el estándard que nos ocupa, el usuario equivale al agente de usuario, el cliente equivale al emisor SMTP y el servidor equivale al receptor SMTP.

5.2.2. Direcciones de correo

Para que el sistema de correo sea capaz de entregar un mensaje, se precisa algún mecanismo que permita definir direcciones para los buzones de los usuarios. En los protocolos Internet, la **dirección de buzón** está formada por una cadena que identifica a un usuario (persona, sistema o proceso) y una cadena que identifica el sistema (dominio) en que se encuentra el buzón.

Nombre de dominio

El nombre de dominio suele estar formado por la secuencia de nombres de los subdominios de los que depende jerárquicamente separados por el carácter ".".

```
dirección = usuario@dominio
dominio = subdominio*(.subdominio)
```

Con este tipo de direccionamiento electrónico, se tiene la funcionalidad siguiente:

- El mensaje se envía al sistema identificado por el nombre de dominio que se encuentra en la dirección a la derecha del signo `@` (es decir, `dominio`).
- Una vez en el sistema, el mensaje se entrega al buzón del usuario identificado en la dirección a la izquierda del signo `@` (es decir, `usuario`).
- El SMTP proporciona también la posibilidad de definir **listas de correo**, que constituyen listas de destinatarios identificadas por una única dirección. Esta última es la que se utiliza para enviar mensajes a la lista, y el sistema SMTP se encarga de expandirla y enviar el mensaje a todos los usuarios que sean miembros de la misma en aquel momento. Este método permite la modificación de la lista sin necesidad de cambiar la dirección.

Buzones y dominios

Cada ordenador que actúa como oficina de correos suele definir un dominio de correo, que se identifica con un nombre de dominio. Este nombre es el que se encuentra en la parte ***`dominio`*** de la dirección electrónica de los usuarios que tienen los buzones en estas máquinas.

5.2.3. Envío de correo y mensajes a terminales

Además del envío de mensajes de correo a buzones (en el estándar, `mail`), que es su propósito principal, el SMTP también soporta la entrega de mensajes al terminal del destinatario (en el estándar, `send`). Como la implementación de estos dos métodos es muy similar, el SMTP define comandos en que combina.

5.2.4. Conceptos básicos del SMTP

El SMTP está basado en conexiones TCP y el puerto que tiene asignado es el 25.

Como hemos comentado al describir el modelo, el emisor SMTP hace llegar mensajes de correo al receptor. Para conseguirlo, se establece un diálogo entre los dos con **comandos** que envía al emisor y **respuestas** con las que contesta el receptor.

Tanto los comandos como las respuestas SMTP siguen las reglas específicas del protocolo Telnet. Es decir, constituyen cadenas de caracteres codificados con bytes según el código ASCII y se utiliza la secuencia <CRLF> para representar el final de línea.

- Los comandos SMTP están formados por un código constituido por cuatro caracteres alfanuméricos y, según los comandos, un espacio y una serie de parámetros.
- Las respuestas SMTP están formadas por un código numérico de tres dígitos que, habitualmente, va seguido de un texto descriptivo.

En los apartados siguientes se describirán los comandos definidos y las posibles respuestas.

5.2.5. Funcionalidad del SMTP

Es preciso diferenciar entre funcionalidad básica y funcionalidad adicional.

Funcionalidad básica

Una vez conectado, el emisor SMTP se identifica ante el receptor SMTP con el comando `HELO`.

HELO ***dominio***

Cuando se quiere iniciar el envío de un mensaje de correo, se utiliza el comando `MAIL`, que incluye la identificación del sistema desde el que se envía el mensaje. Este comando da lugar al campo `FROM:` del mensaje.

MAIL FROM: ***originador***

Con el comando `RCPT` se identifican los receptores del mensaje. Se debe utilizar uno para cada receptor, y cada llamada da lugar a un campo de cabecera `TO:` en el mensaje.

RCPT TO: ***receptor***

El comando `DATA` indica el inicio del envío del cuerpo del mensaje. Las líneas siguientes a este comando se tratan como contenido del mensaje. Este último se acaba con una línea que sólo incluye un punto; es decir, con la secuencia `<CRLF>.<CRLF>`.

DATA

Los datos que se envían dentro de este campo son mensajes RFC 822, por lo que pueden incluir campos de cabecera en el inicio. Cuando ello sucede, entre

los campos de cabecera y el cuerpo del mensaje debe haber una línea en blanco, es decir, la secuencia <CRLF><CRLF>.

Una vez iniciada la transacción de envío de mensaje, y antes de acabar, el emisor SMTP siempre puede interrumpirla por medio del comando RSET.

RSET

El comando NOOP sirve para que el receptor SMTP envíe una respuesta afirmativa para informar de que la conexión todavía está abierta.

NOOP

Para cerrar el canal de transmisión, el SMTP proporciona el comando QUIT. Cuando este último llega al receptor SMTP, éste envía una respuesta afirmativa y cierra el canal de transmisión.

QUIT

Funcionalidad adicional

El protocolo STMP posibilita también una funcionalidad adicional con el objetivo de enviar mensajes a terminales.

Para iniciar la entrega de un mensaje, o varios, a terminales, el emisor SMTP dispone del comando SEND.

SEND FROM: ***originador***

El comando SOML permite iniciar la entrega de un mensaje, o varios, a terminales si el usuario se encuentra en el terminal, o, en caso contrario, permite entregar el correo al buzón o a los buzones.

SOML FROM: ***originador***

El comando SAML permite iniciar la entrega de un mensaje, o varios, a terminales y, al mismo tiempo, al buzón o a los buzones.

SAML FROM: ***originador***

El emisor SMTP también puede pedir la confirmación de que una cadena identifica a un usuario por medio del comando VRFY.

VRFY cadena

El comando EXPN permite solicitar confirmación para una cadena que identifica una lista de correo y, en caso de respuesta positiva, requerir la relación de los miembros de la lista.

EXPN cadena

El comando HELP permite pedir ayuda al receptor SMTP sobre los comandos que soporta. En caso de incluir una cadena como argumento, esta última puede identificar un comando, del que se retorna información específica.

HELP [*cadena*]

El comando TURN permite cambiar el papel de emisor SMTP y receptor SMTP. El resultado de la petición puede ser que el receptor SMTP envíe una respuesta afirmativa y haga el papel de emisor SMTP, o que envíe una respuesta negativa y mantenga su papel.

TURN

5.2.6. Códigos de respuesta

El SMTP define una serie de códigos de respuesta para los diferentes comandos, tanto en caso de éxito (E), como en caso de resultado intermedio (I), de fallo (F) y de error (X). La tabla siguiente recoge estos códigos:

Codigos de respuesta															
	Establecimiento de connexión	HELO	MAIL	RCPT	DATA	RSET	SEND	SOML	SAML	VRFY	EXPN	HELP	NOOP	QUIT	TURN
211 Estatus del sistema o respuesta a petición de ayuda												E			
214 Mensaje de ayuda												E			
220 El servidor está preparado	E														
221 El servidor está cerrando el canal de transmisión														E	
250 Acción completada correctamente		E	E	E	E	E	E	E	E	E	E	E	E		E
251 Usuario no local, el mensaje se reenviará				E						E					
354 Principio de entrada de mensaje; es preciso acabar con <CRLF>.<CRLF>					I										
421 Servicio no disponible; cierre del canal de transmisión	F	X	X	X	X	X	X	X	X	X	X		X		
450 Acción no ejecutada: buzón no accesible				F											
451 Acción abortada por error local			F	F	F		F	F	F						
452 Acción abortada por capacidad de almacenamiento insuficiente			F	F	F		F	F	F						
500 Error de sintaxis; comando no reconocido		X	X	X	X	X	X	X	X	X	X	X	X	X	X
501 Error de sintaxis en los parámetros o argumentos		X	X	X	X	X	X	X	X	X	X	X			
502 Comando no implementado							X	X	X	X	X	X			F
503 Secuencia de comandos errónea				X	X										X
504 Parámetro de comando no implementado		X				X				X	X	X			
550 Acción no ejecutada: buzón no encontrado				F						F	F				
551 Usuario no local; intentar otra dirección				F						F					
552 Acción abortada por exceso de capacidad almacenada			F	F	F		F	F	F						
553 Acción no ejecutada; nombre de buzón no permitido				F						F					
554 Fallo en la transacción					F										

El estándar establece el conjunto de comandos mínimo que deben soportar todos los receptores SMTP:

- `HELO` *dominio*
- `MAIL FROM:` *originador*
- `RCPT TO:` *receptor*
- `DATA`
- `RSET`
- `NOOP`
- `QUIT`

5.2.7. Extensiones SMTP para mensajes de 8 bits

Se consideró oportuno añadir un mecanismo para extender el SMTP. Cuando un cliente SMTP desea utilizar las extensiones SMTP, así como las extensiones para mensajes de 8 bits, tiene que solicitarlo al receptor SMTP con el comando `EHLO` en lugar del comando `HELO`:

```
EHLO dominio
```

Si el receptor SMTP soporta las extensiones, retorna una respuesta o más de éxito (250) e indica las extensiones de fallo (550) o de error (501) que soporta. Si el receptor no las soporta, retorna una respuesta de error (500, 502, 504 ó 421).

Una de las funcionalidades adicionales es la posibilidad de enviar mensajes de 8 bits. Cuando el servidor acepta mensajes de 8 bits, la respuesta de éxito (250) incluye la cadena `8BITMIME`. Cuando se quiera enviar el mensaje, debe indicarse que es de 8 bits. Ello se aplica a los comandos `MAIL`, `SEND`, `SAML` o `SOML` de la manera siguiente:

```
MAIL FROM: originador BODY = valor-cuerpo
SEND FROM: originador BODY = valor-cuerpo
SAML FROM: originador BODY = valor-cuerpo
SOML FROM: originador BODY = valor-cuerpo
valor-cuerpo = 7BIT | 8BITMIME
```

Lecturas complementarias

Para saber más sobre las extensiones SMTP para mensajes de 8 bits, consultad las obras siguientes:

J. Klensin; N. Freed; M. Rose; E. Stefferud; D. Crocker (1995, noviembre). *RFC 1869 - SMTP Service Extensions*.

J. Klensin; N. Freed; M. Rose; E. Stefferud; D. Crocker (1994, julio). *RFC 1652 - SMTP Service Extension for 8bit-MIME transport*.

5.2.8. Ejemplo

En este apartado se presenta un ejemplo en el que se puede observar la secuencia de comandos que envía un emisor SMTP (en negrita) y las respuestas del receptor SMTP en una transacción de envío de correo:

```
220 peru.uoc.es SMTP/smap Ready.
HELO campus.uoc.es
250 (campus.uoc.es) pleased to meet you.
HELP
```

```
214-Commands
214-HELO    MAIL    RCPT    DATA    RSET
214 NOOP    QUIT    HELP    VRFY    EXPN
NOOP
220 OK
EXPN xc@campus.uoc.es
250-Jordi Inyigo <jinyigo@uoc.edu>
250-Jose Barcelo<jbarcelo@uoc.edu>
250-Llorenc Cerda <lcerda@uoc.edu>
250-Ramon Marti <rmarti@uoc.edu>
250-Enric Peig <epeig@uoc.edu>
250 Xavier Perramon <xperramon@uoc.edu>
MAIL FROM: jinyigo@uoc.edu
250 jinyigo@uoc.edu... Sender OK
RCPT TO: rmarti@uoc.edu
250 rmarti@uoc.edu OK
RCPT TO: rmarti@uoc.edu
501 Syntax error
RCPT TO: xperramon@uoc.edu
250 xperramon@uoc.edu OK

DATA
354 Enter mail, end with "." on a line by itself
Subject: Redes de computadores
Date: 20 Jun 2003
Esto es un mensaje de correo de ejemplo.
.
250 Mail accepted
QUIT
250 Closing connection
```

Telnet como cliente genérico de los protocolos

El programa telnet está pensado para hacer de cliente de servidores del protocolo Telnet. Sin embargo, si se considera que lo que hace es enviar al servidor cadenas de caracteres tal como se introducen por el teclado y volver por pantalla lo que recibe el servidor, también se puede utilizar como cliente SMTP* puesto que, como hemos comentado al describirlo, los mensajes que se intercambian en este protocolo son cadenas de caracteres ASCII.

*** El mismo razonamiento se puede aplicar al resto de los protocolos que veremos: POP3, IMAP, NNTP y HTTP.**

De este modo, si se utiliza el programa telnet como cliente SMTP, se pueden enviar comandos SMTP al servidor escribiéndolos por el teclado y viendo sus respuestas por pantalla. Para ello, el programa telnet admite como segundo parámetro el número de puerto donde debe conectarse.

Por ejemplo, el diálogo anterior se podría haber realizado haciendo:

telnet peru.uoc.es 25

5.3. Acceso simple a los buzones de correo: el POP3

En sistemas pequeños no es práctico, ni usual, soportar el SMTP, puesto que implica tener el sistema conectado y dispuesto a recibir mensajes en cualquier momento. Por este motivo, se vio la necesidad de definir un protocolo que permitiera la recuperación de mensajes de buzones de correo remotos y se definió el **POP3***.

*** Recordad que POP3 corresponde a post office protocol - version 3.**

POP/POP2

El POP y el POP2, dos protocolos definidos en las RFC 918 y RFC 937 respectivamente, parten de la misma filosofía que el POP3; sin embargo, disponían de un conjunto de comandos más reducido.

No obstante, en este protocolo es preciso disponer de sistemas en los que se encuentren los buzones –un servidor POP3–, y deben estar conectados en todo momento, tanto para recibir los mensajes, como para recibir las peticiones de acceso a los buzones. Por lo que respecta a los clientes POP3, sólo es necesario que se conecten cuando quieran acceder a su correo.

Lectura complementaria

Si queréis más información sobre el POP3, consultad la obra siguiente:

J. Myers; M. Rose (1996, mayo). *RFC 1939 - Post Office Protocol - Version 3*.

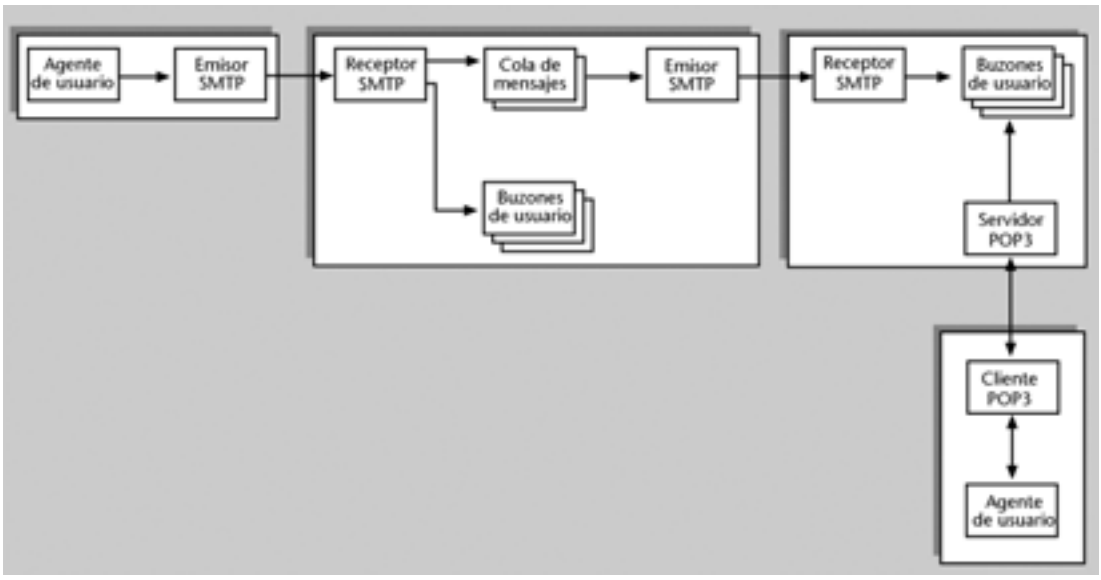
El POP3 no especifica ningún método para el envío de correo; otros protocolos de transferencia de correo, como el SMTP, proporcionan esta funcionalidad.

5.3.1. Modelo del POP3

El modelo funcional del POP3 se basa en los elementos siguientes:

- **Agente de usuario**: utiliza el cliente POP3 para acceder a su correo.
- **Cliente POP3**: se comunica con el servidor POP3 por medio del protocolo POP3 para acceder a su buzón de correo.
- **Servidor POP3**: recibe peticiones de los clientes POP3 y se las sirve accediendo a los buzones correspondientes.

En la figura siguiente se presentan los elementos del modelo funcional del POP3 integrados en un sistema en el que se utiliza el SMTP para enviar el correo, y el POP3 para acceder a los buzones:



5.3.2. Conceptos básicos del POP3

El POP3 se basa en comunicaciones TCP sobre el puerto 110.

El mecanismo normal de utilización de dicho protocolo es el siguiente: cuando el cliente POP3 necesita acceder al buzón, se conecta con el servidor POP3, recupera la información que le interesa y cierra la conexión.

Cada vez que sea necesario volver a acceder al buzón, se establece una nueva conexión.

- Los **comandos POP3** constituyen cadenas de caracteres ASCII imprimibles acabados con `<CRLF>`. Todos los comandos incluyen un código alfanumérico de cuatro caracteres que identifica el comando, seguido de cero o más parámetros.
- Las **respuestas POP3** también son cadenas de caracteres ASCII, y se representan con un indicador de estado positivo (`+OK`) o negativo (`-ERR`) y, posiblemente, información adicional, de la manera siguiente:

```
+OK el comando se ha ejecutado con éxito
-ERR el comando no se ha ejecutado con éxito
```

Estados

La norma define tres estados por los que debe pasar toda sesión POP3:

- Una vez se ha abierto la conexión, la sesión entra en el **estado de autorización**, en que el cliente debe identificarse ante el servidor POP3.
- Una vez autorizado, la sesión pasa al **estado de transacción**. En este último, el cliente pide acciones al servidor POP3 con los comandos necesarios.
- Cuando el cliente llama el comando `QUIT`, la sesión entra en el **estado de actualización**. El servidor libera los recursos, se despide y cierra la conexión TCP.

5.3.3. Funcionalidad del POP3

Desde el punto de vista de funcionalidad, y retomando los planteamientos del correo postal, el POP3 proporciona los comandos necesarios para el acceso a buzones de correo. A continuación, describiremos los comandos correspondientes a cada uno de los estados por los que debe pasar una sesión POP3:

1) Estado de autorización. En este estado el cliente se identifica ante el servidor POP3. Para realizar el proceso de identificación, se dispone de los comandos siguientes:

a) Identificación de usuario (USER)

Lo primero que debe hacer un cliente POP3 es identificarse ante el servidor POP3. Uno de los métodos es hacerlo mediante el comando USER, dentro del cual se envía el nombre que identifica al usuario.

```
USER nombre
```

b) Envío de la contraseña (PASS)

Una vez identificado el usuario mediante el comando USER, debe llevarse a cabo la autenticación por medio del envío de una contraseña con el comando PASS. El servidor utiliza la cadena enviada en este último junto con el nombre de usuario para dar acceso al usuario al buzón o denegárselo.

```
PASS contraseña
```

c) Identificación y autenticación del usuario con seguridad (APOP)

El método de identificación y autenticación mediante los comandos USER y PASS tiene el problema de que tanto el nombre como la contraseña viajan por la red sin ningún mecanismo de seguridad. Un método alternativo de identificación y autenticación del usuario consiste en utilizar el comando APOP, que incorpora mecanismos de seguridad en el envío de la contraseña.

```
APOP nombre resumen
```

El comando APOP

El comando APOP actúa de la manera siguiente:

Al conectarse, el servidor envía una cadena al cliente. Este último concatena la cadena recibida a la contraseña y crea un resumen de la misma por medio de un algoritmo de creación de resúmenes (*hash* criptográfico). Este resumen se envía junto con el nombre identificativo del usuario como argumentos del comando APOP.

El servidor, para verificar el ususario, genera el mismo resumen y lo compara con el que ha recibido.

2) Estado de transacción. En este estado, el cliente solicita acciones al servidor POP3. Las acciones que puede pedir son las siguientes:

a) Estado (STAT)

Una vez autenticado el usuario, el cliente POP3 puede requerir información sobre el estado del buzón del usuario por medio del comando STAT que no tiene argumentos y devuelve el número de mensajes y los bytes que ocupa el buzón del usuario.

```
STAT
```

b) Listado (LIST)

Cuando ya se sabe el número de mensajes que hay en el buzón, el paso siguiente es la petición de información sobre uno de los mensajes o sobre todos. El POP3 proporciona el comando LIST para esta tarea. Este comando devuelve, para cada mensaje, un número de mensaje y los bytes que ocupa.

```
LIST [mensaje]
```

c) Recuperación de mensajes (RETR)

Una vez se conocen los mensajes que hay en el buzón, deben recuperarse los que quiere leer. Ello puede hacerlo el cliente POP3, para cada mensaje, con el comando RETR, que incluye como argumento el identificador del mensaje que se quiere recuperar.

```
RETR mensaje
```

d) Borrado de mensajes (DELE)

Tras leer un mensaje, puede interesar borrarlo. El comando DELE indica al servidor POP3 que marque como mensaje a borrar el identificado como tal en el argumento; sin embargo, el mensaje no se borrará hasta que no se entre en el estado de actualización.

```
DELE mensaje
```

e) Operación nula (NOOP)

El comando NOOP sirve para saber si la conexión con el servidor todavía está abierta. Con dicho comando, el servidor POP3 no hace nada, excepto devolver una respuesta afirmativa.

```
NOOP
```

f) Desmarcado de mensajes para borrar (RSET)

Una vez se ha llevado a cabo una llamada al comando DELE, y antes de entrar en el estado de actualización, el usuario puede echarse atrás y pedir al servidor POP3, mediante el comando RSET, que desmarque todos los mensajes marcados para borrar.

```
RSET
```

g) Recuperación de la parte superior de un mensaje (TOP)

En ocasiones, puede interesar recuperar sólo la parte superior de un mensaje para decidir si vale la pena recuperarlo todo o no. El comando TOP permite recuperar la cabecera y *n* líneas del mensaje identificado en el argumento.

```
TOP mensaje n
```

h) Lista de identificadores únicos (UIDL)

Todos los mensajes del buzón tienen un identificador único (*unique-id*) permanente (a diferencia del número de mensaje, que es único dentro del buzón, pero que puede ir variando). El comando UIDL permite obtener el número de mensaje y el identificador único de uno de los mensajes del buzón o de todos.

```
UIDL [mensaje]
```

i) Paso al estado de actualización (QUIT)

Una vez finalizadas las transacciones, el cliente POP3 debe llamar el comando QUIT para pasar al estado de actualización.

```
QUIT
```

3) Estado de actualización. En este estado, el servidor POP3, en primer lugar, borra todos los mensajes marcados para borrar y, con posterioridad, libera to-

dos los recursos y cierra la conexión TCP. El estado de actualización no dispone de ningún comando asociado.

El estándar establece que los servidores POP3 deben soportar como mínimo los comandos siguientes:

```
USER nombre
PASS cadena
STAT
LIST [mensaje]
RETR mensaje
DELE mensaje
NOOP
RSET
QUIT
```

5.3.4. Ejemplo

A continuación, se presenta un ejemplo de los comandos de acceso de un cliente POP3 (en negrita) a un servidor POP3 para recuperar el mensaje enviado en el ejemplo anterior:

```
+OK QPOP (version 2.52) at pop.uoc.es starting.
USER rmarti
+OK Password required for rmarti.
PASS prueba
-ERR Password supplied for "rmarti" is incorrect.
+OK Pop server at pop.uoc.es signing off.
PASS password
+OK rmarti has 6 message(s) (190885 bytes).
STAT
+OK 6 190885
LIST
+OK 6 messages (190885 bytes)
1 3140
2 3326
3 1911
4 180846
5 861
6 801
.
RETR 6
+OK 801
```

```
Received: from campus.uoc.es by peru.uoc.es
  (8.8.5/8.8.5) with ESMTP id SAA14826
  for <rmarti@uoc.edu>; Fri, 27 Jun 2003 18:35:52
  +0200 (MET DST)
From: Jordi Inyigo <jinyigo@uoc.edu>
Message-Id: <199809211639.SAA20364@peru.uoc.es>
To: rmarti@uoc.edu, xperramon@uoc.edu
Subject: Redes de computadores
Date: 27 Jun 2003
Content-Type: text
Status: RO

Este es un mensaje de correo de ejemplo.
.
QUIT
+OK Pop server at dns signing off
```

5.4. Acceso complejo a los buzones de correo: el IMAP4rev1

El protocolo de acceso a mensajes Internet, versión 4rev1, IMAP4rev1, permite al cliente acceder a los mensajes de correo electrónico de un servidor y manipularlos.

El IMAP4rev1 (a partir de ahora lo llamaremos *IMAP4*) permite al usuario disponer de diferentes buzones estructurados de manera jerárquica y, al mismo tiempo, poderlos manipular de manera remota, tal como se hace con los buzones locales.

El IMAP4 también proporciona a los clientes la capacidad de resincronización con el servidor.

IMAP2 / IMAP3 / IMAP4

El IMAP4rev1 surgió de la evolución de las especificaciones IMAP2 [RFC 1176], IMAP3 [RFC 1203] e IMAP4 [RFC 1730].

El IMAP4 no especifica ningún método para el envío de correo; otros protocolos de transferencia de correo, como el SMTP, proporcionan esta funcionalidad.

5.4.1. Modelo del IMAP4

El modelo funcional del IMAP4 se basa en los elementos que presentamos a continuación:

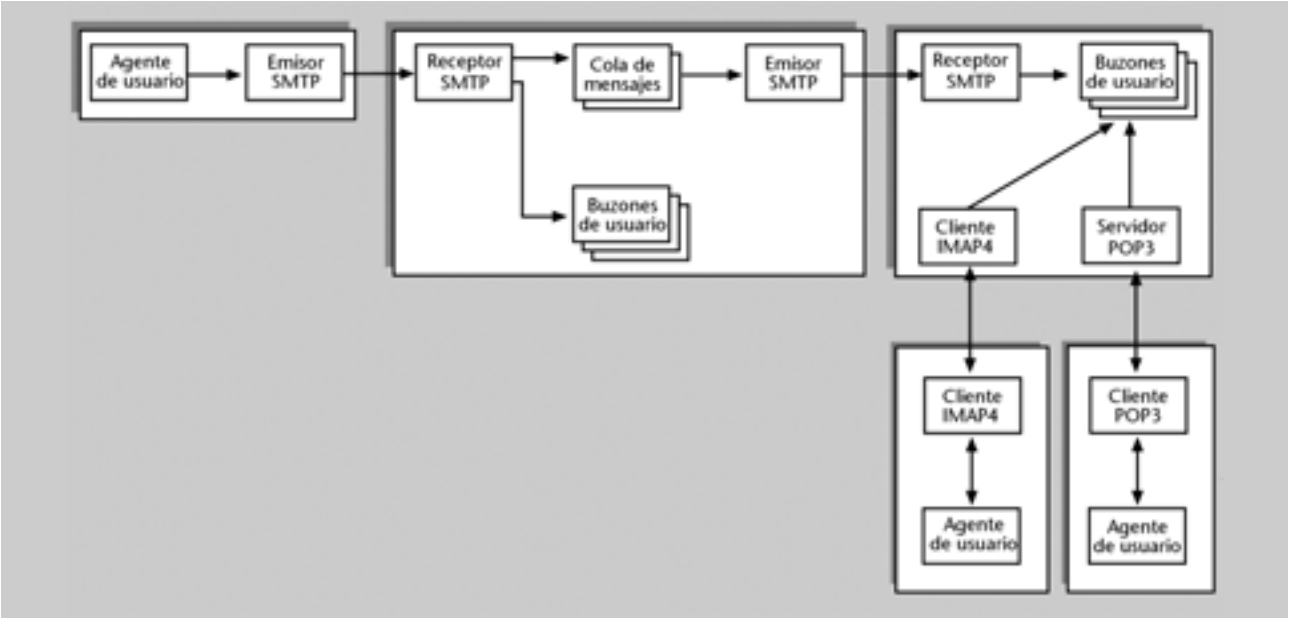
Lectura complementaria

Si queréis más información sobre el IMAP4rev1, consultad la obra siguiente:

M. Crispin (1996, deciembre). *RFC 2060 - Internet Message Access Protocol - Version 4rev1.*

- **Agente de usuario**: utiliza el cliente IMAP4 para leer el correo de su buzón.
- **Cliente IMAP4**: se comunica con el servidor IMAP4 por medio del IMAP4 para acceder a su buzón de correo.
- **Servidor IMAP4**: recibe peticiones de los clientes IMAP4 y se las sirve accediendo a los buzones correspondientes.

La figura siguiente presenta los elementos del modelo funcional del IMAP4 integrados en un sistema en el que se utiliza SMTP para enviar el correo e IMAP4 para acceder a los buzones:



5.4.2. Conceptos básicos del IMAP4

El IMAP4 puede utilizarse con cualquier protocolo de transporte fiable. Por norma general, se utiliza el TCP y, en este caso, se utiliza el puerto 143. Todas las interacciones entre cliente y servidor se llevan a cabo en forma de líneas ASCII acabadas con un carácter `<CRLF>`.

Cada comando del cliente empieza con un identificador (por lo general, una cadena alfanumérica corta) llamado ***tag***. El cliente debe generar un *tag* diferente para cada comando.

El servidor puede enviar datos tanto en respuesta a un comando del cliente, como de manera unilateral, y el cliente debe estar a punto para recibirlos en todo momento. Los datos transmitidos por el servidor hacia el cliente y las respuestas de estatus que no implican la finalización del comando empiezan con el *token*. La respuesta final de culminación de comando empieza con el mismo *tag* que el comando del cliente que ha dado lugar a la respuesta.

Además de las respuestas específicas de cada comando, casi todos los comandos disponen, como mínimo, de los **resultados de estatus** siguientes:

```
OK [Parámetros]: el comando se ha ejecutado.
NO [Parámetros]: el comando no se ha ejecutado.
BAD [Parámetros]: comando desconocido o argumentos in-
                  válidos.
```

Ejemplo

```
a006 logout
* BYE IMAP4rev1 server terminating connection
a006 OK LOGOUT completed
```

La primera línea es la llamada al comando precedida del *tag* (`a006`). En este caso, el comando `LOGOUT*`. Después, se puede observar la respuesta, que está formada por dos líneas. La última línea de la respuesta empieza con el mismo *tag* que la llamada, mientras que las líneas anteriores lo hacen con el *token*.

*** Consultad el comando LOGOUT en el subapartado 5.4.3 de este módulo didáctico.**

El cliente puede enviar un comando después de otro sin esperar el resultado del primero. De manera similar, un servidor puede empezar a procesar un comando antes de acabar de procesar el que se ejecuta en aquel momento.

Además del texto, cada mensaje tiene asociados diferentes atributos que se encuentran almacenados dentro del buzón y que son son los siguientes:

1) Identificador único (**UID**, *unique identifier*): a cada mensaje se le asocia un valor de 32 bits que, cuando se utiliza conjuntamente con los valores de validez de identificador único*, forma un valor de 64 bits que identifica de manera única un mensaje dentro de un buzón. Los UID se asignan de manera ascendente dentro del buzón, pero no necesariamente de manera contigua.

*** Los valores de validez de identificador único (*unique identifier validity value*) son unos identificadores que se asocian a cada buzón de manera única.**

2) Número de secuencia del mensaje (*message sequence number*): este atributo proporciona la posición relativa del mensaje dentro del buzón, desde el 1 hasta al número total de mensajes.

Esta posición debe ordenarse siguiendo los UID ascendentes. Los números de secuencia del mensaje se pueden reasignar durante la sesión (por ejemplo, cuando se elimina un mensaje del buzón).

3) Indicadores: cada mensaje tiene asociada una lista de cero indicadores, o más, que informan del estado:

- `\Seen`: mensaje leído
- `\Answered`: mensaje contestado
- `\Flagged`: mensaje marcado por atención urgente/especial
- `\Deleted`: mensaje marcado para ser borrado por un *Expunge* posterior
- `\Draft`: mensaje no editado del todo
- `\Recent`: mensaje acabado de llegar en esta sesión

4) Fecha interna (*internal date*): fecha y hora que cada mensaje lleva asociadas de manera interna dentro del servidor, que reflejan cuándo ha llegado el mensaje al servidor. No es la fecha del mensaje RFC 822.

5) Longitud [RFC 822] ([RFC 822] *size*): es el número de bytes del mensaje expresado en el formato RFC 822.

6) Estructura del sobre (*envelope structure*): representación analizada de la información del sobre RFC 822.

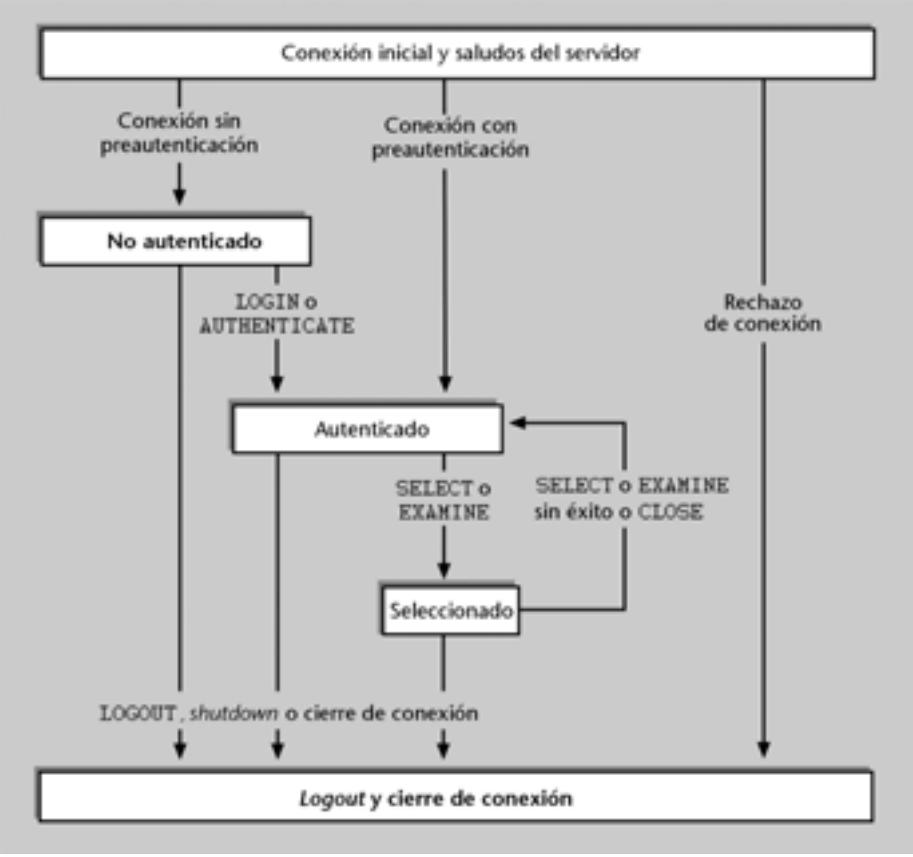
7) Estructura del cuerpo (*body structure*): representación analizada de la información de la estructura del cuerpo MIME.

8) Textos de mensaje: además de permitir la recuperación de los mensajes RFC 822 enteros, con el IMAP4 también se pueden efectuar recuperaciones parciales. En concreto, permite recuperar la cabecera y/o el cuerpo del mensaje RFC 822, una parte del cuerpo MIME o una cabecera MIME.

El IMAP4 especifica cuatro estados:

- **Estado no autenticado**: el cliente debe proporcionar sus credenciales. Se llega a este estado cuando se empieza una conexión, salvo que ya se haya preautenticado.
- **Estado autenticado**: el cliente debe seleccionar un buzón al que accederá antes de tener permiso para efectuar comandos sobre los mensajes.
- **Estado seleccionado**: se entra en este estado cuando se ha seleccionado con éxito un buzón.
- **Estado de *logout***: la conexión se acaba y el servidor la cerrará. Se puede entrar en este estado como resultado de una petición del cliente o de manera unilateral por parte del servidor.

La figura siguiente muestra el diagrama de flujo entre los diferentes estados definidos por el IMAP4:



Cada usuario tiene su correo en el servidor, depositado en un conjunto de buzones estructurados jerárquicamente que se pueden manipular remotamente a través del IMAP4.

Buzones IMAP4

Aunque el IMAP4 utiliza la palabra *buzón* para identificar todos los sitios donde se pueden guardar mensajes, en realidad lo que se entiende por *buzón donde se reciben los mensajes* es la bandeja de entrada (buzón *inbox*), mientras que los otros buzones son más bien carpetas en las que se clasifican estos mensajes recibidos.

Cada buzón se identifica por un nombre relativo, una cadena de caracteres que lo diferencia de los buzones hermanos. Asimismo, cada buzón dispone de un nombre que lo identifica dentro de la estructura formada por la secuencia de los nombres relativos de los buzones que definen los niveles de jerarquía superiores, de izquierda a derecha, separados con un único carácter (por norma general, el carácter "/"). Debe utilizarse el mismo separador en todos los niveles de la jerarquía dentro de un nombre.

5.4.3. Funcionalidad del IMAP4

La funcionalidad proporcionada por el IMAP4, como la del POP3, imita la que se requiere para el acceso a los buzones de correo postal. Como mejora respecto al POP3, el IMAP4 proporciona una estructura jerárquica del buzón en forma de carpetas, así como facilidades de suscripción, lo que da lugar a nuevos comandos que permiten gestionar todos estos elementos.

Las funciones que se utilizarán según el estado en que se encuentre la comunicación serán las siguientes:

1) Cualquier estado (comandos/estados universales). Sea cual sea el estado de la conexión, se pueden utilizar los comandos siguientes:

a) Petición de capacidades (`CAPABILITY`)

El comando `CAPABILITY`, que no tiene ningún argumento, sirve para solicitar la lista de capacidades que soporta el servidor

```
CAPABILITY
```

b) Operación nula (`NOOP`)

El comando `NOOP` permite al cliente IMAP4 averiguar si la conexión con el servidor todavía está abierta.

```
NOOP
```

c) Finalización de conexión (`LOGOUT`)

El comando `LOGOUT` permite al cliente notificar al servidor que quiere acabar la conexión.

```
LOGOUT
```

2) Estado no autenticado. En este estado, un cliente proporciona sus credenciales; para ello, se utilizan los comandos siguientes:

a) Indicador de autenticación (`AUTHENTICATE`)

El comando `AUTHENTICATE` sirve para indicar al servidor IMAP4 un mecanismo de autenticación; es decir, cuál de los diferentes mecanismos de autenticación posibles utiliza el cliente.

`AUTHENTICATE` *tipo_autenticación*

b) Identificación de usuario (`LOGIN`)

Como todo protocolo, el IMAP4 proporciona un comando para permitir que el cliente se identifique ante el servidor por medio del envío de un identificador de usuario y una contraseña. Este comando es `LOGIN`.

`LOGIN` *id_usuario contraseña*

3) Estado autenticado. En este estado el IMAP4 proporciona algunas funcionalidades nuevas:

a) Selección de un buzón (`SELECT`)

Una vez autenticado, el cliente IMAP4 puede seleccionar un buzón para acceder a sus mensajes. El comando `SELECT` proporciona esta funcionalidad.

`SELECT` *buzón*

Este comando, además de los resultados de estatus, retorna las respuestas obligatorias sin *tag* siguientes:

```
FLAGS: informa de los identificadores que se pueden
       aplicar al buzón del comando.
EXISTS: número de mensajes existentes en el buzón.
RECENT: número de mensajes con el indicador \Recent.
```

También puede retornar las respuestas `OK` sin *tag*:

```
NSEEN: número del primer mensaje sin el indicador \Seen.
PERMANENTFLAGS: lista de indicadores que pueden
                modificarse permanentemente.
```

b) Examen de un buzón (`EXAMINE`)

El comando `EXAMINE` permite un acceso al buzón similar al del comando `SELECT`, pero de manera que el buzón sea sólo de lectura. Este comando, además de los resultados de estatus, puede retornar las mismas respuestas que el comando `SELECT`.

`EXAMINE` *buzón*

c) Creación de un buzón (`CREATE`)

Al usuario le puede interesar crear un buzón con un nombre determinado. El IMAP4 proporciona el comando `CREATE` para ello.

`CREATE` *buzón*

d) Borrado de un buzón (`DELETE`)

El IMAP4 también facilita la eliminación permanente de un buzón con un nombre determinado por medio del comando `DELETE`.

`DELETE` *buzón*

e) Renombramiento de un buzón (`RENAME`)

En ocasiones, el usuario desea cambiar el nombre de un buzón por uno nuevo. El cliente IMAP4 puede solicitar esta acción al servidor por medio del comando `RENAME`.

`RENAME` *buzón nuevobuzón*

f) Suscripción de un buzón (`SUBSCRIBE`)

En el IMAP4, no es preciso que todos los buzones estén activos a cada momento. Con el comando `SUBSCRIBE`, el nombre del buzón pasa a formar parte de la lista de buzones activos o suscritos.

`SUBSCRIBE` *buzón*

g) Eliminación de la suscripción de un buzón (`UNSUBSCRIBE`)

El comando `UNSUBSCRIBE` permite desactivar un buzón eliminando su nombre de la lista de buzones activos o suscritos.

`UNSUBSCRIBE` *buzón*

h) Listado de buzones (`LIST`)

El comando LIST permite obtener los nombres de buzones que cumplen los criterios de búsqueda deseados dentro de un buzón de referencia. Este comando, además de los resultados de estatus, puede retornar la respuesta sin *tag* siguiente:

`LIST` *buzón 1*criterio_búsqueda*

```
LIST: nombre del buzón que cumple los criterios
      de búsqueda deseados.
```

Puede haber más de una respuesta `LIST`.

i) Listado de buzones suscritos (`LSUB`)

El comando `LSUB` permite obtener los nombres de los buzones activos o suscritos que cumplen los criterios de búsqueda deseados dentro de un buzón de referencia. Este comando, además de los resultados de estatus, puede retornar la respuesta sin *tag* siguiente:

`LSUB` *buzón 1*criterio_búsqueda*

```
LSUB: nombre del buzón que cumple los criterios
      de búsqueda deseados.
```

Puede haber más de una respuesta `LSUB`

j) Estado del buzón (`STATUS`)

El comando `STATUS` permite conocer el estado de un buzón.

`STATUS` *buzón (1#atributo_estado)*

Los atributos de estado definidos son los siguientes:

- `MESSAGE`: número de mensajes en el buzón.
- `RECENT`: número de mensajes con el indicador `\Recent`.
- `UIDNEXT`: UID que se asignará al mensaje siguiente que llegue al buzón.
- `UIDVALIDITY`: valor del UID del buzón.
- `UNSEEN`: número de mensajes sin el indicador `\Seen`.

Este comando, además de los resultados de estatus, puede retornar la respuesta sin *tag* siguiente:

```
STATUS: nombre de buzón que cumple el estatus deseado y
        la información de estatus requerida.
```

k) Añadido de un mensaje al buzón (`APPEND`)

El comando `APPEND` permite al cliente el IMAP4 añadir un texto literal como nuevo mensaje al final de un buzón seleccionado, con la fecha, la hora y los indicadores deseados.

`APPEND` *buzón[lista_flags] [fecha_hora] literal*

4) Estado seleccionado. En el estado seleccionado, el IMAP4 proporciona nuevas funcionalidades, además de los comandos universales y los del estado autenticado:

a) Control del buzón (`CHECK`)

El comando `CHECK` permite al cliente IMAP4 pedir al servidor un punto de control del buzón seleccionado en un momento determinado.

`CHECK`

b) Cierre del buzón (`CLOSE`)

El comando `CLOSE` permite cerrar un buzón seleccionado y eliminar permanentemente todos sus mensajes que tienen el indicador `\Deleted`.

`CLOSE`

c) Eliminación de mensajes (`EXPUNGE`)

El comando `EXPUNGE` permite eliminar de manera permanente todos los mensajes que tienen el indicador `\Deleted` del buzón seleccionado y sin necesidad de cerrarlo. Este comando, además de los resultados de estatus, puede retornar la respuesta sin *tag* siguiente:

`EXPUNGE`

```
EXPUNGE: número de secuencia de mensaje especificado que
         ha sido eliminado permanentemente.
```

d) Búsqueda de mensaje (SEARCH)

El comando SEARCH permite al cliente IMAP4 buscar dentro del buzón los mensajes que contienen un conjunto de caracteres especificado y que cumplen unos criterios de búsqueda deseados. Este comando, además de los resultados de estatus, puede retornar la respuesta sin *tag* siguiente:

```
SEARCH [CHARSET
   conjunto_caracteres]
   1#criterio_búsqueda
```

```
SEARCH: un número de secuencia o más de los mensajes que
        cumplen el criterio de búsqueda deseado.
```

e) Recuperación de mensajes (FETCH)

El comando FETCH permite la recuperación de un conjunto de mensajes (total o parcialmente), especificando unos atributos de recuperación. Este comando, además de los resultados de estatus, puede retornar la respuesta sin *tag* siguiente:

```
FETCH conjunto_mensajes ALL |
   FULL | FAST |
   atrib_recup | (1#atrib_recup)
```

```
FETCH: información del mensaje que presenta los atribu-
       tos de recuperación especificados.
```

f) Modificación de almacén (STORE)

El comando STORE permite modificar los indicadores del almacén que proporcionan los atributos a un conjunto de mensajes del buzón. Este comando, además de los resultados de estatus, puede retornar la respuesta sin *tag* siguiente:

```
STORE conjunto_mensajes
   indicadores_atrib_almacén
```

```
FETCH: información de los mensajes.
```

g) Copia de mensaje(s) (COPY)

El comando COPY permite copiar un conjunto de mensajes al final de un buzón especificado.

```
COPY conjunto_mensajes buzón
```

h) Retorno de identificador único (UID)

La sentencia UID, seguida de los parámetros COPY, FETCH, STORE o SEARCH, en lugar de retornar el número o números de secuencia de mensaje, retorna sus identificadores únicos. Este comando, además de los resultados de estatus, puede retornar las respuestas sin *tag* siguientes:

```
UID (COPY ... | FETCH ... |
  SEARCH ... | STORE ...)
```

```
FETCH:  información del mensaje que sigue los atributos
        de recuperación especificados.
SEARCH: un UID o más indican los mensajes que cumplen el
        criterio de búsqueda deseado.
```

5) Experimental/expansión

Comando experimental (X<atom>)

El IMAP4 permite especificar comandos experimentales que no son una parte de la especificación, siempre que su nombre empiece con el prefijo X.

X1*carácter*

5.4.4. Ejemplo

A continuación, se presenta un ejemplo de los comandos de un cliente IMAP4 (en negrita) que accede a un servidor IMAP4 para entrar en sus buzones. En el ejemplo se ve cómo se recupera la información del mensaje 12 (`fetch 12 full`). En la información retornada por el servidor, se ve la información "parseada" de la cabecera. A continuación, se recupera toda la cabecera del mismo mensaje (`fetch 12 body [header]`). Al final, antes de cerrar la conexión, se marca el mensaje para que sea borrado.

```
* OK IMAP4rev1 Service Ready
a001 login rmarti secret
a001 OK LOGIN completed
a002 select inbox
* 18 EXISTS
* FLAGS (\Answered \Flagged \Deleted \Seen \Draft)
* 2 RECENT
* OK [UNSEEN 17] Message 17 is the first unseen message
* OK [UIDVALIDITY 3857529045] UIDs valid
a002 OK [READ-WRITE] SELECT completed
a003 fetch 12 full
* 12 FETCH (FLAGS (\Seen) INTERNALDATE "27-Jun-2003
02:44:25 -0700" RFC 822.SIZE 4286 ENVELOPE
("Fri, 27 Jun 2003 02:23:25 -0700 (PDT)"
"Ejemplo de acceso IMAP4rev1"
(("Jordi Inyigo" NIL "jinyigo" "uoc.edu"))
(("Jordi Inyigo" NIL "jinyigo" "uoc.edu"))
(("Jordi Inyigo" NIL "jinyigo" "uoc.edu"))
((NIL NIL "rmarti" "uoc.edu"))
((NIL NIL "xperramon" "uoc.edu")
("JM Marques" NIL "jmmarques" "uoc.edu"))
NIL NIL
“<B27397-0100000@uoc.edu>")
BODY ("TEXT" "PLAIN" ("CHARSET" "US-ASCII") NIL NIL
  "7BIT" 3028 92))
a003 OK FETCH completed
a004 fetch 12 body[header]
* 12 FETCH (BODY[HEADER] {350}
Date: Fri, 27 Jun 2003 02:23:25 -0700 (PDT)
From: Jordi Inyigo <jinyigo@uoc.edu>
```

```
Subject: Ejemplo de acceso IMAP4rev1
To: rmarti@uoc.edu
cc: xperramon@uoc.edu,
    JM Marques <jmmarques@uoc.edu>
Message-Id: <B27397-0100000@uoc.edu>
MIME-Version: 1.0
Content-Type: TEXT/PLAIN; CHARSET=US-ASCII

)
a004 OK FETCH completed
a005 store 12 +flags \deleted
* 12 FETCH (FLAGS (\Seen \Deleted))
a005 OK +FLAGS completed
a006 logout
* BYE IMAP4rev1 server terminating connection
a006 OK LOGOUT completed
```

5.5. Extensiones multimedia: el formato MIME

La norma RFC 822 define un formato de mensaje y un contenido con una única parte de texto en ASCII de 7 bits. Se vio que este formato era muy pobre y que se precisaba algún método para superar sus limitaciones.

El formato **MIME*** redefine el formato del mensaje para permitir, sin perder la compatibilidad con el formato definido por el RFC 822, las características siguientes:

*** MIME es la sigla de *multipurpose Internet mail extensions*.**

- Contenido de texto no sólo ASCII de 7 bits.
- Contenido no texto.
- Contenido con múltiples partes.
- Cabeceras con texto no sólo ASCII de 7 bits.

5.5.1. Nuevos campos de cabecera

Para permitir todas las nuevas extensiones, el MIME define nuevos campos de cabecera: `MIME-Version`, `Content-Type`, `Content-Transfer-Encoding`, `Content-ID` y `Content-Description`.

Indicador de versión (`MIME-Version`)

El campo de cabecera `MIME-Version` indica la versión MIME* que se utiliza en el mensaje. Este campo es útil para que el receptor del mensaje pueda interpretar los campos de cabecera MIME.

*** La versión de MIME que se utiliza actualmente, y que especificamos en este apartado, es la 1.0.**

```
MIME-Version: 1*digit.1*digit
```

Indicador del tipo y subtipo de datos (`Content-Type`)

El campo de cabecera `Content-Type` permite indicar los tipos y subtipos de los datos que se encuentran dentro del mensaje. De este modo, el receptor podrá conocer los tipos de datos que contiene el mensaje y podrá visualizarlos adecuadamente. Según el tipo o subtipo de dato, puede incluir algún parámetro adicional.

```
Content-Type: tipo/subtipo *(; parámetro)
tipo = tipo-discreto | tipo-compuesto
tipo-discreto = text | image | audio | video |
   application | extension-token
tipo-compuesto = message | multipart |
   extension-token
```

La norma define cinco tipos de contenido (*tipo-discreto*), con los subtipos correspondientes. Estos tipos corresponden a datos monomedia:

- `text`: para información de texto.
- `image`: para imágenes.
- `audio`: para sonido.
- `video`: para vídeo.
- `application`: para cualquier otro tipo de datos, por norma general en formato específico de alguna aplicación.

La norma también define dos tipos de datos compuestos (*tipo-compuesto*):

- `multipart`: para datos formados de muchas partes o para datos independientes dentro de un mismo mensaje.

  En los mensajes de tipo `multipart`, el parámetro ***boundary*** incluye una cadena, precedida por dos caracteres de guión, "--", que se utiliza como separador entre las diferentes partes del mensaje. Para indicar que se han acabado todas las partes, se utiliza la cadena del parámetro ***boundary***, precedida y seguida de dos caracteres de guión, "--".

- `message`: para encapsular otro mensaje dentro del mensaje.

La tabla siguiente presenta los valores de tipo, subtipo y parámetros para los `Content-Type` definidos en la norma:

Valores definidos para `Content-Type`		
Tipo	**Subtipo**	**Parámetros**
`text`	`plain`	`charset = ISO-8859-(1 | ... | 9) | us-ASCII`
	`enriched`	`charset = ISO-8859-(1 | ... | 9) | us-ASCII`

Lecturas complementarias

Para obtener más información sobre el formato de mensajes MIME, podéis consultar las obras siguientes:

N. Freed; N. Borenstein (1996, noviembre). *RFC 2045 - Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*.

N. Freed; N. Borenstein (1996, noviembre). *RFC 2046 - Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types*.

K. Moore (1996, noviembre). *RFC 2047 - MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text*.

N. Freed; J. Klensin; J. Postel (1996, noviembre). *RFC 2048 - Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures*.

N. Freed; N. Borenstein (1996, noviembre). *RFC 2049 - Multipurpose Internet Mail Extensions (MIME) Part Five: Conformance Criteria and Examples*.

Valores definidos para `Content-Type`		
Tipo	**Subtipo**	**Parámetros**
image	gif	–
	jpeg	–
audio	basic	–
video	mpeg	quicktime
application	octet-stream	type = *cadena*; padding = *entero*
	postscript	–
multipart	mixed	boundary = *cadena*
	alternative	boundary = *cadena*
	parallel	boundary = *cadena*
	digest	boundary = *cadena*
message	rfc 822	–
	partial	id = *cadena*; number = *entero* [;total = *entero*]
	external-body	access-type = ftp | anon-ftp | tftp | afs | local-file | mail-server [;expiration = *date-time*] [;size = *entero*] [;permission=read | read-write] [;name = *cadena*] [;site = *cadena*] [;dir = *cadena*] [;mode = netascii | octet | mail | ascii | ebcdic | image | localn] [;server = *cadena*] [;subject = *cadena*]

Especificador del tipo de codificación (`Content-Transfer-Encoding`)

En algunos protocolos, como el SMTP, la información que se envía debe ser en ASCII de 7 bits. En ocasiones, los datos originales no tendrán este formato y, entonces, será preciso aplicarles algún tipo de codificación antes de enviarlos.

El campo de cabecera `Content-Transfer-Encoding` sirve para especificar el tipo de codificación que se ha aplicado al contenido del mensaje para que el receptor pueda descodificarlo, si es preciso. La norma define diferentes tipos de codificación:

```
Content-Transfer-Encoding: mecanismo
mecanismo = 7bit | 8bit | binary | quoted-printable |
base64
```

a) Codificación `7bit | 8bit | binary`

Los mecanismos de codificación `7bit`, `8bit` y `binary` sólo sirven para indicar de qué tipo son los datos transmitidos. En estos casos, los datos no se codifican y, por norma general, se utilizan para aplicaciones que no restringen la representación de datos en 8 bits.

b) Codificación `quoted-printable`

El mecanismo de codificación `quoted-printable` se utiliza para la representación y codificación en ASCII de 7 bits de datos, la mayoría de los cuales ya es de bytes representables en este formato. Es decir, este mecanismo se aplica cuando la información es mayoritariamente de caracteres de texto. Las normas básicas de codificación son las siguientes:

- Cualquier byte se puede representar con el carácter "=" seguido por la notación hexadecimal en dos dígitos (y letras en mayúscula) del valor del byte.

- Los bytes con valores decimales entre 33-60 y 62-126, incluídos los cuatro, se pueden representar con el carácter ASCII correspondiente.

c) Codificación `Base64`

El mecanismo de codificación `Base64` ofrece una codificación de información binaria en ASCII de 7 bits que no deba ser legible. Los algoritmos de codificación y descodificación son muy simples.

El proceso de codificación se lleva a cabo tomando grupos de 3 bytes (24 bits). Manteniendo el orden de bits original, estos 24 bits se reagrupan en 4 bloques de 6 bits (6 bits = 64 combinaciones).

Codificación `Base64`

Conviene destacar que la codificación de datos en `Base64` utiliza 4 caracteres para cada 3 bytes. Por este motivo, `Base64` aumenta el tamaño de la información un 33% (4/3).

El fichero codificado se obtiene tomando cada uno de estos bloques de 6 bits y codificándolo como un carácter alfanumérico a partir del valor binario de los 6 bits, según la tabla siguiente:

Tabla de codificación `Base64`

Valor	Carácter	Valor	Carácter	Valor	Carácter	Valor	Carácter
0	'A'	26	'a'	52	'0'	62	'+'
1	'B'					63	'/'
...		...		...		pad	'='
25	'Z'	51	'z'	61	'9'		

Si debe codificarse un número de bytes que no sea múltiple de 3 (la codificación, por tanto, no sería múltiple de 4 bytes), se codifican todos los bytes y, al final de la cadena de caracteres ya codificada, se añaden tantos caracteres "=" (carácter para rellenar, *pad*) como sean necesarios (máximo dos) hasta llegar a un número de caracteres múltiplo de 4.

En el proceso de descodificación, se toman los caracteres alfanuméricos recibidos y se reconvierten en su valor binario correspondiente (6 bits) según la tabla. Cada 4 caracteres recibidos dan lugar a 24 bits, y sólo es preciso irlos reagrupando en 3 bytes para generar el fichero descodificado. Antes de descodificar todo el mensaje, es preciso eliminar todos los caracteres "=" que se encuentren al final.

Identificador del contenido del mensaje (`Content-ID`)

El campo de cabecera `Content-ID` se utiliza para proporcionar un identificador único al contenido del mensaje. Con dicho identificador, se hace referencia al contenido de manera no ambigua.

```
Content-ID: id-msg
```

Informador descriptivo del contenido (`Content-Description`)

El campo `Content-Description` proporciona información descriptiva del contenido en forma de texto.

```
Content-Description: texto
```

5.5.2. Extensiones para texto no ASCII en las cabeceras

El MIME también permite codificar texto no ASCII en las cabeceras de los mensajes RFC 822 de manera no ambigua. Este método permite codificar toda la cabecera o sólo una parte. El texto (que puede ser de más de un carácter) ya codificado, precedido por el conjunto de caracteres y un identificador del tipo de codificación que se ha aplicado, se incluye en el lugar correspondiente de la cabecera entre los caracteres "=?" y "?=":

```
=? charset ? codificación ? texto-codificado ?=
charset = ISO-8859-(1 | ... | 9) | us-ASCII
codificación = Q | B
```

a) Conjunto de caracteres (*charset*): es válido cualquiera de los caracteres permitidos en el `Content-Type text/plain`.

b) Codificaciones: el formato MIME permite dos tipos de codificación similares a las codificaciones que se pueden aplicar al cuerpo del mensaje:

- `Q`: es parecida al `Content-Transfer-Encoding quoted-printable`. Es la que se recomienda cuando la mayoría de los caracteres que deben codificarse son ASCII.
- `B`: es idéntica al `Content-Transfer-Encoding Base64`. Es adecuada para el resto de casos.

5.5.3. Mensajes multiparte

En los mensajes multiparte, cada una de las partes suele estar formada por una pequeña cabecera y el contenido. En la cabecera se pueden encontrar

los campos `Content-Type`, `Content-Transfer-Encoding`, `Content-ID` y `Content-Description`. Todos se refieren al contenido de la parte en cuestión.

5.5.4. Ejemplo

En este ejemplo se puede contemplar un mensaje RFC 822 con extensiones MIME versión 1.0. Es un mensaje de dos partes: la primera es de texto codificado con `quoted-printable` y la segunda es una imagen en formato .gif codificada en `Base64`. Asimismo, se puede ver un campo de cabecera con una letra *í* codificada con codificación `Q`.

La "í" de *Ramon Martí* se ha codificado con codificación `Q`.

```
Date: 27 Jun 2003 0932 PDT
From: Jordi Inyigo <jinyigo@uoc.edu>
To: Ramon =?iso-8859-1?Q?Mart=ED?=
  <rmarti@uoc.edu>
MIME-Version: 1.0
Content-Type: multipart/mixed;
  boundary="=_250699_"

--=_250699_
Content-Type: text/plain; charset="iso-8859-1"
Content-Transfer-Encoding: quoted-printable

Esto es un mensaje RFC 822 que contiene MIME.

--=_250699_
Content-Type: image/gif; name="Readme.gif"
Content-Transfer-Encoding: base64

R0lGODlhIAAgAIAAAAAAAP///yH5BAEAAAAALAAAAAAgACAAA
AJzjI+pywoQXoSywoaontzeRhnXKJYc82GOuaoL2brJlx7pg+
eSzc9yHAHqhpmi8ddLspCT3Y2pXNZ+UxpUpGNNZVsYdBsMf4l
BsMocuqLFyCHO5I6/MWY2sy7PL4F3+nU/kxcHePTl91dnR+WS
yAcimGVQAAA7

--=_250699_--
```

6. Servicio de noticias: el NNTP

El servicio de noticias* permite el envío de mensajes, como el servicio de correo electrónico, pero con la diferencia de que el originador no especifica el destinatario o destinatarios, sino que cualquier usuario con acceso al servicio puede leerlos. Esta funcionalidad, pues, se puede comparar a la de un tablón de anuncios, en el que todo el mundo puede leer los mensajes que se encuentran colgados.

*** El servicio de noticias se conoce también por la palabra inglesa *news*.**

El servicio de noticias en Internet se conoce con el nombre ***Usenet***, que proviene del de la red en que se desarrolló originariamente este servicio. En la actualidad, la distribución de noticias se ha extendido por todo Internet, de manera que los anuncios enviados se pueden leer en cualquier parte del mundo. En la práctica, sin embargo, es posible restringir el ámbito en que se quiere distribuir los mensajes, puesto que algunos sólo serán interesantes, por ejemplo, en una determinada zona geográfica.

6.1. El modelo NNTP

En el servicio de noticias, la distribución de los mensajes o anuncios, que en Usenet se denominan ***artículos***, se efectúa de manera descentralizada. Teniendo en cuenta el enorme volumen de tráfico que genera este servicio, no sería viable tener un servidor central en el que todo el mundo dejara sus artículos y fuera a leer los de los demás. Por este motivo, se utiliza un mecanismo de propagación en el que el autor de un artículo lo envía a un servidor de noticias, el cual se encargará de reenviarlo a una serie de servidores próximos o con los que esté conectado directamente, que a su vez lo reenviarán a otros servidores, y así sucesivamente.

Con este mecanismo de propagación, se consigue que un artículo esté disponible para ser leído, idealmente, en todos los servidores de la red. Los usuarios que deseen leer los artículos podrán hacerlo, pues, conectándose a cualquier servidor, preferiblemente al que tengan más próximo.

Por otro lado, los servidores sólo almacenan cada artículo durante cierto tiempo. Hay artículos que tienen una fecha de caducidad predeterminada, y el servidor los borra automáticamente en la misma. Otros se borran, por ejemplo, cuando el servidor decide que los usuarios que puedan estar interesados en los mismos ya han tenido bastante tiempo para leerlos.

El método utilizado para propagar los artículos tiene ciertas limitaciones; por ejemplo, no se puede garantizar que un artículo llegue a todos los servidores

en un plazo determinado (o, en ocasiones, que llegue a un servidor concreto), es preciso controlar el camino por donde pasa cada artículo para evitar bucles, etc. Sin embargo, este método es más práctico y eficiente que la solución de un solo servidor central.

Los servidores de noticias se comunican entre sí para intercambiarse artículos por medio del **NNTP***, especificado en el documento RFC 977.

*** NNTP es la sigla de** ***network news transfer protocol.***

Un usuario que desee acceder al servicio, tanto para enviar artículos como para leerlos, puede ser que tenga acceso directo a alguno de los servidores de noticias. Entonces, la comunicación con el servidor es un asunto local. El caso más general, sin embargo, es que el usuario quiera acceder al servicio desde otro sistema, que actuará como cliente. Entonces, la comunicación entre el cliente y el servidor se lleva a cabo también por medio del NNTP.

Antiguamente...

... se utilizaban otros métodos de transmisión de artículos entre servidores, como el UUCP (*Unix to Unix copy protocol*).

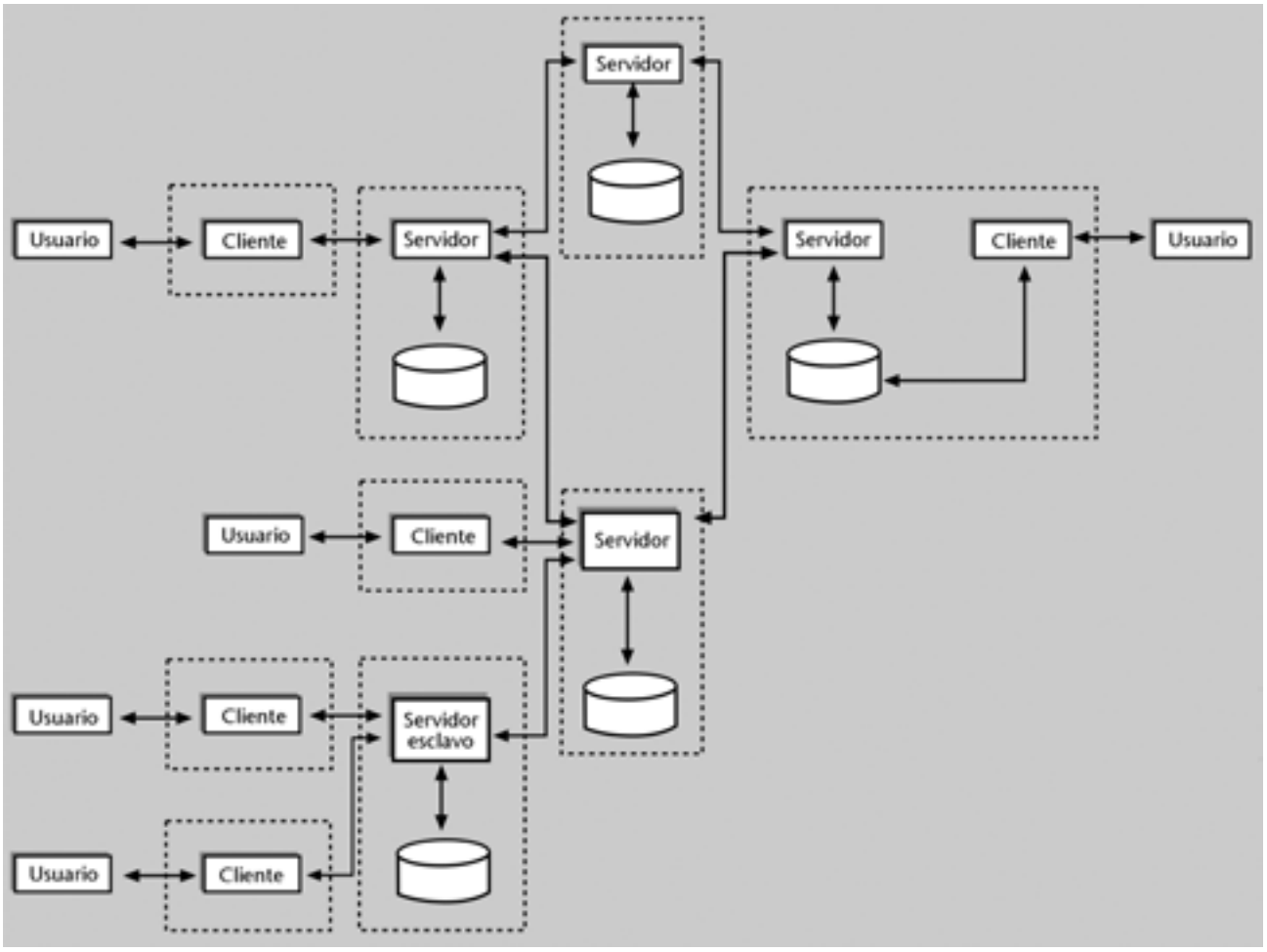
Tanto si el cliente es local, como si es remoto, por norma general habrá otro proceso encargado de la interfaz con el usuario. Este tipo de programa se suele llamar **lector de noticias**.

Lectura complementaria

Si queréis más información sobre el NNTP, consultad la obra siguiente:

B. Kantor; P. Lapsley (1986, febrero). *RFC 977 - Network News Transfer Protocol*.

Asimismo, existe la posibilidad de que un grupo de clientes acceda a un servidor central de una organización por medio de un servidor esclavo:



El servidor esclavo puede mejorar la eficiencia en el acceso, por ejemplo, almacenando localmente copias de los últimos artículos leídos, por si los solicitan otros clientes.

Los artículos disponibles en un servidor deben estar organizados en grupos, según su tema, para facilitar el acceso a los usuarios que quieran leerlos. Esta organización sigue un modelo jerárquico: en el nivel más alto de los grupos se encuentran los temas generales que, en un segundo nivel, están divididos en subtemas, los cuales, a su vez, pueden estar subdivididos en niveles inferiores.

Es posible que un mismo artículo pertenezca a más de un grupo. Cuando un usuario envía un artículo al sistema de noticias, además de poder especificar en qué ámbito se debe distribuir, es preciso que indique obligatoriamente a qué grupo o grupos lo quiere enviar.

La nomenclatura que se utiliza para designar los grupos consiste en concatenar los nombres de cada nivel, de más alto a más bajo, separándolos con "`.`" (por ejemplo, `news.announce` o `comp.os.linux.hardware`). En el nivel más alto de la jerarquía, los grupos originales de Usenet eran los siguientes:

- `comp`: temas relacionados con ordenadores e informática.
- `news`: artículos sobre el sistema de noticias mismo.
- `rec`: actividades recreativas, *hobbies*, etc.
- `soc`: temas sociales, culturales, humanísticos, etc.
- `sci`: temas científicos.
- `talk`: debates, discusiones, opiniones, etc.
- `misc`: otros temas no clasificables en los apartados anteriores.
- `alt`: la jerarquía alternativa, que algunos aprovechan para saltarse las normas establecidas en los grupos "oficiales".

En la actualidad,...

... existen otros grupos del nivel superior, como los que sólo contienen artículos destinados a una región geográfica. Por ejemplo `eu` para Europa, `es` para España, `fr` para Francia, etc.

En la comunidad Usenet se han establecido una serie de reglas, basadas en un sistema de votaciones, para decidir la creación de nuevos grupos o la eliminación de alguno de los ya existentes. Este carácter de democracia asamblearia, y otros aspectos como la posibilidad de publicar escritos* o dar a conocer las ideas de una persona de manera global y casi instantánea (un sueño que, hasta la llegada de Internet, sólo estaba al alcance de los grandes medios de comunicación), hicieron de Usenet el gran fenómeno sociológico de Internet durante unos cuantos años.

*** El mismo criterio se sigue para otros tipos de contenidos, como fotografías y vídeos.**

Hoy día el servicio Usenet está eclipsado por la enorme popularidad del servicio WWW, que tiene una orientación mucho más comercial, pero no es tan participativo.

6.2. Conceptos básicos del NNTP

Por norma general, el NNTP utiliza el protocolo de transporte TCP. El número de puerto asignado al servicio de noticias es el 119.

Formato de los comandos NNTP

La especificación NNTP establece que las líneas de comandos no deben tener más de 512 caracteres. Por otro lado, los comandos y los parámetros se pueden escribir indistintamente en mayúsculas o minúsculas.

En el NNTP, se utiliza un esquema de peticiones y respuestas como el del SMTP. Cada **petición** constituye una línea acabada con `<CRLF>` que contiene un comando, posiblemente con parámetros.

Las respuestas también siguen la estructura general utilizada en el FTP o el SMTP. Cada **respuesta** se representa con una línea que empieza con un código numérico de tres dígitos. A continuación, según el código de respuesta, puede haber una serie de parámetros seguidos de un texto arbitrario opcional, hasta el final de la línea, que acaba con <CRLF>.

Los significados del primer dígito del código de respuesta son similares a los del FTP y el SMTP; los del segundo son los siguientes:

- **x0x**: respuesta referente a la conexión, inicialización, etc.
- **x1x**: selección de un grupo de noticias.
- **x2x**: selección de un artículo.
- **x3x**: distribución de artículos.
- **x4x**: envío de artículos.
- **x8x**: extensiones no estándar.
- **x9x**: mensajes informativos de prueba (*debugging*).

Véase el anexo 2 de este módulo didáctico.

Algunas respuestas van seguidas de un texto formado por una secuencia de líneas. En este caso, cada línea acaba con <CRLF>, y el final de la secuencia se indica con una línea en la que sólo se encuentra el carácter "." antes de <CRLF>. Si alguna de las líneas de la respuesta debe empezar con ".", el emisor inserta otro "." al principio de la línea y, por consiguiente, el receptor debe eliminar todos los "." iniciales que encuentre antes del final de la respuesta.

Cuando el cliente establece la conexión, el servidor responde con un código 200 ó 201 para indicar que permite que el cliente envíe artículos o que no lo permite, respectivamente.

6.3. Formato de los artículos

El formato de los artículos Usenet está definido en la especificación RFC 1036, y se basa en el de los mensajes de correo electrónico*, aunque con algunas restricciones adicionales. Por tanto, cada artículo consta de una cabecera formada por una serie de campos, y de un cuerpo separado de la misma por una línea en blanco.

*** El formato de los mensajes de correo electrónico está definido en la especificación 822.**

Hay seis **campos obligatorios en la cabecera**, que son los que se exponen a continuación:

- `From`: dirección de correo electrónico del originador del artículo (y, opcionalmente, también su nombre).

- `Date`: día y hora en que se ha originado el artículo.

Lectura complementaria

Si queréis más información sobre el formato de los artículos Usenet, consultad la obra siguiente:

M.R. Horton; R. Adams (1987, diciembre). *RFC 1036 - Standard for interchange of USENET messages.*

- `Newsgroups`: lista de los grupos a los que se envía el artículo, separados por comas si hay más de uno.

- `Subject`: asunto del que trata el artículo, que se utilizará como título.

- `Message-ID`: identificador único del artículo. Aparte de este identificador único, para facilitar el acceso a los artículos, cada servidor les asigna un identificador local consistente en el nombre del grupo y un número correlativo dentro del grupo*. Dichos identificadores sólo son significativos para un servidor, puesto que otro servidor probablemente asignará identificadores locales diferentes a los mismos artículos.

*** Es decir, si un artículo se ha enviado a diferentes grupos, tendrá más de un identificador local.**

- `Path`: lista de servidores por los que ha pasado el artículo, separados con símbolos de puntuación que no sean "`.`" (por norma general, se utiliza el carácter "`!`"). Cada servidor debe añadir su nombre al principio de la lista. El valor de este campo permite que un servidor sepa si el artículo ya ha pasado por otro servidor y, por tanto, no es preciso volvérselo a enviar.

Asimismo, la cabecera de un artículo puede incluir los **campos opcionales** siguientes:

- `Reply-To`: dirección a la que deben enviarse los mensajes de correo en respuesta al artículo (por defecto, la misma del campo `From`).

- `Sender`: identificación del usuario que ha enviado el artículo al sistema de noticias, si no coincide con el del campo `From`.

- `Followup-To`: lista de grupos de noticias, separados por comas, a los que deben enviarse los artículos de respuesta (por defecto, los mismos del campo `Newsgroups`).

- `Expires`: fecha y hora en que el artículo se puede considerar caducado y, por tanto, puede ser borrado por los servidores (en ausencia de este campo, el servidor decide cuándo los borrará).

- `References`: lista de identificadores de otros artículos a los que se hace referencia en el artículo.

- `Control`: sirve para indicar que el artículo contiene un mensaje de control del sistema de noticias. El mensaje se especifica en el valor de este campo y sólo es interesante para los servidores (los usuarios no necesitan leer estos artículos). Los mensajes de control sirven para cancelar artículos, crear y borrar grupos, informar de qué artículos tiene un servidor y cuáles pide otro, etc.

- `Distribution`: lista de distribuciones a la que se debe enviar el artículo. Cada distribución es un conjunto de servidores que, por norma general, pertenecen a una misma área geográfica, una misma organización, etc.

- `Organization`: nombre de la organización (empresa, institución, etc.) a la que pertenece el originador del artículo.

- `Keywords`: lista de palabras clave relacionadas con el contenido del artículo.

- `Summary`: breve resumen del contenido del artículo.

- `Approved`: algunos grupos de noticias son moderados, lo que significa que los usuarios no pueden enviar los artículos directamente. En este caso, los artículos deben enviarse por correo electrónico a una persona, llamada **moderador del grupo**, que es la única que está autorizada para poner artículos en el grupo. De la totalidad de mensajes que recibe, el moderador decide cuáles envía al grupo y cuáles no. Los mensajes que finalmente se envían deben llevar en el campo `Approved` la dirección de correo del moderador.

- `Lines`: número de líneas del cuerpo del artículo.

- `Xref`: este campo lo genera cada servidor y no debe transmitirse de un servidor a otro. Contiene el nombre del servidor y una lista de identificadores locales que tiene el mismo artículo en otros grupos a los que se haya enviado. Una vez que el usuario ha leído el artículo, este campo permite a los lectores de noticias marcarlo como leído en todos los grupos en que aparezca.

La especificación RFC 1036 permite que la cabecera incluya otros campos no estándar. 

Ejemplo de artículo Usenet

```
From: usuari@acme.com (Ernest Udiant)
Path: News.uoc.es!news.rediris.es!news.eu.net!newsfeed.omninet.
      org!nntp.acme.com!usuari
Newsgroups: soc.culture.espanol
Subject: Nuevo libro de recetas tradicionales
Message-ID: <KvjsSAam9Ly@acme.com>
Date: Thu, 12 Dec 2002 09:12:30 GMT
Expires: Sat, 19 Dec 2002 00:00:00 GMT
Organization: ACME Inc.
Lines: 2

La semana que viene se publicará un nuevo libro de recetas
de cocina tradicional. Seguiremos informando...
```

6.4. Comandos del NNTP

A continuación, se detallan los comandos definidos en la especificación RFC 977 del NNTP:

Consultad la tabla de los códigos de respuesta al final de este subapartado.

1) Listar grupos (`LIST`)

Este comando sirve para obtener una lista de los grupos de noticias disponibles. El servidor responde con un código 215 y, a continuación, envía una secuencia de líneas (acabada con "`.`"), una para cada grupo de noticias disponible, cada una con el formato siguiente:

LIST

```
grupo último primero permiso
```

- `grupo`: nombre del grupo.
- `último`: número correlativo (identificador local) del último artículo que hay en este grupo.
- `primero`: número del primer artículo.
- `permiso`: puede ser `y` o `n` para indicar si se pueden enviar artículos a este grupo, o no, respectivamente.

2) Seleccionar un grupo (`GROUP`)

Este comando permite seleccionar un grupo concreto. El servidor envía una respuesta 211 con los parámetros siguientes (por este orden): número estimado de artículos del grupo, número del primer artículo, número del último y nombre del grupo. Si el grupo no existe, la respuesta es 411.

GROUP *grup*

El número de artículos que hay en el grupo no tiene por qué coincidir con la diferencia entre el número del último artículo y el anterior al primero, puesto que puede ser que se hayan borrado artículos intermedios.

3) Leer el artículo (`ARTICLE`)

El parámetro puede ser un identificador único de artículo o un número de artículo dentro del grupo actualmente seleccionado (el artículo leído pasa a ser considerado como el artículo actual). Sin parámetro, este comando lee el artículo actual. El servidor responde con un código 220 seguido de dos parámetros: el número del artículo y su identificador único. A continuación, envía el contenido del artículo (cabecera y cuerpo), acabado en "`.`". Si hay algún error, el servidor responde con los códigos 412, 420, 423 o 430, según el caso.

ARTICLE [< *identificador* > | *número*]

4) Leer la cabecera (`HEAD`)

Este comando es idéntico a `ARTICLE`; sin embargo, el servidor sólo devuelve la cabecera del artículo. El código de respuesta será el 221 en lugar del 220, con los mismos parámetros.

HEAD [< *identificador* > | *número*]

5) Leer el cuerpo (`BODY`)

Este comando es idéntico a `ARTICLE`; sin embargo, el servidor sólo devuelve el cuerpo del artículo. El código de respuesta será el 222 en lugar del 220, con los mismos parámetros.

```
BODY [ < identificador > |
 número]
```

6) Obtener el estatus (`STAT`)

Este comando es idéntico a `ARTICLE`; sin embargo, el servidor no devuelve ningún texto, sino sólo la línea de respuesta. El código será el 223 en lugar del 220, con los mismos parámetros.

```
STAT [ < identificador > |
 número]
```

7) Seleccionar el artículo siguiente (`NEXT`)

Este comando selecciona el artículo siguiente del grupo y hace que pase a ser considerado el artículo actual (es decir, lo que se leerá si se envía el comando llamado `ARTICLE` sin parámetro). El código de respuesta normal será el 223, como en el comando `STAT`. Un código de error específico es el 421.

```
NEXT
```

8) Seleccionar el artículo anterior (`LAST`)

Este comando selecciona el artículo anterior al actual. Los códigos de respuesta son como los del comando `NEXT`, pero cambiando el 421 por el 422.

```
LAST
```

9) Listar grupos nuevos (`NEWGROUPS`)

`NEWGROUPS` dispone de las mismas funciones que el comando `LIST`, excepto que el código de respuesta es 231 (en lugar de 215) y la lista de grupos está restringida a los que se han creado después de la fecha indicada por los parámetros (el primero son seis dígitos que representan año, mes y día, y el segundo, seis más que representan hora, minutos y segundos). Opcionalmente, se puede restringir todavía más la lista especificando una o más jerarquías de alto nivel separadas por comas (por ejemplo, `news`, `rec`, etc.).

```
NEWGROUPS aammdd hhmmss
  [GMT][< jerarquias >]
```

10) Listar artículos nuevos (`NEWNEWS`)

El servidor responde a este comando con un código 230 y, a continuación, envía una secuencia de líneas (acabada con "`.`"), una para cada artículo de los grupos especificados en el primer parámetro que se haya enviado o recibido después de la fecha indicada por el segundo y el tercero. Las líneas contienen los identificadores únicos de los artículos.

```
NEWNEWS grups aammdd hhmmss
 [GMT][< jerarquias >]
```

El primer parámetro es un nombre de grupo o una lista de nombres separados por comas. En un nombre, puede aparecer el carácter especial "`*`". En este caso, se considera que equivale a todos los nombres de grupos que, en lugar del "`*`", tengan una secuencia cualquiera de caracteres, que puede incluir el sepa-

rador "." (Por ejemplo, `alt.biz*` puede equivaler a `alt.biz.misc`, `alt.bizarre`, etc.). Por otro lado, si un nombre tiene el prefijo "!", el grupo o grupos correspondientes se excluirán de la lista (por ejemplo, `comp.lang.*, !comp.lang.c.*` equivale a los grupos que empiecen por `comp.lang.`, pero no por `comp.lang.c.`).

Opcionalmente, se puede restringir la lista de artículos a un conjunto de una o más jerarquías si se especifican en el último parámetro separadas por comas. La lista sólo contendrá artículos que pertenezcan al menos a un grupo de las jerarquías.

11) Enviar un artículo (`POST`)

Este comando se utiliza para enviar un artículo. El servidor responde con un código 440 para indicar que no se permite la operación, o con un 340 para solicitar el artículo. En el segundo caso, el cliente debe enviar el contenido del artículo (cabecera y cuerpo) con el formato definido en la especificación RFC 1036 y acabado con una línea en que sólo haya un ".". Entonces, el servidor enviará la respuesta definitiva, que será 240 o, si ha habido algún error, 441.

```
POST
```

12) Ofrecer un artículo (**`IHAVE`**)

Cuando un servidor se conecta a otro, puede utilizar el comando `IHAVE` para hacerle saber de qué artículos dispone.

```
IHAVE < identificador >
```

En cada artículo, el servidor receptor enviará en la respuesta un código 335 si quiere recibir una copia del mismo o 435 si no le interesa. Si es preciso pasar el artículo, se hace por medio del comando `POST`, y el servidor receptor enviará en la respuesta un código 235 o, en caso de error, uno 436. Asimismo, puede suceder que al receptor no le interese el artículo después de examinar su contenido; en este caso, la respuesta será 437.

Doble ofrecimiento

Un servidor no debería ofrecer a otro servidor artículos que ya le haya enviado antes o que ya hayan pasado por este último (el campo `Path` de la cabecera lo indica).

13) Conexión desde un servidor esclavo (`SLAVE`)

Este comando sirve para informar al servidor de que la conexión proviene de un servidor esclavo. El servidor puede decidir, por ejemplo, dar más prioridad a esta conexión que a las que provienen directamente de clientes porque se supone que atiende a más usuarios. El código de respuesta es el 202.

```
SLAVE
```

14) Mensaje de ayuda (`HELP`)

Este comando se utiliza para obtener un mensaje de ayuda. El servidor responde con un código 100 y, a continuación, un texto informativo de los comandos que acepta, acabado con una línea con un carácter ".".

```
HELP
```

15) Cerrar la conexión (`QUIT`)

Este comando se utiliza para cerrar la conexión. El servidor responde enviando un código 205 y cerrando la conexión.

```
QUIT
```

La especificación RFC 977 permite que las implementaciones añadan otros comandos a los estándares; sin embargo, recomienda que el nombre de estos nuevos comandos empiece con X para evitar posibles conflictos con posteriores extensiones oficiales.

Algunos ejemplos de comandos implementados por muchos servidores son los siguientes:

- XGTITLE: confiere un título descriptivo de uno o más grupos.

- XHDR y XOVER: dan, respectivamente, el valor de un campo y un resumen de la cabecera de uno o más artículos.

- XPAT: proporciona los valores de un campo de la cabecera que concuerdan con un patrón, junto con los números de los artículos correspondientes.

La tabla siguiente resume los códigos de respuesta del protocolo NNTP:

Códigos de respuesta	
Código	**Significado**
100	Mensaje de ayuda.
200	Servidor preparado; se permite enviar artículos.
201	Servidor preparado; no se permite enviar artículos.
202	Conexión desde servidor esclavo.
205	El servidor cierra la conexión.
211	Grupo seleccionado.
215	Lista de grupos.
220	Contenido del artículo.
221	Cabecera del artículo.
222	Cuerpo del artículo.
223	Identificador del artículo.
230	Lista de artículos nuevos.
231	Lista de grupos nuevos.
235	Artículo recibido.
240	Artículo enviado.
335	Preparado para recibir un artículo de otro servidor.
340	Preparado para recibir un artículo del cliente.
400	Servicio no disponible.
411	No existe el grupo.
412	No hay ningún grupo seleccionado.
420	No hay ningún artículo seleccionado.
421	No hay artículo siguiente.

Códigos de respuesta	
Código	**Significado**
422	No hay artículo anterior.
423	No hay ningún artículo con este número.
430	No hay ningún artículo con este identificador.
435	No se desea recibir el artículo.
436	Error en la recepción del artículo.
437	Artículo rechazado.
440	No se permite enviar artículo.
441	No se ha podido enviar el artículo.
500	Comando desconocido.
501	Error de sintaxis en el comando.
502	Permiso denegado.
503	Error interno.

7. Servicio hipermedia: WWW

7.1. Documentos hipermedia

El **servicio WWW*** ofrece acceso a información multimedia, que puede incluir contenidos de diferentes tipos (texto, imágenes, audio, vídeo, etc.) y referencias a otros elementos de información, según el modelo de los sistemas hipertexto.

*** WWW es la sigla de *world wide web*.**

Un **sistema hipertexto** permite recorrer un documento de manera no necesariamente lineal o secuencial, sino siguiendo las referencias o enlaces que el usuario selecciona y saltando a la parte referenciada. Es decir, en lugar de seguir un único camino lineal, se puede realizar un recorrido por el documento pasando por los arcos de un grafo. Esta manera de acceder a la información se suele llamar familiarmente *navegar*.

Cuando se generaliza la funcionalidad hipertextual para que permita navegar por elementos de información multimedia y no sólo por texto, se suele hablar de **sistemas hipermedia**.

En la terminología WWW, cada uno de los elementos de información a los que se puede acceder se llama *recurso*. Los **documentos hipermedia** constituyen un caso particular de recurso que contiene información estructurada, posiblemente con diferentes tipos de contenido y enlaces a otros recursos.

En la estructura de un documento, en general, se pueden distinguir los elementos siguientes:

1) La **estructura lógica**, es decir, la división del contenido en partes que se encuentran en diferentes niveles, como capítulos, secciones, apartados, subapartados, títulos, párrafos, figuras, tablas, encabezamientos, notas, etc.

2) La **estructura física**, es decir, la disposición de las partes del documento en el medio de presentación (por norma general, papel o pantalla): la división del documento en páginas y grupos de páginas, la distribución del área de cada página en zonas para cabeceras y pies, los márgenes, los espacios reservados a figuras, tablas, etc.

En el modelo general de procesado de los documentos estructurados, el autor crea la estructura lógica, con el contenido correspondiente, y puede incluir ciertas directrices para controlar el aspecto visual que deberá tener el documento. En numerosas ocasiones, a un determinado conjunto de directrices de

este tipo se le da el nombre de *estilo*. A partir de esta información, un proceso denominado **de formateo** se encarga de generar la estructura física.

Existen diferentes **representaciones normalizadas de los documentos estructurados**, las cuales van destinadas a objetivos diferentes:

a) Representaciones que se centran en la especificación de la estructura lógica, como el **SGML** (*Standard Generalized Markup Language*, publicado en el estándar internacional ISO 8879).

b) Representaciones que se centran en la especificación de la estructura física, como el **SPDL** (*Standard Page Description Language*, publicado en el estándar ISO/IEC 10180).

El SPDL...

... está basado en otro lenguaje muy popular que sirve para representar documentos formateados: el **lenguaje PostScript**.

c) Representaciones que comprenden tanto la estructura lógica como la física, por ejemplo el **estándar ODA** (*Open Document Architecture*, publicado en el estándar ISO/IEC 8613 y en la serie de recomendaciones ITU-T T.410).

7.2. Marcado: el SGML

El SGML contiene la especificación de un lenguaje que sólo permite representar la estructura de un documento desde el punto de vista lógico. En principio, un usuario puede aplicar los estilos que quiera para visualizar el documento. Sin embargo, lo más habitual es que las aplicaciones que utilizan este lenguaje proporcionen una serie de reglas para interpretar la estructura del documento y, en particular, para formatearlo. Por otro lado, se ha publicado otro estándar, llamado **DSSSL***, que permite definir estilos de presentación con las transformaciones necesarias para convertir un documento SGML en otra notación, como el SPDL.

*** DSSSL es la sigla de d*ocument style semantics and specification language*, publicado en el estándar ISO/IEC 10179.**

Un ejemplo de aplicación basada en SGML es el **HTML** (*Hypertext Markup Language*) diseñado para especificar documentos hipermedia.

Actualmente, la especificación oficial del HTML está bajo el control de un grupo de empresas y organizaciones conocido como World Wide Web Consortium (W3C). Con anterioridad, la responsabilidad era de un grupo de trabajo de la Internet Engineering Task Force (IETF), que cesó sus actividades en 1996 tras publicar el documento RFC 1866 con la especificación HTML 2.0.

Podéis consultar las características del HTML en el anexo 3 de este módulo didáctico.

El SGML se publicó en 1986 con el objetivo de facilitar la especificación de documentos estructurados. El lenguaje definido en este estándar, sin embargo, es lo suficientemente general para permitir la representación de muchos otros tipos de información estructurada (bases de datos, etc.).

Lectura complementaria

Podéis consultar la obra siguiente para estudiar con detenimiento en el SGML:

Charles Goldfarb (1990). *The SGML Handbook*. Oxford: Oxford University Press.

Según el modelo SGML, un documento consta de un conjunto de elementos contenidos unos dentro de otros formando un árbol. Estos elementos consti-

tuyen los componentes de la estructura lógica del documento. Por ejemplo, los elementos de nivel más alto pueden ser capítulos, que pueden contener apartados, los cuales, a su vez, pueden contener subapartados, etc.

La característica básica del SGML es el **uso de marcado generalizado** para delimitar los elementos que componen el documento. En lugar de asociar instrucciones de procesado a los elementos (como "dejar dos líneas de espacio vertical y cambiar a letra negrita con cuerpo de 12 puntos"), en el SGML se les asigna una etiqueta descriptiva (por ejemplo "título de subapartado" o "nota a pie de página"). Es decir, en el SGML el marcado sirve para especificar qué representan los elementos de la estructura y no qué apariencia deberán tener en el documento formateado.

La asignación de los formatos correspondientes a cada elemento constituye un proceso completamente independiente de la especificación SGML (que se puede llevar a cabo, por ejemplo, por medio del DSSSL).

7.2.1. Representación de un documento SGML

Los documentos SGML están contenidos en unidades de información, las **entidades** (por norma general, cada entidad se almacena en un fichero). La información consiste en una combinación de **datos** (por ejemplo, el texto de los párrafos o el contenido de las figuras) y **marcado**.

En el SGML se consideran tres tipos de marcado:

- Etiquetas o marcado descriptivo.
- Referencias a entidades.
- Declaraciones.

El estándar SGML...

... define un cuarto tipo de marcado, que sirve para incluir instrucciones de procesado en el texto, pero desaconseja su uso porque contradice la filosofía del marcado generalizado.

El estándar define una sintaxis abstracta que especifica cómo deben utilizarse los delimitadores para incluir marcado en el documento. La finalidad de los delimitadores es que los analizadores SGML puedan distinguir el marcado de los datos. La representación de los delimitadores en un documento está determinada por la sintaxis concreta asociada al documento, que por lo general es la denominada *sintaxis concreta de referencia*, definida en el mismo estándar.

Delimitadores

Por norma general, un delimitador sólo se reconoce como tal en el contexto en que puede aparecer. Por ejemplo, tras abrir una etiqueta o una declaración, el carácter ">"indica su final; sin embargo, en cualquier otro punto del documento es un carácter normal, sin ningún significado especial.

Otro ejemplo son los delimitadores "<" y "&", que actúan como tales sólo cuando van seguidos de un posible carácter inicial de nombre SGML (es decir, una letra).

La tabla siguiente muestra las secuencias de caracteres que la sintaxis concreta de referencia asigna a cada delimitador de marcado:

	Delimitador para abrir	Delimitador para cerrar
Etiqueta de inicio	<	>
Etiqueta de final	</	>
Referencia a entidad	&	;
Declaración	<!	>

Por tanto, de ahora en adelante, cuando hablamos de representaciones concretas de delimitadores, nos referiremos a las representaciones definidas en la sintaxis concreta de referencia. 

Etiquetas

Existen dos tipos de etiquetas SGML: las de inicio y las de final. Dichas etiquetas sirven para indicar dónde empiezan y dónde acaban, respectivamente, los elementos del documento. Cada etiqueta contiene el identificador genérico del elemento correspondiente; es decir, un nombre que indica de qué tipo de elemento se trata. Cada aparición de un elemento en el documento se denomina una *instancia del tipo de elemento correspondiente*.

Un identificador genérico se representa por medio de un nombre SGML. En la sintaxis concreta de referencia, un nombre SGML puede constar de hasta ocho caracteres, el primero de los cuales debe ser una letra y los otros pueden ser letras, dígitos o los caracteres "." o "-". En la mayoría de los contextos, como en el caso de los identificadores genéricos, las letras mayúsculas y minúsculas de un nombre SGML se consideran equivalentes.

La información comprendida entre una etiqueta de inicio y la correspondiente etiqueta de final es el contenido del elemento. En el contenido del elemento puede haber datos y/u otros elementos.

Capítulo de texto SGML

Este sencillo ejemplo de texto SGML podría servir para representar un capítulo (elemento `CAP`) formado por un título (elemento `TITULO`) y una secuencia de párrafos (elementos `PAR`), cada uno de los cuales contiene un texto que puede incluir términos que conviene destacar (elementos `TERMINO`) y notas a pie de página (elementos `NOTA`) intercaladas:

```
<CAP><TITOL>Etiquetas</TITOL>
<PAR>Hay dos tipos de etiquetas SGML: las de inicio y las de
final.</PAR>
<PAR>Cada etiqueta contiene el<TERMINO>identificador genèrico
</TERMINO>
<NOTA>El identificador genérico es un nombre que indica de qué
tipo de elemento se trata.</NOTA> del elemento correspondiente.
</PAR>
</CAP>
```

Además del identificador genérico, en la etiqueta de inicio también puede haber una lista de atributos, cada uno formado por un nombre y un valor, que sirven para especificar propiedades del elemento. El nombre del atributo es un nombre SGML, y el valor puede ser una secuencia arbitraria de caracteres, delimitada en el inicio y en el final con los símbolos "`"`" o "`'`", y separada del nombre con el símbolo "`=`". Si el valor es un *token*, los delimitadores "`"`" o "`'`" se pueden omitir. Se entiende por *token* una secuencia de caracteres que pueden formar parte de un nombre SGML (letras, dígitos, "`.`" y "`-`").

Notación

Si el valor del atributo de una etiqueta debe contener el carácter `', entonces el delimitador deberá ser ", y viceversa.

Ejemplo de etiqueta de inicio con atributos

```
<REFER NOMBRE="Etiquetas de inicio" TIPO=ref2>
```

Referencias a entidades

Cuando no sea conveniente incluir directamente una porción de los datos en el documento, esta última puede ser sustituida por una **referencia a entidad**. Cuando se procese el documento, cada referencia se considerará equivalente al valor de la entidad referenciada. El valor de las entidades se define por medio de las declaraciones SGML correspondientes.

En ocasiones...

... no conviene incluir ciertos datos en el documento porque se pueden confundir con el marcado, porque la longitud es muy grande, porque se repiten muchas veces, etc.

Una referencia a entidad contiene simplemente el nombre de la entidad, que debe ser un nombre SGML (en la sintaxis concreta de referencia, no se consideran equivalentes las letras mayúsculas y minúsculas en los nombres de entidad). El delimitador que cierra la referencia (;) puede omitirse si el carácter siguiente no se puede confundir con un carácter del nombre.

Ejemplo

```
<PAR>La figura &num-cap;.3 muestra la solución propuesta en el llibro
&manual-2 para el caso general.</PAR>
<FIGURA>&fig3</FIGURA>
```

Un caso especial de referencias a entidades es el de las referencias a carácter, que se abren con el delimitador `&#` y que, en lugar de un nombre de entidad, contienen un número decimal. Cada referencia de este tipo equivale al carácter que tiene el código indicado por el número, de acuerdo con el juego de caracteres utilizado en el documento; por ejemplo,... si se utiliza el código ASCII, la referencia `&#60;` equivale al carácter "`<`".

Tipos de declaraciones SGML

Existen diferentes tipos de declaraciones SGML; sin embargo, las más utilizadas son las declaraciones de elementos, las de atributos, las de entidades y las vacías. Las tres primeras se distinguen por la palabra que aparece justo después

del delimitador que las abre (<!): ELEMENT, ATTLIST y ENTITY, respectivamente. En las declaraciones vacías, justo después del delimitador inicial se encuentra el delimitador final.

1) Declaraciones de elementos. Permiten declarar un identificador genérico y definir el modelo de contenido de los elementos que tengan este identificador. Un **modelo de contenido** es una expresión que especifica cuáles son las combinaciones válidas de elementos (indicados por sus identificadores genéricos) y/o datos que puede haber en el contenido del elemento.

Se puede expresar por medio de un grupo o como contenido declarado:

a) Un **grupo** es una construcción que se abre con el delimitador "(" y se cierra con el delimitador ")". Contiene una serie de subgrupos combinados con los operadores siguientes:

- ",": indica que deben estar en secuencia.
- "&": indica que pueden aparecer en cualquier orden.
- "|": significa que sólo puede haber uno.

Cada subgrupo consta de la palabra #PCDATA (que significa que puede aparecer una secuencia de cero o más caracteres), un identificador genérico o bien otro grupo. En estos dos últimos casos, a continuación del subgrupo puede haber uno de los indicadores siguientes:

- "?": el subgrupo es opcional (puede aparecer una vez o ninguna).
- "+": el subgrupo es obligatorio y repetible (puede aparecer una vez o más).
- "*": el subgrupo es opcional y repetible (puede aparecer cero veces o más).

El símbolo #...

... se utiliza como prefijo de las palabras clave del lenguaje en contextos en los que se podrían confundir con nombres. En un modelo de contenido, pues, #PCDATA representa una secuencia de caracteres (*parsed character data*), mientras que PCDATA representaría un identificador genérico como cualquier otro.

b) Un **contenido declarado** es una palabra clave, que puede ser CDATA si el contenido es una secuencia de caracteres en la que no se reconoce ningún marcado más que el que indica el final del elemento, ANY si es una combinación arbitraria de caracteres y/o cualquier otro elemento, o EMPTY si el elemento no tiene contenido. Un elemento con contenido CDATA, pues, sólo puede contener caracteres. Las instancias de los elementos sin contenido se representan sólo con una etiqueta de inicio; es decir, no debe especificarse la etiqueta de final.

Las palabras clave...

... del contenido declarado no se pueden confundir con un identificador genérico (si el modelo de contenido no empieza con "(", sólo puede ser contenido declarado) y, por tanto, no necesitan el delimitador "#".

Ejemplo de declaración de elementos

En los ejemplos siguientes de declaraciones de elementos, se muestra la posibilidad de utilizar un subgrupo en lugar de un identificador genérico cuando hay diferentes elementos con el mismo modelo de contenido:

```
<!ELEMENT carta   (encab, fecha?, cuerpo, firma, postdata*)>
<!ELEMENT encab                      (remit, destin)>
<!ELEMENT (remit|destin)             (nom, dir?)>
<!ELEMENT (cuerpo|postdat)           (parr+)>
<!ELEMENT (parr|nom|dir|data|firma)   CDATA>
```

2) Declaraciones de atributos. Cada declaración consta del identificador genérico de los elementos a los que se aplica, seguido de una serie de tripletas (una para cada atributo) formadas por el nombre del atributo, sus posibles valores y el valor por defecto. No puede haber más de una declaración de atributos asociada al mismo identificador genérico. 

Existen dos maneras de especificar los valores posibles de un atributo. Si el valor puede ser un *token* que es preciso elegir de entre un conjunto finito, se expresa en forma de subgrupo. Cada *token* debe ser diferente de los demás, no sólo de los del mismo conjunto, sino también de cualquier otro que se encuentre en la misma declaración. La otra opción es usar una palabra clave determinada: `CDATA` si el valor es una secuencia arbitraria de caracteres, `NAME` si es un nombre SGML, `NUMBER` si es una secuencia de dígitos, `TOKEN` si es un *token*, y `NAMES`, `NUMBERS` o `TOKENS` si es una secuencia de nombres SGML, números o *token*s, respectivamente, etc.

El valor por defecto es el que se aplica si se omite el atributo en la etiqueta de inicio de un elemento. Puede ser uno de los valores posibles del atributo o bien una palabra clave: `#REQUIRED` si el atributo no se puede omitir nunca, `#CURRENT` si el valor por defecto es igual al último especificado para el mismo atributo en los elementos anteriores, `#IMPLIED` si el valor puede ser determinado por la aplicación que procese el documento, etc.

Ejemplo de declaración de atributo

En este ejemplo, se declaran los atributos que pueden especificarse en un elemento de tipo `informe`:

```
<!ATTLIST informe autores  NAMES            #REQUIRED
                 titulo    CDATA            #REQUIRED
                 version   NUMBER           1
                 estado    (prelim|final)   final
                 fecha     CDATA            #IMPLIED >
```

3) Declaraciones de entidades. Constan del nombre de la entidad seguido de su valor. Este último se puede expresar como un literal, o bien como una referencia a una entidad externa:

Nombres iguales

Si en un documento hay más de una entidad declarada con el mismo nombre, sólo se considera la primera y las restantes se ignoran.

a) Un **literal** es una secuencia arbitraria de caracteres delimitada con los símbolos “"” o “'” (en este caso, los delimitadores no se pueden omitir).

b) Si el valor está almacenado en una entidad externa (por norma general, un fichero), se utiliza un **identificador externo** para referenciarlo. El uso de entidades externas es conveniente cuando se referencia el valor desde diferentes documentos, o cuando el contenido de la entidad no son datos SGML (por ejemplo, una imagen fotográfica), puesto que estos últimos podrían confundir a los analizadores de marcado.

Los identificadores externos están formados por la palabra `PUBLIC` seguida de un identificador público, o bien por la palabra `SYSTEM`. Opcionalmente, se puede especificar a continuación un identificador de sistema. El identificador público permite referenciar una entidad que se conozca más allá del contexto del documento que se procesa. La palabra `SYSTEM`, en cambio, indica que el sistema local sólo debe saber cómo se accede al mismo a partir de su nombre. Tanto en un caso como en el otro, el identificador de sistema (opcional) puede proporcionar información adicional para encontrar la entidad; por ejemplo, el nombre del directorio en que se encuentra el fichero que la contiene.

Los identificadores públicos y los de sistema se representan como literales. En el caso de los primeros, su valor puede seguir la estructura de los llamados ***identificadores públicos formales***.

Ejemplo de declaración de entidad literal

Éste es un ejemplo típico de declaración de entidad literal que permite incluir el carácter "<" en el texto de un documento:

```
<!ENTITY lt "<">
```

Una vez declarada esta entidad, se puede utilizar la referencia `<` para representar el símbolo "<" sin que se interprete como un delimitador.

Ejemplo de declaración de entidad externa

Este ejemplo es una declaración de entidad externa con identificador público formal:

```
<!ENTITY % ll-greg PUBLIC "ISO 8879-1986//ENTITIES Greek Letters//EN">
```

- La primera parte del identificador (`ISO 8879-1986`) indica el propietario o autor del texto público. Debe empezar por `ISO` si el texto está definido en un estándar internacional; por `+//` si el propietario está registrado oficialmente (de acuerdo con la norma ISO/IEC 9070) o por `-//` en cualquier otro caso.
- La parte siguiente, separada de la anterior con `//`, es una palabra clave que indica de qué tipo de texto público se trata: `DTD` si es una declaración de tipo de documento, `ENTITIES` si es un conjunto de declaraciones de entidades, `TEXT` si es una entidad que contiene texto SGML, etc.
- A continuación, hay una descripción que sirve para identificar el texto público (en este ejemplo, `Greek Letters`).
- Por último, separado con `//`, un código de dos letras según la norma ISO 639 indica el idioma utilizado en el texto público, por si hay diferentes versiones traducidas (`EN` corresponde al idioma inglés).

El símbolo "%"...

... antes del nombre de la entidad indica que esta última no se referenciará desde el texto del documento, sino desde una declaración. En este caso, el delimitador que abre la referencia no es "&", sino "%": `% ll-greg`

Las declaraciones de elementos, entidades y atributos sólo pueden aparecer en el prólogo del documento, como veremos a continuación. Las declaraciones vacías, en cambio, pueden figurar en cualquier parte del documento.

Dentro de un comentario...

... puede haber cualquier carácter, por ejemplo, ">". La única restricción es que la secuencia "--" se interpreta como el delimitador de final de comentario.

Dentro de una declaración puede haber **comentarios**, que son secuencias de caracteres cuyo inicio y final viene marcado por el delimitador "--". La utilidad principal de las declaraciones vacías es incluir comentarios en el documento.

Ejemplos de declaraciones con comentarios

```
<!-- Declaración vacía -->
<!-- Declaración vacía -- -- con dos comentarios -->
<!ATTLIST parr -- atributo tipo  defecto --
                indent  NUMBER  0
                letra   NAME    #CURRENT
              -- por defecto el actual -->
```

7.2.2. Estructura de un documento SGML

Un documento SGML está formado por los tres componentes siguientes (en este orden):

1) La **declaración SGML** (opcional), que sirve para especificar las convenciones sintácticas que se utilizan en el resto del documento (por ejemplo, la sintaxis concreta). No es preciso incluir explícitamente la declaración SGML en el principio de los documentos si van ser procesados por un mismo sistema o aplicación. Si no hay declaración SGML, el sistema puede aplicar por defecto una predefinida.

2) El **prólogo** o declaración de tipo de documento, que consta de la palabra `DOCTYPE`, seguida del identificador genérico del elemento raíz* del documento, y de un conjunto de declaraciones delimitado por los símbolos "[" y "]" al inicio y al final, respectivamente. Opcionalmente, la declaración de tipo de documento puede referenciar un conjunto externo de declaraciones por medio de un identificador externo.

*** El elemento raíz también se denomina *elemento documento*. Su identificador genérico actúa como nombre del tipo de documento.**

3) El **contenido** o instancia del documento, que es simplemente una instancia del elemento raíz, es decir, de aquél que tiene por contenido el de todo el documento.

Ejemplo de documento SGML completo

Sin tener en cuenta la declaración SGML, éste podría ser un (sencillísimo) documento SGML completo:

```
<!DOCTYPE informe [
 <!ELEMENT informe (titulo, autor?, texto)>
 <!ATTLIST informe version NUMBER 1>
 <!ELEMENT texto   (parr+)>
 <!ELEMENT (titulo|autor|parr) CDATA>
]>
<informe version=2><titulo>Ejemplo</titulo><autor>Anónimo
</autor> <texto><parr>Primer párrafo.</parr><parr>Párrafo
final.</parr> </texto></informe>
```

Si la declaración del tipo de documento (DTD)* estuviera contenida en una entidad externa, el prólogo se podría escribir, por ejemplo, de este modo:

*** La DTD es el conjunto de reglas, incluyendo una declaración de tipo de documento, que especifican la manera como una determinada aplicación utiliza el marcado SGML para representar un tipo particular de documento.**

```
<!DOCTYPE informe PUBLIC "-//ACME S.A.//DTD Informe confidencial//CA">
```

Minimizaciones

Como el marcado de un documento SGML puede contener mucha redundancia, el estándar define una serie de **opciones de minimización** que permiten reducir la información a especificar. Su uso en un documento concreto debe indicarse en su declaración SGML. A continuación, describiremos tres de estas opciones:

a) La opción llamada **OMITTAG** permite omitir etiquetas redundantes. Cuando se utiliza, cada declaración de elemento debe incluir, después del nombre del elemento, dos símbolos que indican si es válido omitir las etiquetas de inicio y de final: "o" significa que se permite la omisión y "-", que no se permite.

La declaración de un elemento...

... no debería permitir la omisión de la etiqueta de inicio si este último tiene asociados atributos #REQUIRED o si su modelo de contenido es EMPTY.
En este último caso, la omisión de la etiqueta de final es obligatoria.

Aunque la declaración lo permita, sólo es posible omitir una etiqueta si su ausencia no crea ambigüedades o errores en la interpretación del marcado. Tampoco puede omitirse una etiqueta de inicio si contiene atributos con valores diferentes de los valores por defecto, o si el contenido del elemento está vacío.

Ejemplo de documento SGML utilizando la opción OMITTAG

Utilizando la opción OMITTAG, el documento del ejemplo anterior se podría escribir del modo siguiente:

```
<!DOCTYPE informe [
<!ELEMENT informe                   o o (titulo, autor?, texto)>
<!ATTLIST informe                        version NUMBER 1>
<!ELEMENT texto                     o o (parr+)>
<!ELEMENT (titulo|autor|parr)        - o (#PCDATA)>
]>
<informe version=2><titulo>Ejemplo<autor>Anónimo
<parr>Primer párrafo.<parr>Párrafo final.
```

Las etiquetas de inicio y de final del elemento informe son redundantes porque coinciden con el inicio y el final del documento (aquí no se ha podido omitir, sin embargo, la de inicio a causa del atributo version). Las etiquetas de final de los elementos titulo, autor y parr también lo son porque, cuando empieza el elemento siguiente, significa que ha acabado el actual. Y las etiquetas del elemento texto también son redundantes: teniendo en cuenta su modelo de contenido, cuando empieza el primer parr significa que empieza el texto, y cuando se acaba el documento también se acaba el texto.

#PCDATA

El modelo de contenido de los elementos terminales se ha cambiado a #PCDATA para poder reconocer las etiquetas de inicio de los elementos siguientes, puesto que, en un elemento con contenido CDATA, el único marcado que se reconoce es su etiqueta de final.

b) La opción denominada **SHORTTAG** permite, entre otras cosas, abreviar la lista de atributos en una etiqueta de inicio. Existen dos métodos de abreviación posibles: si el valor de un atributo coincide con el valor por defecto, puede omitirse la especificación de dicho atributo (cuando no se utiliza, sólo se pueden omitir los atributos #IMPLIED) y, si los valores posibles de un atributo son un conjunto finito de *tokens*, se puede omitir el nombre del atributo y el símbolo "=" y dejar únicamente el valor (sin delimitadores).

Ejemplo de minimización de etiquetas de inicio con la opción SHORTTAG

Con el primer ejemplo de declaración de atributos que ya hemos visto, y utilizando la opción de minimización `SHORTTAG`, estas etiquetas de inicio serían equivalentes:

```
<informe autores="anon" titulo="Ejemplo" version=1 estado=prelim>
<informe autores="anon" titulo="Ejemplo" prelim>
```

c) La opción denominada **`SHORTREF`** permite que ciertos delimitadores actúen como referencias a entidad. El conjunto de delimitadores que pueden actuar de este modo incluye los saltos de línea, las líneas vacías, los espacios en blanco y las secuencias de un espacio o más. Para utilizar esta opción, es preciso especificar una lista con los delimitadores que se utilizarán y el nombre de la entidad asociado a cada uno. Aparte, es necesario declarar el valor de cada entidad. Un uso típico de la opción `SHORTREF` es facilitar la composición del documento sin necesidad de poner explícitamente todo el marcado.

Se puede definir una entidad que equivalga a una etiqueta de inicio de párrafo y asociarla al delimitador línea vacía. De este modo, cada línea en blanco que haya en el documento se interpretará como inicio de un nuevo párrafo. Otro ejemplo muy común es hacer equivaler una secuencia de espacios a un solo espacio.

7.3. Transferencia de hipermedia: el HTTP

El HTTP* es la base del servicio WWW. Sus orígenes fueron los trabajos de un grupo de investigadores del CERN (Centro Europeo de Investigación Nuclear) que, al final de los años ochenta y principios de los noventa, definieron un protocolo para acceder de manera sencilla y cómoda a la información distribuida en las diferentes sedes del centro. Cuando el conjunto de sistemas accesibles con este mecanismo se extendió a otras organizaciones y países de todo el mundo, nació lo que hoy conocemos como WWW (*World Wide Web*).

*** HTTP es la sigla de *hypertext transfer protocol.***

De la misma manera que el HTML, el HTTP también evoluciona constantemente a medida que los implementadores incorporan nuevas funcionalidades para adaptarlo a diferentes tipos de aplicaciones particulares. En 1996, se publicó el documento RFC 1945, en que se define la versión HTTP/1.0 (este documento también contiene las especificaciones de la denominada *versión 0.9* para facilitar la interoperabilidad con implementaciones antiguas). Sin embargo, el año siguiente ya se publicó la especificación HTTP/1.1 en el documento *RFC 2068*.

A continuación, veremos las características principales de la funcionalidad que ofrece el HTTP. Sin embargo, antes de entrar en los detalles del protocolo, estudiaremos el método general utilizado en el servicio WWW para identificar la información a la que se quiere acceder.

7.3.1. Direccionamiento: identificadores uniformes de recurso (URI)

Desde un documento se pueden referenciar recursos especificando sus direcciones, que se representan por medio de lo que en la terminología WWW se denomina *identificador uniforme de recurso* o *URI*.

Definición de URI

La definición de URI constituye otro ejemplo de especificación en constante evolución, al menos hasta que se publicó el documento *RFC 2396*, en 1998.

La forma general de un URI se puede expresar del modo siguiente:

```
esquema: identificador
```

Ésta es la forma correspondiente a un **URI absoluto**. La palabra que se encuentra antes del separador "`:`" es el esquema, que determina la sintaxis del identificador. Esta sintaxis también puede permitir la especificación de **URI relativos**, en que se omite el esquema y el separador "`:`".

Un URI puede ser un **localizador** (**URL**), si especifica cómo se accede al recurso, y/o un **nombre** (**URN**) si identifica el recurso por medio de un conjunto de atributos:

```
ftp://[usuario[:contraseña]@]servidor/camino
```

Según el esquema, éstas son algunas sintaxis posibles de los URL:

a) `ftp`

Este tipo de URL identifica un recurso accesible por medio del protocolo FTP: `servidor` es el nombre del ordenador servidor y `camino` es el nombre completo de un fichero o directorio. Si el nombre de usuario no se encuentra presente, significa que es preciso utilizar `anonymous` y, como contraseña, cualquier cadena de caracteres.

b) `news`

Este URL identifica un conjunto de artículos si `identificador` es el nombre de un grupo de noticias Usenet, en el caso de que sólo sea un identificador de mensaje (campo `Message-ID`), entonces sólo identificará un determinado artículo:

```
news: identificador
```

c) `mailto`

Este identificador URL representa una dirección de correo electrónico:

```
mailto: dirección
```

d) `telnet`

Este URL representa un servicio accesible por medio del protocolo Telnet en el servidor y el puerto especificados:

```
telnet://[usuario[:contraseña]@]servidor[:puerto][/]
```

e) `http`

Cuando deba accederse al recurso identificado por medio del HTTP, se utilizará un URL de acuerdo con la sintaxis siguiente:

```
URL-HTTP = URL-HTTP-absoluto | URL-HTTP-relativo
URL-HTTP-absoluto = http://servidor[:puerto]
[camino-absoluto]
URL-HTTP-relativo = camino-absoluto | camino-relativo
camino-absoluto = /camino-relativo
camino-relativo = [camino] *(;parámetro)[? consulta]
[# fragmento]
camino = segmento *(/segmento)
```

En esta sintaxis, `servidor` es el nombre del ordenador con que es preciso establecer la conexión HTTP para acceder al recurso, y `puerto` es el número de puerto en que debe efectuarse la conexión que, por defecto, es el número asignado al servicio WWW, es decir, el 80.

Cada uno de los segmentos que forman el `camino` se asocia, por norma general, a un nombre de directorio o de fichero. Estos nombres no pueden incluir los caracteres "`/`", "`;`", "`?`", "`\`", "`#`", "`"`", "`<`", "`>`" o espacio. El carácter "`/`" sí que puede estar presente en un parámetro, y tanto "`/`" como "`;`" y "`?`" también pueden estar presentes en los componentes `consulta` y `fragmento`.

Nota

Como caso particular, `segmento`, `parámetro`, `consulta` o `fragmento` pueden ser una cadena vacía.

Si en el `camino` aparece el carácter "`%`", debe ir seguido de dos dígitos hexadecimales, y esta secuencia de tres caracteres equivale al carácter que tenga el código indicado por los dígitos. Por tanto, se puede utilizar una secuencia "`%HH`"

para representar cualquiera de los caracteres no permitidos en un URL, incluyendo el mismo carácter "%" (que se representaría con "%25").

Esquemas de URL

A continuación, presentamos diferentes ejemplos de URL con esquemas diferentes (lo dos últimos son URL HTTP relativos):

```
ftp://ftp.uoc.es/pub/doc/README
news:comp.infosystems.www.misc
mailto:Ernest.Udiant@campus.uoc.edu
http://www.uoc.es/
http://www.acme.com/%7Eadmin/
http://www.acme.com/buscador/busca.cgi?nom=Internet
http://www.acme.com/doc/ayuda.html#buscador
/doc/ayuda.html#buscador
ayuda.html#buscador
```

Los URL relativos se pueden utilizar en documentos HTML, en concreto en los atributos que representan direcciones de otros recursos (`HREF`, `SRC`, `ACTION`). Estos URL se consideran relativos a una determinada dirección base, que es la indicada por el elemento `BASE` del documento o, si este elemento no se encuentra presente, la dirección que se ha utilizado para acceder al mismo documento HTML.

Para obtener el correspondiente URL absoluto, es preciso aplicar las reglas siguientes:

1) Si el URL relativo contiene un camino absoluto, sustituye al camino de la dirección base.

2) Si el URL relativo contiene un camino relativo (y el camino no está vacío), se sustituye el último segmento de la dirección base (la parte del camino que se encuentre después del último "`/`") por éste. Como caso especial, cada segmento con nombre "`..`" que se encuentre en el inicio del camino relativo debe suprimirse junto con el último segmento que quede al final del camino de la dirección base.

3) Si el camino del URL relativo está vacío, pero hay alguno de los otros componentes (parámetros, consulta, fragmento), se toma el camino de la dirección base y se sustituyen los componentes correspondientes por los del mismo.

URL relativo vacío

Si el URL relativo está completamente vacío, se toma directamente toda la dirección base, incluyendo el fragmento, si hay (en los otros casos, el fragmento de la dirección base no se considera).

Interpretación del URL relativo

Dada la dirección base `http://www.uoc.edu/extern/ct/home/home.html`, los URL relativos siguientes deberían interpretarse de este modo:

URL relativo	URL absoluto
/accesset98/	http://www.uoc.edu/accesset98/
otras/pagina.html	http://www.uoc.edu/extern/ct/home/otras/pagina.html
../../logo.jpeg	http://www.uoc.edu/extern/logo.jpeg
#final	http://www.uoc.edu/extern/ct/home/home.html#final

7.3.2. Conceptos básicos del HTTP

El HTTP sigue el modelo general de peticiones y respuestas entre un cliente y un servidor. Se basa en un servicio de transporte fiable; pero, es independiente del mecanismo de transporte concreto utilizado. No obstante, lo más habitual es que cliente y servidor se comuniquen por medio del TCP. En este caso, el puerto por defecto para establecer las conexiones es el asignado oficialmente al servicio WWW, es decir, el 80.

En el HTTP/1.0, el cliente establece una conexión con el servidor y le envía un mensaje HTTP con la petición; y, a continuación, el servidor envía al cliente otro mensaje HTTP con la respuesta y cierra la conexión. Si quiere efectuar más peticiones, el cliente debe establecer una nueva conexión para cada una. En el HTTP/1.1, en cambio, es posible intercambiar diferentes peticiones y respuestas en una misma conexión que se denomina *conexión persistente*. Éste es el modo de funcionamiento por defecto en el HTTP/1.1.

Un **mensaje HTTP** consta de una primera línea, en la que hay información específica del protocolo, seguida de un mensaje con el mismo formato que los mensajes de correo electrónico, según la especificación RFC 822. Es decir, después de la primera línea debe haber una cabecera formada por una serie de campos, una línea en blanco y un cuerpo. En casos particulares, la cabecera y/o el cuerpo pueden estar vacíos, pero la línea en blanco que los separa siempre debe estar presente.

El cuerpo del mensaje, junto con los campos de la cabecera que proporcionan información sobre su contenido, forman lo que en el HTTP se denomina una *entidad*. Cada entidad corresponde a un recurso.

Dependiendo de si el mensaje HTTP es una petición o una respuesta, la primera línea recibe el nombre de *línea de petición* o *línea de estatus*, respectivamente.

La **sintaxis de una línea de petición** es la siguiente:

```
método <SP> URI <SP> versión <CRLF>
```

En esta línea, `método` especifica qué tipo de operación (o, en la terminología HTTP, qué método) solicita el cliente, `URI` identifica el recurso a que se debe aplicar la operación y `versión` debe ser la cadena `HTTP/1.0` o `HTTP/1.1`, según la versión del protocolo. El URI debe ser un URL HTTP relativo que contenga un camino absoluto; es decir, que empiece por "`/`" (excepto cuando el servidor sea un *proxy*, como veremos más adelante).

Ejemplo de utilización de un cliente para obtener un recurso

Siempre que un usuario quiera utilizar un cliente para obtener el recurso dado por la dirección `http://www.acme.com:8000/doc/principal.html`, el cliente debe conectarse al puerto 8000 del servidor `www.acme.com` y enviarle una petición que empiece con esta línea:

```
GET /doc/principal.html HTTP/1.0
```

En este caso, el método utilizado es `GET` (obtener recurso). Ahora bien, si la dirección fuera `http://www.acme.com:8000/`, la línea que debería enviarse sería simplemente la siguiente:

```
GET / HTTP/1.0
```

Versión por defecto

Si la cadena `versión` no está presente en la línea de petición, se entiende que la versión del protocolo es el HTTP/0.9. En este caso, el mensaje HTTP de petición consta sólo de esta línea; el único método permitido es `GET`, y en el mensaje HTTP de respuesta sólo puede encontrarse el cuerpo.

La **sintaxis de las líneas de estatus** es la siguiente:

versión <SP> *código* <SP> *frase* <CRLF>

La cadena `versión` es igual que en la línea de petición, el código es un número de tres dígitos, el primero de los cuales indica de qué tipo de respuesta se trata, y `frase` es un texto explicativo en formato libre, inteligible para los usuarios humanos.

Los posibles **códigos de una respuesta HTTP/1.0** son los siguientes:

Código	Significado
200	Operación efectuada.
201	Se ha creado un nuevo recurso.
202	Petición aceptada.
204	La respuesta está vacía.
301	El recurso solicitado ha cambiado de dirección.
302	El recurso solicitado ha cambiado temporalmente de dirección.
304	El contenido del recurso no ha cambiado.
400	Petición incorrecta.
401	Usuario no autorizado.
403	Acceso prohibido.
404	No se ha encontrado el recurso.
500	Error interno del servidor.
501	Operación no implementada.
502	Error en otro servidor.
503	Servicio no disponible en la actualidad.

Primer dígito del código de respuesta HTTP/1.0

En el HTTP, el significado del primer dígito del código de respuesta varía un poco de lo que se considera común (que encontraréis en el anexo 2 de este módulo didáctico) y es el siguiente:

- 1: Información.
- 2: Éxito.
- 3: Redireccionamiento.
- 4: Error en el cliente.
- 5: Error en el servidor.

En el HTTP/1.1 se han añadido a esta lista hasta 22 nuevos códigos de respuesta. No obstante, si un cliente recibe un código *xyz* que no conoce, debe interpretarlo como si fuera *x*00. En este sentido, el código 100 significa 'continuar' y el código 300 quiere decir 'recurso accesible en una dirección o más'.

Almacenamiento local de respuestas (memoria caché)

Para optimizar el tráfico de la red, es habitual que los clientes HTTP guarden una copia de las respuestas que reciben de los servidores, junto con el URL que han utilizado en la petición. El almacén destinado a este fin recibe el nombre de **memoria caché**. Si el usuario quiere volver a acceder a un recurso que ya había solicitado previamente, el cliente puede utilizar la copia de la memoria caché en lugar de volver a enviar la misma petición al servidor.

El cliente puede decidir, o el usuario puede configurar, el tiempo que se guardarán los elementos de la memoria caché, su dimensión máxima o la política que deberá seguirse para borrar sus elementos cuando esté llena. Asimismo, el HTTP también proporciona directrices para determinar qué recursos se pueden guardar en la memoria caché y cuáles no, o cuánto tiempo se puede guardar como máximo un determinado recurso.

Sin embargo, no sólo los clientes pueden guardar copias de las respuestas obtenidas de los servidores; también pueden hacerlo los servidores intermediarios o *proxies*.

Consultad los servidores intermediarios en el subapartado 7.3.4 de este módulo didáctico.

Cabeceras de los mensajes HTTP

Los campos que puede haber en la cabecera de un mensaje HTTP se pueden clasificar en cuatro grupos:

1) Campos generales que pueden estar presentes tanto en los mensajes de petición, como en los de respuesta.

2) Campos referentes a la entidad contenida en el cuerpo del mensaje y que también pueden estar presentes en peticiones y respuestas.

3) Campos propios de las peticiones.

4) Campos propios de las respuestas.

El HTTP proporciona un mecanismo de extensión que permite incluir campos no estándar en las cabeceras. Si una implementación no reconoce un determinado campo, lo que debe hacer es simplemente ignorarlo (o retransmitirlo tal como lo encuentra si es un servidor intermediario).

Los campos que puede contener la cabecera son los siguientes:

1) Campos generales definidos en HTTP/1.0:

a) `Date: fecha`. Indica la fecha y la hora en que se originó el mensaje. Los valores de los campos HTTP que representan fechas se deben expresar de una de estas tres maneras, siempre referidas al tiempo universal (`TU`, anteriormente conocido como **hora del meridiano de Greenwich** o `GMT`):

```
Fri, 06 Dec 2002 09:22:15 GMT (RFC 1123, forma recomendada)
Friday, 06-Dec-2002 09:22:15 GMT (RFC 850)
Fri Dec 6 09:22:15 2002 (formato ANSI C)
```

b) `Pragma: 1#directriz`. Este campo sirve para proporcionar una lista de directrices a los sistemas que intervienen en la transferencia (clientes o servidores). La semántica de las directrices depende de las implementaciones; sin embargo, hay una definida por el protocolo, que se representa con la palabra `no-cache`. Si un mensaje de petición contiene esta directriz, significa que debe enviarse la petición al servidor correspondiente aunque exista una copia de la respuesta guardada en la memoria caché.

La especificación HTTP/1.1

La especificación 1.1 añade cinco nuevos campos generales:

- `Cache-Control`: directrices sobre la política de memoria caché.
- `Connection`: opciones de conexión. La opción `close` indica que el emisor cerrará la conexión después de que se haya enviado la respuesta.
- `Transfer-Encoding`: codificación aplicada al cuerpo del mensaje.
- `Upgrade`: versiones del protocolo soportadas.
- `Via`: intermediarios por los que ha pasado el mensaje.

2) Campos referentes a la entidad definidos en el HTTP/1.0:

a) `Content-Length: 1*dígito`. Indica la longitud en bytes del cuerpo de la entidad. Si este campo no está presente en una respuesta, el cliente sólo puede saber cuál es el final del cuerpo cuando el servidor cierra la conexión. En un mensaje de petición que incluya un cuerpo, el campo `Content-Length` es obligatorio, puesto que de otro modo el servidor no podría enviar la respuesta y no se podría cerrar la conexión.

b) `Content-Type: tipo`. Indica el tipo de contenido del cuerpo de la entidad (por ejemplo, `text/html` si es un documento HTML). El valor de este campo se representa según la especificación `MIME` y, por tanto, puede incluir parámetros (por ejemplo, `charset=ISO-8859-1` si los caracteres están codi-

Consultad los nuevos campos de cabecera en el subapartado 5.5.1 de este módulo didáctico.

ficados según el alfabeto latín1). Todos los mensajes HTTP con cuerpo deberían incluir el campo `Content-Type`.

c) `Content-Encoding:` *`codificación`*. Indica si se ha aplicado alguna transformación al contenido de la entidad (por ejemplo: `x-gzip`). El valor de este campo se representa según la especificación `MIME`.

d) `Last-Modified:` *`fecha`*. Indica la fecha y la hora en que el recurso contenido en el cuerpo se modificó por última vez.

e) `Expires:` *`fecha`*. Indica la fecha y la hora a partir de la cual el contenido del cuerpo se puede considerar obsoleto o caducado al efecto de su almacenamiento en la memoria caché. La presencia de este campo significa que, posiblemente, el recurso se modificará en la fecha indicada o dejará de existir; sin embargo, no implica que los cambios, si se producen, deban efectuarse necesariamente en esta fecha. Ahora bien, si la fecha de caducidad es igual o anterior a la especificada en el campo `Date`, la entidad no debe almacenarse en la memoria caché.

f) `Allow:` *`1#método`*. Indica los métodos HTTP que se pueden aplicar al recurso solicitado.

La especificación HTTP/1.1

La especificación 1.1 añade seis nuevos campos de entidad adicionales:

- `Content-Base`: dirección base para interpretar URL relativos.
- `Content-Language:` lenguaje que se ha utilizado.
- `Content-Location`: URI de la entidad, en caso de que el recurso correspondiente disponga de más de una.
- `Content-MD5`: secuencia de bits para comprobar la integridad del contenido.
- `Content-Range`: por si una entidad se envía en diferentes fragmentos.
- `ETag`: etiqueta asociada a la entidad, por si el recurso dispone de más de una.

3) Campos propios de las peticiones HTTP/1.0:

a) `From:` *`dirección`*. Este campo contiene la dirección de correo electrónico del usuario que solicita el recurso.

b) `User-Agent:` *`1*implementación`*. Este campo permite identificar la implementación del cliente. Se expresa como una lista de "productos" (por ejemplo, tipo de navegador, librerías de soporte que utiliza, etc.) con números de versión opcionales (separados de los nombres con "`/`"), que puede incluir comentarios entre paréntesis. El servidor puede utilizar esta información para generar estadísticas, detectar qué implementaciones producen ciertos errores, adaptar las respuestas al tipo de cliente, etc.

Ejemplos de lista de productos

- `Lynx/2.8rel.2 libwww-FM/2.14`.
- `NCSA_Mosaic/2.6 (X11;SunOS 4.1.4 sun4m) libwww/2.12`.
- `Harvest/1.5.17`.

c) **`Referer: URI`**. Si el usuario selecciona un enlace de un documento HTML o de otro recurso que tiene dirección propia, el cliente puede incluirla en el campo `Referer` del mensaje de petición. El servidor puede utilizar esta información para generar estadísticas, detectar desde qué recurso se referencia una dirección incorrecta u obsoleta, etc.

d) **`If-Modified-Since: fecha`**. Cuando el cliente ya dispone de una copia del recurso solicitado en la memoria caché, puede utilizar este campo para llevar a cabo una operación `GET` condicional: si el recurso se ha modificado con posterioridad a la fecha indicada, se efectuará la operación de manera normal y, si no, el servidor enviará una respuesta sin cuerpo y con el código 304. Este campo sólo es aplicable al método `GET`.

e) **`Authorization: esquema#parámetro`**. Con este campo, un usuario puede presentar sus credenciales a un servidor (por norma general, un nombre y una contraseña) para que le permita acceder a recursos no públicos, es decir, de acceso restringido. La primera parte de la credencial indica el esquema de autenticación a utilizar. El HTTP/1.0 sólo define un esquema denominado *básico*, pero se puede utilizar cualquiera mientras sea conocido por el cliente y el servidor. Si se utiliza el esquema básico, el valor de este campo debe ser la palabra `Basic` seguida de la cadena de caracteres que resulta de codificar en `Base64` el nombre de usuario y su contraseña separados por "`:`".

Consultad la codificación en Base64 en el subapartado 5.5.1 de este módulo didáctico.

Ejemplo de codificación

Si el nombre de usuario es `webmaster` y su contraseña es `Secret`, se codificará en `Base64` la cadena `webmaster:Secret`. El campo quedaría de este modo:

```
Authorization: Basic d2VibWFzdGVyOlNlY3JldA==
```

Obviamente, la descodificación de este campo es trivial y, por tanto, no hay protección contra terceros que tengan la posibilidad de inspeccionar los mensajes y deseen obtener un acceso no autorizado. Para conseguir que el método de autenticación sea más seguro, se puede utilizar un esquema más sofisticado o alguna variante del protocolo basada en el cifrado de los datos, por ejemplo el HTTPS.

La respuesta a una petición que incluya el campo `Authorization` no debería guardarse en la memoria caché.

La especificación HTTP/1.1

En la especificación 1.1 se añaden los campos de petición siguientes:

`Accept`: tipo de contenido que el cliente puede aceptar.

- `Accept-Charset`.
- `Accept-Encoding`.
- `Accept-Language`.

- `Host`: nombre del servidor a quien va dirigida la petición, por si tiene más de uno; este campo, obligatorio en las peticiones HTTP/1.1, permite que un ordenador con diferentes nombres actúe como si fuera diferentes servidores al mismo tiempo.
- `If-Match`: permite comparar entidades por sus etiquetas.
- `If-None-Match`.
- `If-Range`.
- `If-Unmodified-Since`.
- `Max-Forwards`.
- `Proxy-Authorization`.
- `Range`: sirve para solicitar un fragmento de una entidad.

4) Campos propios de las respuestas HTTP/1.0:

a) **`Server:`** ***`1*implementación`***. Este campo es similar a `User-Agent`, pero referido al servidor.

b) **`Location:`** ***`URI`***. Indica que el recurso solicitado debe obtenerse en otra dirección. Éste será el caso cuando el código de respuesta sea 301 ó 302. Por norma general, cuando el cliente reciba una respuesta que incluya este campo, generará de manera automática una nueva petición con la dirección indicada.

c) **`WWW-Authenticate:`** ***`1#(esquema`*** **`realm = "`*** ***`reino`*** **`" *( , `*** ***`parámetro`*** **`))`**. Este campo es obligatorio cuando el código de las respuestas es 401. Indica que es necesario presentar una credencial para acceder al recurso solicitado, o que la que se ha presentado no es válida.

Un servidor puede agrupar sus recursos de acceso restringido en reinos, cada uno con su esquema de autenticación y conjunto de usuarios autorizados (con una misma credencial debería poderse acceder a todos los recursos de un reino). El valor de este campo es una lista con los nombres de los reinos aplicables al recurso solicitado, junto con sus esquemas y parámetros opcionales para poder llevar a cabo la autenticación con el campo `Authorization` de la petición.

La especificación HTTP/1.1

En la especificación 1.1 se añaden los campos de respuesta siguientes:

- `Age`.
- `Proxy-Authenticate`.
- `Public`: lista de métodos soportados por el servidor.
- `Retry-After`.
- `Vary`: lista de campos de la petición que se han utilizado para seleccionar una entidad, cuando el recurso dispone de más de una.
- `Warning`: información adicional sobre el estatus de la respuesta.

7.3.3. Métodos del servicio HTTP

Paso de los valores de los parámetros

Cuando el camino de un URI identifica un programa, la manera como se le pasan los valores de los parámetros o las consultas es un asunto local del servidor. Por ejemplo, un mecanismo utilizado habitualmente es el denominado CGI (*Common Gateway Interface*).

El protocolo HTTP/1.0 define los tres métodos siguientes:

1) Método `GET`. Este método sirve para obtener la entidad correspondiente al URI especificado en la línea de petición. Por norma general, el servidor traducirá el camino del URI a un nombre de fichero o de programa:

- En el primer caso, el cuerpo de la entidad será el contenido del fichero.
- En el segundo caso, el servidor ejecutará el programa y la entidad será el resultado que genere.

Los componentes `parámetro` y/o `consulta` del URI se pueden utilizar como argumentos del programa.

2) Método `HEAD`. Este método es igual que el `GET`, excepto que en la respuesta el cuerpo estará vacío y, por tanto, sólo tendrá cabecera (que deberá ser idéntica a la que se habría enviado si el método fuera el `GET`). Por norma general, se utiliza el método `HEAD`, por ejemplo, para comprobar si un URL es válido o para obtener información sobre un recurso sin necesidad de transferir su contenido.

3) Método `POST`. Este método sirve para enviar una entidad que el servidor debe incorporar al recurso identificado por el URI de la línea de petición. La semántica de este método depende del tipo de recurso de que se trate. Por ejemplo, se puede utilizar para añadir contenido a un recurso existente, para enviar un mensaje a un tablón de anuncios o un grupo de noticias, para crear un nuevo registro en una base de datos, para pasar datos a un programa que debe ejecutarse en el servidor, etc. Un caso típico de este último ejemplo son los datos de un formulario HTML.

El código de respuesta a una operación `POST` por norma general será 201, si como resultado, se ha creado un nuevo recurso (en este caso, el cuerpo de la respuesta debería contener una referencia a este recurso), o bien 200 ó 204 si no se ha creado ninguno (el cuerpo de una respuesta 200 contendrá una descripción del resultado obtenido, y el de una respuesta 204 simplemente estará vacío).

Una propiedad del método POST es que, si se envían dos peticiones iguales, el resultado de la segunda no debe ser necesariamente el mismo que el de la primera. Por tanto, la respuesta a una operación `POST` no debería guardarse en la memoria caché.

La especificación HTTP/1.1

En el HTTP/1.1 se han añadido una serie de métodos nuevos, entre lo cuales, los siguientes:

- `PUT`: para crear un recurso con el URI especificado en la petición.
- `DELETE`: para borrar un recurso.
- `OPTIONS`: para obtener información sobre las opciones de transferencia.
- `TRACE`: para obtener una copia del mensaje como ha llegado a su destino final.

7.3.4. Intermediarios: *proxies* y pasarelas

El modelo general del HTTP no sólo considera la conexión directa de un cliente con el servidor, sino también la conexión por medio de servidores intermediarios, como es el caso de los *proxies*:

En la configuración con *proxy*, el cliente establece la conexión con el servidor *proxy* que, a su vez, establece otra conexión, como cliente, con el servidor final. Cuando el *proxy* recibe el mensaje de petición del cliente, puede generar una respuesta propia o retransmitirla al servidor final. En el segundo caso, el *proxy* puede introducir modificaciones en la petición, según la aplicación para la que esté diseñado, y, cuando reciba la respuesta del servidor final, la retransmitirá al cliente, también con la posibilidad de efectuar cambios.

Asimismo, es posible que haya más de un *proxy* en la cadena; es decir, que el primer *proxy* no se conecte directamente al servidor final, sino por medio de otro *proxy*, y así sucesivamente.

En las peticiones que un cliente (u otro *proxy*) envía a un *proxy*, existe una variación respecto al caso de la conexión directa de cliente a servidor: el URI de la línea de petición no debe ser un URL HTTP relativo, sino que debe ser un URI absoluto. De lo contrario, el *proxy* no sabría cuál es el servidor final al que va destinada la petición.

El uso de un servidor *proxy* tiene diferentes aplicaciones. Las principales son las siguientes:

a) Actuar como cortafuegos que aísle una red local del resto de las redes. En esta configuración, los clientes no tienen acceso directo al exterior de su red, y toda comunicación con los servidores remotos tiene lugar por medio del *proxy*. Ello permite minimizar el riesgo de que usuarios externos comprometan la seguridad de los sistemas locales (accesos no autorizados, sabotajes, etc.).

b) Tener una memoria caché compartida entre los usuarios de la red local. Si diferentes clientes solicitan directamente un mismo recurso, por norma general guardarán la misma copia de la respuesta en sus respectivas memorias ca-

ché. Si lo solicitan por medio de un *proxy*, la primera petición necesitará un acceso al servidor remoto; sin embargo, las siguientes quizá puedan aprovechar la copia ya guardada en la memoria caché, aunque provengan de clientes diferentes.

c) Construir una jerarquía de memorias caché de *proxies*: en el nivel más bajo se encuentran los *proxies* a que acceden directamente los clientes, en un segundo nivel existen los *proxies* a que acceden los del primer nivel, etc. Incluso puede haber *proxies* a escala de todo un país. Hay un protocolo denominado **ICP*** que permite coordinar los diferentes servidores *proxy* de una jerarquía.

*** ICP es la sigla de *Internet cache protocol.***

Además de los *proxies*, existe otro tipo de servidor intermediario llamado **pasarela**. Las pasarelas actúan como si fueran el servidor final, con la diferencia respecto al *proxy* de que el cliente no sabe que no se conecta directamente al servidor final. Una pasarela se puede utilizar como cortafuegos en la red del servidor o para traducir las peticiones HTTP de manera que se pueda acceder a recursos almacenados en sistemas no HTTP.

8. Acceso al servicio de directorio

El servicio de directorio X.500 permite el almacenamiento de todo tipo de información de manera distribuida. El X.500 define la estructura de la información, así como los protocolos necesarios para acceder a la misma.

Además de los protocolos definidos por el OSI, existe el **LDAP***, que es un protocolo diseñado para proporcionar acceso al servicio de directorio X.500 de manera simple. El LDAP está orientado a aplicaciones que sólo necesitan acceso simple de lectura/escritura al directorio y, por norma general, sobre conexiones TCP/IP.

*** LDAP es la sigla de *lightweigth directory access protocol.***

El LDAP sólo describe el protocolo, y no el modelo de información, puesto que se utiliza el que hay definido para el X.500. Por este motivo, antes de describir el LDAP, es preciso explicar algunas de las características definidas para el X.500 que también se aplican al LDAP.

8.1. El X.500: el servicio de directorio

El servicio de directorio surge de la necesidad que tienen las aplicaciones de disponer de información sobre diferentes tipos de objetos. El directorio opera como un sistema integrado único en el ámbito mundial de almacenamiento de información sobre objetos. Se basa en la colaboración entre sistemas individuales, cada uno de los cuales almacena la información de parte de los objetos y la proporciona al resto de los sistemas. El servicio de directorio también aporta el mecanismo para que tanto personas como procesos puedan acceder a esta información.

Lectura complementaria

Si queréis más información sobre el X.500, consultad la obra siguiente:

ISO 9594 / ITU-T Serie de recomendaciones X.500, Information technology - Open Systems Interconnection - The Directory

El servicio de directorio no se ha ideado como una base de datos de propósito general, sino que intenta proporcionar un mecanismo de almacenamiento de información distribuida. Por ello, el servicio de directorio se basa en los tres supuestos siguientes:

Directorio y DNS

Aunque el servicio de directorio puede parecer similar al servicio de nombres de dominio (DNS), este último sólo tiene información sobre ordenadores y dominios, mientras que el primero puede incluir información sobre cualquier tipo de objeto, así como sobre el protocolo para acceder a la misma.

- Las peticiones de información son mucho más frecuentes que las actualizaciones.
- Las condiciones transitorias, con información no del todo consistente, pueden ser usuales.
- Para acceder a la información, se utilizan nombres con estructura jerárquica, y no la dirección en que se encuentra la información.

8.1.1. El modelo de información

El modelo de información del directorio se fundamenta en la base de información del directorio (DIB*) y en el árbol de información del directorio (DIT**).

*** DIB es la sigla de *directory liformation base.***
**** DIT es la sigla de *directory information tree.***

1) Base de información del directorio

El directorio contiene información sobre objetos y hace que sean accesibles para los usuarios. Cada objeto está representado dentro del directorio por una única entrada, que contiene toda la información sobre el objeto. El conjunto completo de entradas de que dispone el directorio constituye la **base de información del directorio (DIB)**.

2) Árbol de información del directorio

A causa de la ingente cantidad de entradas que hay en el directorio, se han distribuido de acuerdo con una jerarquía natural de organizaciones y se han ordenado dentro de un árbol, el **árbol de información del directorio (DIT)**. Conviene destacar que hay un único DIT a escala mundial, que incluye absolutamente todas las entradas que pueden ser accesibles por medio del X.500.

Cada vértice del DIT (no hay vértice raíz) corresponde a una entrada que se refiere a un objeto: en los vértices se encuentra la información asociada a cada objeto. De la misma manera, cada entrada puede tener una serie de hijos asociados. Los vértices están estructurados jerárquicamente y agrupados a partir de clases, las cuales se suelen identificar con un código de una o dos letras de la manera siguiente: países (C, *country*), organizaciones (O, *organization*), localidades (L, *locality*), unidades organizativas (OU, *organizational unit*) y personas (CN, *common name*).

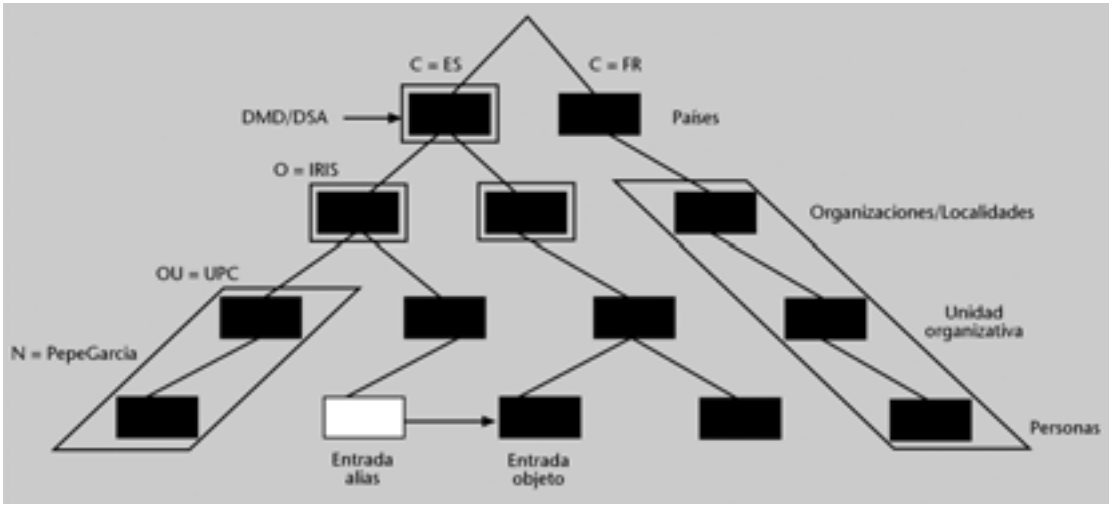
El X.500 define dos tipos de entradas: 

- **Entradas objeto**: entradas que contienen información.

- **Entradas alias**: entradas que sirven para referenciar cualquier entrada objeto dentro el DIT, excepto otras entradas alias.

Un objeto siempre se encuentra representado por una sola entrada objeto pero esta entrada (tanto si es entrada hoja, como no hoja) puede ser referenciada por una entrada alias o más. A diferencia de las entradas objeto que figuran en el DIT, las entradas alias son siempre entradas hoja; es decir, se encuentran en el último nivel del árbol.

A continuación, mostramos la estructura del árbol de información del directorio (DIT), donde se aprecian los diferentes tipos de entradas y su estructura jerárquica:



8.1.2. Entradas de directorio

Cada entrada contiene el conjunto de toda la información conocida sobre el objeto que representa. Las entradas del directorio están formadas por una serie de atributos. Cada atributo consta de los campos siguientes:

- Un tipo de atributo (*attribute type*): identifica la clase de información que hay en el atributo.

  Existen dos tipos de atributo que se utilizan en la gestión interna del directorio:

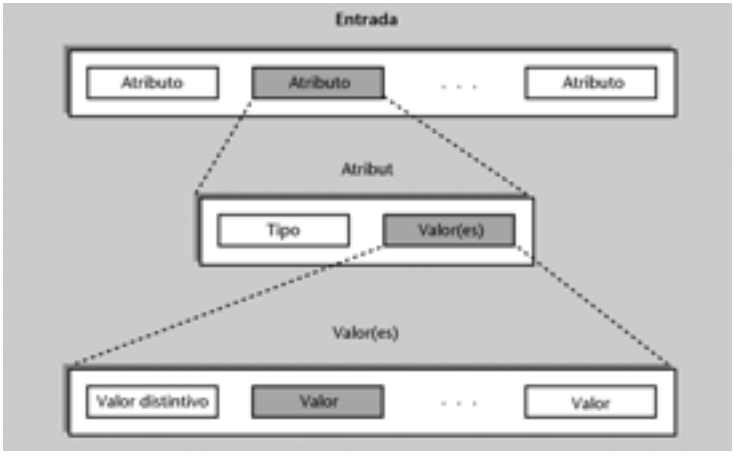
  - *ObjectClass:* se encuentra en todas las entradas, incluyendo los alias, e identifica el tipo (clase) de objeto.
  - *AliasedObjectName:* se encuentra en todas las entradas alias y contiene el nombre de la entrada objeto a la que se refiere el alias.

- Un valor de atributo o más (*attribute value*): incluye uno o más valores del atributo, uno de los cuales puede ser lo que se denomina ***valor distintivo***.

Tipos de atributos

ObjectClass
AliasedObjectName
CountryName (C)
Organization (O)
LocalityName
OrganizationalUnit (OU)
CommonName (CN)
BusinesCategory
PostalAdress
PostalCode
PostOffceBox
Surname
TelephoneNumber
StateOrProvinceName

La figura siguiente representa la estructura de las entradas de directorio:



8.1.3. Nombres de directorio

El X.500 utiliza un método de nombres para identificar la entrada correspondiente a cada objeto dentro del directorio y poder acceder a la información del mismo. Por este motivo, cada objeto (y, por tanto, cada entrada) posee un **nombre relativo de directorio** (**RDN**, *Relative Directory Name*) y un **nombre de directorio** (**DN**, *Directory Name*).

1) Nombre relativo de directorio

El nombre relativo de directorio (RDN) sirve para diferenciar un objeto (y, por tanto, una entrada) de sus hermanos dentro del DIT. Se forma a partir de los atributos de una entrada que tienen valor distintivo. Por este motivo, estos últimos deben ser persistentes y tener pocas probabilidades de quedar obsoletos.

Si una entrada sólo tiene un atributo con valor distintivo, el RDN está formado por un único par `tipo=valor`. En otros casos, estará formado por tantos pares `tipo=valor` como sea necesario para diferenciar entradas a un nivel particular del DIT. Informalmente, estos pares y el RDN correspondiente se pueden expresar con la notación siguiente:

```
tipo=valor
RDN {tipo=valor, tipo=valor...}
```

2) Nombre de directorio

El nombre de directorio (DN) permite diferenciar un objeto del resto de objetos del directorio. Como en muchos de los casos en que hay una estructura jerárquica, el nombre de directorio está integrado por la secuencia de RDN que va desde la raíz, pasando por los vértices intermedios, hasta la entrada que representa un objeto. La representación es similar al caso del RDN:

```
RDN {tipo=valor...;
     tipo=valor...; ...}
```

Es conveniente señalar que el nombre distintivo de un alias no coincide con el nombre distintivo del objeto al que se refiere.

Ejemplo

En este ejemplo, se pueden ver dos nombres relativos, el segundo de los cuales está compuesto por dos pares `tipo= valor` y un nombre absoluto.

```
RDN {CN=Ramon Martí}
RDN {O=UOC, L=Barcelona}
DN {C=ES; O=IRIS; OU=UOC, L=Barcelona; CN=Ramon Martí}
```

8.1.4. Modelo del X.500

El X.500 sigue el modelo típico de aplicación cliente/servidor y consta de los elementos siguientes:

- **Usuario**: es el usuario que quiere acceder al X.500 a partir del cliente correspondiente.

- **Agente de usuario del directorio** (**DUA***): es la parte que actúa como cliente del X.500.

  * DUA es la sigla de *directory user agent*.

- **Agente de servicio del directorio** (**DSA***): es el servidor X.500. Como tal, hace de punto de acceso al directorio para los usuarios, así como es el encargado de almacenar y gestionar parte de las entradas del DIT. Sin embargo, los diferentes DSA interaccionan entre sí para proporcionar de forma distribuida el servicio de directorio.

  * DSA es la sigla de *directory service agent*.

8.1.5. Protocolos X.500

Los elementos que hemos descrito en el subapartado anterior se comunican por medio de los dos protocolos siguientes:

- **Protocolo de acceso al directorio** (**DAP***): es el protocolo que utilizan los DUA para acceder a los DSA.

  * DAP es la sigla de *directory access protocol*.

- **Protocolo de sistema del directorio** (**DSP***): es el protocolo utilizado por los DSA para colaborar entre sí.

  * DSP es la sigla de *directory system protocol*.

La figura siguiente presenta el modelo X.500, en que se pueden ver los diferentes elementos y protocolos definidos en el estándar:

8.1.6. Dominios de gestión

Tal como ya se ha dicho, el servicio de directorio es un servicio distribuido, por lo que la información también se encuentra distribuida y es gestionada por diferentes organizaciones. El X.500 define lo que se denomina ***dominio de gestión del directorio*** (**DMD***), que es una parte del servicio directorio gestio-

  * DMD es la sigla de *directory management domain*.

nada por una única entidad. Desde el punto de vista del modelo, un DMD por norma general corresponderá a un DSA (o más), con los DUA que gestiona.

Al mismo tiempo, desde el punto de vista de la información, cada dominio de gestión corresponde a una parte del DIB (o DIT), que es la parte almacenada y gestionada por el DSA asociado al dominio. Es preciso decir que el protocolo X.500 define dos categorías de dominios de gestión del directorio:

DMD administrativos (ADDMD*), gestionados por la Administración.

DMD privados (PRDMD**), gestionados por organizaciones privadas.

*** ADDMD es la sigla de *administrative DMD*.**
**** PRDMD es la sigla de *private DMD*.**

8.1.7. Funcionalidad del protocolo X.500

La funcionalidad proporcionada por el X.500 se puede clasificar en los cuatro conjuntos (lectura, búsqueda, modificación y conexión) que soportan los dos protocolos definidos en el X.500, aunque de manera diferente:

- **DAP**: en este caso, el DUA puede pedir todas las operaciones al DSA (y no al revés).
- **DSP**: en este caso, los DSA se pueden requerir las operaciones entre sí.

A continuación, se describen los diferentes conjuntos y las operaciones que incluyen:

1) Lectura

Incluye las operaciones que permiten el acceso a las entradas y a sus atributos.

- **Lectura**: el X.500 permite extraer información sobre una entrada (por norma general, valores de los atributos deseados) por medio de la operación `Read`.
- **Comparación**: en ocasiones, no se necesita la información de una entrada, sino sólo comprobar si un atributo de un elemento tiene un valor especificado. Para esta funcionalidad, el X.500 dispone de la operación `Compare`.
- **Abandono de búsqueda**: una vez iniciada una búsqueda de información, puede interesar abandonarla antes de que haya acabado. La operación `Abandon` proporciona esta funcionalidad.

2) Búsqueda

Esta funcionalidad posibilita la búsqueda de entradas a partir de criterios de selección determinados.

- **Listado**: el X.500 proporciona un método para obtener el listado de las entradas inmediatamente subordinadas a una entrada determinada. Este método es la operación `List`.

- **Búsqueda**: cuando se tiene información, con frecuencia se quieren obtener atributos específicos de las entradas que satisfacen los criterios de selección deseados. Esta búsqueda se lleva a cabo con la operación `Search`.

3) Modificación

Sirve para modificar las entradas y sus atributos, incluyendo también los atributos con valor distintivo.

- **Añadir una entrada**: para crear una nueva entrada dentro de la base de información del directorio, el X.500 proporciona la operación `Add-entry`.

- **Eliminación de una entrada**: si lo que se quiere es eliminar una entrada, el X.500 tiene la operación `Remove-entry`.

- **Modificación de una entrada**: el X.500 también tiene en cuenta que en ocasiones interesa modificar los atributos de una entrada. Por este motivo, define la operación `Modify-entry`.

- **Modificación de RDN**: cuando conviene modificar los valores distintivos que forman el RDN de una entrada, es preciso utilizar una operación específica, que es `Modify-RDN`.

4) Conexión

Permite la conexión y la desconexión.

- **Conexión**: el X.500 define la operación `Bind` para permitir la conexión.

- **Desconexión**: el X.500 define la operación `Unbind` para permitir la desconexión.

8.2. Modelo del LDAP

El LDAP es el protocolo que permite el acceso al X.500 por medio de mecanismos y de protocolos de comunicación que, por norma general, se utilizan en Internet.

Lectura complementaria

Si queréis más información sobre el modelo LDAP, consultad la obra siguiente:

W. Yeong; T. Howes; S. Kille (1995, marzo). *RFC 1777 - Lightweight Directory Access Protocol*.

El modelo LDAP se basa en los elementos siguientes:

- **Cliente LDAP**, que accede a servidores LDAP utilizando el LDAP como protocolo.

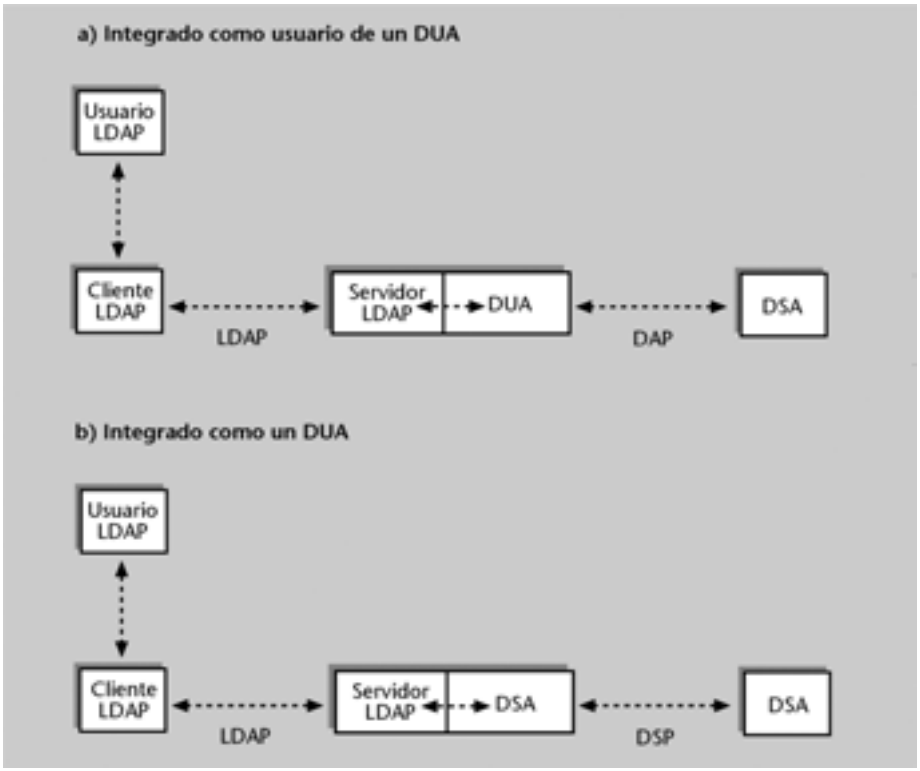
- **Servidor LDAP**, que confiere servicio a los clientes LDAP y accede al servicio X.500.

Para poder acceder al X.500 utilizando el LDAP, la **implementación del servidor** LDAP puede seguir uno de los modelos siguientes:

- El **servidor LDAP** actúa como usuario del DUA, el cual accede al X.500 utilizando el DAP.

- El **servidor LDAP** actúa como DUA, el cual accede a un DSA. En este caso, el DSA interacciona con otros DSA por medio del DSP.

La figura siguiente presenta los dos modelos funcionales del LDAP integrado con el acceso al X.500, uno como usuario del DUA y el otro como DUA:



8.3. Funcionalidad del LDAP

El protocolo LDAP utiliza conexiones TCP sobre el puerto 389. Cada una de ellas puede incluir tantos comandos como el usuario necesite. Una vez se ha accedido a toda la información que se quiere, se cierra la conexión.

El LDAP proporciona la misma funcionalidad que el DAP, excepto la **lectura** y el **listado**, que no los soporta. Por lo que respecta a los argumentos, el LDAP soporta un subconjunto de los argumentos proporcionados por el X.500:

- **Inicio de sesión**: indica el principio de una sesión entre un cliente y un servidor, y la autenticación del cliente. Debe ser la primera operación.

- **Finalización de sesión**: cierra la sesión.

- **Búsqueda**: solicita al servidor que realice una búsqueda en el directorio.

- **Modificación**: solicita que se modifique el DIB.

- **Inclusión**: solicita que se añada una nueva entrada al directorio.

- **Eliminación**: solicita el borrado de una entrada del directorio.

- **Modificación RDN**: solicita que se cambie el RDN de una entrada del directorio.

- **Comparación**: solicita que se compare un valor con una entrada del directorio.

- **Abandono**: solicita que el servidor abandone una petición pendiente.

9. Protocolos Internet en tiempo real

En la actualidad, hay una serie de aplicaciones, como la transmisión de audio o vídeo en tiempo real, en que es preciso que los datos se vayan presentando al destinatario a medida que van llegando. En estas aplicaciones, es primordial la regularidad en la transferencia de datos.

Teniendo en cuenta la cantidad de aplicaciones distribuidas en Internet que necesitan transmitir datos en tiempo real, ha surgido un conjunto nuevo de protocolos orientados a solucionar los problemas que de ello se derivan.

A continuación, se da una relación de cada uno de estos protocolos según su funcionalidad:

- RTP*: transmisión de datos en tiempo real.
- RTCP**: control de la transmisión de datos en tiempo real.
- RTSP***: transmisión de flujo de datos multimedia.
- RSVP****: reserva de recursos para la transmisión de datos en tiempo real.

*** RTP:** *Real-time Transport Protocol.*
****RTCP:** *RTP Control Protocol*
*****RTSP:** *Real Time Streaming Protocol.*
******RSVP:** *Resource reSerVation Protocol.*

9.1. El estándar RTP

El **RTP** es un estándar que proporciona servicios de entrega punto a punto para datos con características de tiempo real como audio y vídeo.

Para iniciar una sesión RTP que incluya el transporte de datos y el control de la conexión, es preciso disponer, además de una dirección de red, de dos puertos: uno para el RTP y otro para el RTCP, los dos protocolos de que consta el estándar.

9.1.1. El RTP

El **protocolo de transporte de tiempo real** (**RTP**) proporciona facilidades para el transporte de datos en tiempo real. Entre los que encontramos los siguientes: la reconstrucción temporal, la detección de pérdidas, la identificación de contenido y la seguridad.

Lectura complementaria

Si queréis más información sobre el RTP,consultad la obra siguiente:

Audio-Video Transport Working Group; H. Schulzrinne; S. Casner; R. Frederick; V. Jacobson (1996, enero). *RFC 1889 - RTP: A Transport Protocol for Real-Time Applications.*

Por norma general, el RTP se utiliza sobre redes de tipo UDP/IP. No obstante, se ha llevado a cabo todo tipo de esfuerzos para conseguir que sea independiente del mecanismo de transporte. El RTP utiliza los servicios de multiplexación y *checksum* del UDP. En las sesiones multimedia, en que es preciso

transmitir diferentes tipos de datos al mismo tiempo, se necesita una sesión RTP separada para cada uno de los tipos de datos.

El RTP no está orientado a la reserva de recursos o al control de la calidad del servicio. Para estas funciones, es conveniente utilizar protocolos de reserva de recursos, como el RSVP.

El RTP proporciona un formato a los datos que deben transmitirse. En el RTP se ha definido un formato que incluye una cabecera en la que se encuentran los diferentes campos que proporcionan soporte para las necesidades de los datos en tiempo real:

- **Identificación del tipo de datos**: sirve para conocer el tipo de datos.
- **Número de secuencia**: permite la detección de pérdidas.
- **Sello temporal**: posibilita la reconstrucción temporal.

9.1.2. El RTCP

El **protocolo de control de tiempo real** (**RTCP**) proporciona soporte para conferencias en tiempo real de grupos de cualquier tamaño: identificación de la fuente de los datos y los elementos de soporte para pasarelas, tales como: *bridges* de audio y vídeo, y traductores de *multicast* a *unicast*.

Del mismo modo que el RTP, el RTCP se utiliza sobre redes de tipo UDP/IP.

El RTCP proporciona información sobre la calidad del servicio de los receptores en grupos *multicast*, así como el soporte para la sincronización de diferentes flujos de datos de distintos medios. El RTCP lleva a cabo esta tarea definiendo unos paquetes de control que cada uno de los participantes en una sesión RTP envía periódicamente al resto de participantes. La información recibida se puede utilizar para el control del rendimiento, para tareas de diagnóstico o para otras funciones.

El RTCP da soporte a las funciones siguientes:

a) Provisión de información a la aplicación

La principal función del RTCP es proporcionar información a las aplicaciones sobre la calidad de la distribución de sus datos. Por este motivo, cada paquete RTCP contiene informes del emisor y/o del receptor con estadísticas útiles para las aplicaciones. Dichas estadísticas incluyen el número de paquetes enviados, el número de paquetes perdidos, el *jitter* de interllegada, etc.

La recepción de esta información sobre la calidad de los datos es útil para las aplicaciones, puesto que pueden ir modificando los parámetros de transmisión para mejorar la calidad de su servicio.

b) Identificación de la fuente RTP

El RTCP lleva un identificador de nivel de transporte para la fuente RTP. Este último sirve a los receptores para asociar diferentes flujos de datos del mismo participante, pero de sesiones RTP diferentes.

Recordad

En las sesiones multimedia, cada tipo de dato necesita una sesión RTP separada.

c) Control del intervalo de transmisión RTCP

Para prevenir que el tráfico de control (paquetes RTCP) no sature los recursos de la red, se limita como máximo al 5% del total del tráfico de la sesión. Este límite debe conseguirse ajustando el intervalo de tiempo que se deja pasar cuando se envían los paquetes RTCP. Es preciso que cada participante conozca el número total de participantes y calcule el intervalo a partir de este dato.

d) Envío de información mínima de control de sesión

Como función opcional, el RTCP se puede utilizar también para enviar cantidades de información pequeñas a todos los participantes de la sesión.

9.2. El RTSP

El **protocolo de flujo de tiempo real (RTSP)** es un protocolo a nivel de aplicación que proporciona el control del flujo de datos multimedia, tanto si es *unicast* como *multicast*.

Lectura complementaria

Si queréis más información sobre el RTSP, consultad la obra siguiente:

H. Schulzrinne; A. Rao; R. Lanphier (1998, abril). *RFC 2326 - Real Time Streaming Protocol (RTSP).*

El RTSP se considera más un entorno de trabajo (*framework*) que un protocolo. Está orientado al control de múltiples sesiones de entrega de datos y proporciona mecanismos para elegir canales de entrega como el UDP, el TCP/IP *multicast* así como mecanismos de entrega basados en el RTP.

Asimismo, el RTSP proporciona mecanismos de control para el establecimiento de sesión y cuestiones de licencias. El RTSP está diseñado para trabajar sobre el RTP para el control y la entrega de contenido en tiempo real. El RTSP también puede utilizar el RSVP para el establecimiento y la gestión de sesiones de flujo con ancho de banda reservado.

El RTSP es, intencionadamente, muy parecido al HTPP/1.1, pero con métodos nuevos e identificadores de protocolo diferentes. 

Con este protocolo, los datos se fragmentan en múltiples paquetes del tamaño apropiado al ancho de banda disponible entre el cliente y el servidor. Cuando el cliente ha recibido suficientes paquetes, el programa servidor puede reproducir un paquete, mientras descomprime otro y recibe un tercero. El usuario puede empezar a reproducir los datos casi inmediatamente, sin necesidad de esperar a recibirlos todos.

9.3. El RSVP

El **protocolo de reserva de recursos (RSVP)** está orientado a la calidad de servicio (QoS*). Por consiguiente, antes de explicar el protocolo es preciso saber que cuando se habla de *calidad de servicio* en las comunicaciones, por norma general, se hace referencia a aspectos como los siguientes:

* QoS es la sigla de la expresión inglesa *quality of service*.

- **Retraso**: tiempo que tardan los datos en ir desde el emisor hasta el receptor.
- ***Jitter***: variación relativa del tiempo de recepción entre diferentes muestras que deberían recibirse en intervalos fijos.
- **Ancho de banda**: cantidad de bits necesarios para enviar los datos.
- **Pérdida de datos**: porcentaje de datos que se considera razonable perder.

Lectura complementaria

Si queréis más información sobre el RSVP, consultad la obra siguiente:

R. Braden (ed.); L. Zhang; S. Berson; S. Herzog; S. Jamin (1997, septiembre). *RFC 2205 - Resource ReSerVation Protocol (RSVP) - Version 1 Functional Specification.*

Parámetros de calidad

Los parámetros de calidad de servicio necesarios varían según el tipo de datos que se transmiten. Por ejemplo, un vídeo en tiempo real puede ser sensible al retraso, al *jitter* y al ancho de banda, pero no serlo tanto a la pérdida de datos, mientras que la transferencia de un fichero puede ser sensible a la pérdida de datos y al ancho de banda, pero no serlo al retraso ni al *jitter*.

El RSVP opera sobre el IP, ocupando el sitio del protocolo de transporte; sin embargo, al mismo tiempo proporciona servicios del nivel de sesión, puesto que no transporta ningún tipo de datos. Asimismo, el RSVP utiliza protocolos de direccionamiento en niveles inferiores para determinar hacia dónde deben dirigirse las peticiones de reserva.

Funcionalidad del RSVP

Las aplicaciones utilizan el RSVP para solicitar una calidad de servicio punto a punto para un flujo de datos concreto. El RSVP es más útil en los sistemas en que la reserva de calidad de servicio se lleva a cabo recolocando recursos, que en los sistemas en que se efectúa añadiendolos.

Todos los sistemas que hay entre el cliente y el servidor por los que debe pasar el flujo de datos deben soportar el RSVP. Por consiguiente, cada sistema reserva los recursos de sistema necesarios para satisfacer la petición de calidad de servicio. Cuando una aplicación requiere una calidad de servicio para un flujo de datos concreto, el RSVP transporta la petición a través de la red, por cada uno de los nodos que deben transportar los datos. En cada nodo, el RSVP intenta reservar los recursos necesarios.

En el caso de datos multimedia, como ya se ha indicado, cada uno de los tipos de datos va en sesiones RTP separadas, y hay paquetes RTCP que controlan la

calidad de la sesión. En este caso, los direccionadores se comunican por medio del RSVP para establecer las sesiones con ancho de banda reservado y gestionarlas.

La aplicación que quiere enviar los datos es la que debe iniciar el RSVP. 

10. Seguridad en las comunicaciones

Las redes de computadores permiten llevar a cabo de una manera más cómoda y fiable muchas de las tareas "tradicionales" relacionadas con la comunicación: enviar cartas, conversar por teléfono, intercambiar documentos, rellenar formularios, etc.

Para garantizar el correcto funcionamiento de estos sistemas tradicionales de comunicación, en ocasiones es necesario adoptar ciertas medidas de seguridad, como enviar las cartas en sobres cerrados para que no puedan ser leídas por terceras personas, firmar los documentos para que quede constancia de quién los ha escrito, ponerles sellos para evitar falsificaciones, etc.

En el mundo de las comunicaciones entre ordenadores, también se pueden necesitar medidas de seguridad equivalentes. En concreto, hay dos tipos de situaciones no deseadas contra las que los sistemas pueden proporcionar alguna protección:

1) Accidentes fortuitos como un fallo del *hardware*, la supresión de datos por equivocación, etc.

2) Acciones malintencionadas de los usuarios, como sabotajes, modificación o supresión de datos, intercepción de mensajes privados, etc.

En este apartado, nos centraremos en el segundo tipo de problemas, puesto que su solución constituye la base que permite la implementación de servicios como el comercio electrónico, la notaría electrónica, etc.

10.1. Amenazas

Cuando un usuario A quiere enviar datos a un usuario B, un tercer usuario malintencionado C puede llevar a cabo las acciones hostiles siguientes:

- **Interrupción**: el usuario C puede hacer que los datos enviados por el usuario A no lleguen al usuario B.

- **Intercepción**: el usuario C puede leer los datos enviados, que podrían ser privados, pero dejar que lleguen al usuario B con normalidad.

- **Modificación**: el usuario C puede hacer que al usuario B le lleguen datos diferentes de los que ha enviado el usuario A.

- **Impostura**: el usuario A no quiere enviar datos a B, pero el usuario C se los envía haciéndole creer que se los ha enviado el usuario A.

Existen otras amenazas en las que no interviene ningún tercer usuario, que son las dos siguientes:

- **Repudio**: el usuario A envía un mensaje al usuario B, por ejemplo expresando un determinado compromiso del usuario A; sin embargo, después niega habérselo enviado.

- **Negación de recepción**: el usuario A envía un mensaje al usuario B, por ejemplo expresando una determinada obligación del usuario B; sin embargo, después B niega haberlo recibido.

10.2. Servicios de seguridad

Los sistemas de comunicación pueden ofrecer los servicios de seguridad siguientes con la finalidad de proteger a los usuarios de las amenazas descritas en el subapartado anterior:

1) Confidencialidad: consiste en codificar los datos de manera que sólo el destinatario deseado pueda descodificarlos. Así, aunque un tercer usuario intercepte los datos, no podrá interpretarlos.

2) Integridad: consiste en proporcionar un mecanismo para garantizar que los datos no han sido modificados por ningún usuario diferente del originador.

3) Autenticación: consiste en verificar la identidad de las partes que intervienen en una comunicación para que no sea posible la suplantación.

4) No rechazo: consiste en ofrecer la posibilidad de demostrar que un usuario ha enviado o ha recibido un determinado mensaje.

10.3. Mecanismos de seguridad: técnicas criptográficas

La implementación de los servicios de seguridad antes mencionados se puede conseguir por medio de la **aplicación de técnicas criptográficas**. En un sistema criptográfico se codifican los datos que se quieren proteger por medio de un **cifrado**, para que la codificación inversa o descifrado no sea trivial.

El **proceso de cifrado** consiste en aplicar una transformación a los datos. Por norma general, la correspondencia entre los datos originales y los datos cifrados no es fija, sino que varía según un parámetro denominado **clave**. La ventaja de los métodos criptográficos radica en el hecho de que no permiten

obtener los datos originales a partir de los datos cifrados, aunque el algoritmo de codificación sea conocido, si no se sabe qué clave se ha utilizado para generarlos.

Descodificación "imposible"

Cuando se dice que es "imposible" obtener unos datos a partir de otros, en realidad se quiere decir que no hay ninguna manera conocida de hacerlo en un tiempo razonable y con los recursos disponibles. Naturalmente, siempre existe la posibilidad de intentar descifrar un mensaje aplicando la "fuerza bruta"; es decir, probando todas las combinaciones posibles. Esta solución, en general, es inviable porque implica un número enorme de operaciones de gran magnitud.

Existen dos tipos de métodos de cifrado: los simétricos y los asimétricos que se describen en los subapartados siguientes.

10.3.1. Criptografía simétrica

En un método de criptografía simétrica se utiliza la misma clave para cifrar y descifrar los datos.

De este modo, para garantizar la confidencialidad de un mensaje, el usuario A lo cifra con una clave conocida por él y por el usuario B, y este último lo descifra utilizando esta misma clave.

La ventaja de esta técnica es que los algoritmos de cifrado y descifrado suelen ser sencillos y rápidos. El inconveniente es que A y B deben ponerse de acuerdo para utilizar una determinada clave. Y lo es porque si sólo se pueden comunicar mediante un medio en el que la privacidad puede estar comprometida (por ejemplo, la misma red por la que después se enviarán los mensajes cifrados), una tercera parte puede averiguar cuál es la clave elegida.

10.3.2. Criptografía asimétrica

En los algoritmos de criptografía asimétrica se necesita una clave para cifrar los datos y otra para descifrarlos. La principal propiedad de esta técnica es que, dada una de las claves, es relativamente sencillo obtener la otra, pero dada la segunda es imposible obtener la primera.

Por ello, la primera se denomina ***clave privada***, y la segunda, ***clave pública***, puesto que se puede dar a conocer a cualquier usuario. Así, cada usuario puede elegir su clave privada y, a partir de esta última, obtener la clave pública correspondiente y distribuirla a todos los usuarios con quien quiera establecer comunicaciones seguras.

La ventaja de la criptografía asimétrica es que no requiere que las partes se intercambien claves secretas. El inconveniente es que los algoritmos suelen

ser más complejos y necesitan más recursos (básicamente, tiempo de procesamiento).

Existen dos modalidades de uso de la criptografía asimétrica:

a) Si se utiliza la clave pública para cifrar un mensaje y la clave privada para descifrarlo, el originador del mismo puede tener la certeza de que sólo lo podrá leer e interpretar correctamente el destinatario.

b) Si se utiliza la clave privada para cifrar un mensaje y la clave pública para descifrarlo, todo el mundo puede leer el contenido; sin embargo, se asegura que sólo lo ha podido generar su originador: ésta es la base de las denominadas *firmas digitales*.

El método utilizado habitualmente para firmar un mensaje consiste en cifrar con la clave privada no el mensaje entero, sino un "resumen" o **compendio** lo suficientemente corto como para paliar el principal problema del cifrado asimétrico, que es su complejidad, pero suficientemente largo para que no sea factible un ataque por "fuerza bruta".

Los algoritmos para generar compendios a partir de un mensaje deben tener la propiedad de ser no reversibles; es decir, dado un valor determinado del compendio, tiene que ser imposible encontrar un mensaje cuyo compendio coincida con el valor dado.

Por tanto, el usuario que quiere firmar un mensaje primero calcula su compendio y después lo cifra con la clave privada. El compendio cifrado es la firma digital. Los usuarios que quieren verificar esta firma deben calcular el compendio, descifrar la firma y comprobar si los resultados coinciden.

Las firmas digitales son útiles para proporcionar los servicios de integridad, autenticación y no rechazo.

Resumen

En este módulo se han descrito las aplicaciones distribuidas más usuales en Internet y diferentes estándares relacionados con ellas.

1) Modelo cliente/servidor: sirve como base de la implementación de aplicaciones distribuidas. Este modelo especifica la manera como en todo programa se pueden definir una parte de usuario, una parte de aplicación cliente y una parte de aplicación servidor. Sin embargo, la parte cliente y la parte servidor pueden estar situadas en ordenadores diferentes siempre que se añada un módulo encargado de la comunicación entre ambas.

Este modelo también describe cómo puede interaccionar un conjunto de servidores para proporcionar un servicio global como sistema servidor.

2) Servicio de nombres de dominio: proporciona acceso a una base de datos distribuida por toda la red Internet que permite obtener información sobre un determinado nombre.

Las consultas más habituales son para averiguar qué dirección IP corresponde a un nombre de servidor; sin embargo, es posible también obtener el nombre del DNS o del servidor de correo de un dominio dado, el nombre de un servidor a partir de su dirección IP, la lista de alias asociados a un servidor, etc.

3) Servicios básicos: se ha descrito un conjunto de protocolos que trabajan sobre el TCP/IP, entre los que se encuentran los siguientes:

- **Telnet**: protocolo general para establecer sesiones interactivas con un servidor que permite que el cliente actúe como un terminal virtual.

- **rlogin**: protocolo de terminal virtual con facilidades para simplificar el proceso de identificación del usuario.

- **rsh**, **rexec**: protocolos para la ejecución remota de procesos en el servidor.

4) Transferencia de ficheros: una de las principales aplicaciones utilizadas en la red Internet consiste en la transferencia de ficheros entre ordenadores. En este módulo se han descrito dos protocolos de transferencia de ficheros:

- **FTP**: protocolo que permite copiar ficheros del cliente al servidor, del servidor al cliente y de un servidor a otro. También proporciona operaciones para que el cliente pueda manipular el sistema de ficheros del servidor.

- **TFTP**: protocolo que sólo permite copiar ficheros. Es mucho más simple y más fácil de implementar que el FTP, y por norma general se utiliza para que los clientes sin disco de una red puedan obtener del servidor los ficheros necesarios para arrancar el sistema operativo.

5) Correo electrónico: es un servicio basado en los conceptos del correo postal que adopta su filosofía de almacenamiento y reenvío.

La **norma RFC 822** especifica el formato de los mensajes que se utilizan en el correo electrónico. La norma **MIME** define un conjunto de extensiones para el formato especificado por la norma RFC 822.

Tres protocolos proporcionan las diferentes funcionalidades necesarias para el correo electrónico:

- **SMTP**: transferencia de mensajes.
- **POP3**: acceso simple a buzones de correo.
- **IMAP4rev1**: acceso complejo a buzones de correo.

6) Servicio de noticias: facilita la publicación de mensajes con un alcance que puede ir desde el ámbito local de una organización, hasta el ámbito mundial. Se basa en la propagación de los mensajes entre los diferentes servidores por medio del **NNTP**. Este mismo protocolo puede ser utilizado por los clientes para acceder a los mensajes guardados en un servidor o para enviar otros nuevos.

7) Servicio hipermedia: el servicio WWW permite "navegar'' por un conjunto de elementos de información interconectados con referencias o enlaces entre sí que pueden contener datos con diferentes tipos de representaciones (texto, imágenes, audio, vídeo). El protocolo que proporciona el acceso a esta información es el **HTTP**. Existen diferentes técnicas que permiten representar la información hipermedia, como el lenguaje **HTML**, basado en la sintaxis definida en el estándar **SGML**.

8) Servicio de directorio: permite el almacenamiento y el acceso a información de todo tipo a partir de la definición de un sistema remoto global, en el que diferentes servidores interaccionan para proporcionar el servicio.

En el servicio de directorio, hemos visto los dos estándares siguientes:

- **X.500**: define el formato y un protocolo de acceso (un conjunto de operaciones) a la información. El protocolo definido en el X.500 se suele utilizar en aplicaciones OSI.

- **LDAP**: define un protocolo para acceder a los datos definidos en el X.500 utilizando mecanismos de comunicación típicos de Internet. Las operaciones definidas son un subconjunto de las definidas en el X.500.

9) Protocolos Internet en tiempo real: hay un conjunto nuevo de protocolos cuya finalidad consiste en la transmisión de datos en tiempo real y su gestión.

En este módulo se han visto los cuatro protocolos siguientes:

- **RTP**: transmisión de datos en tiempo real.
- **RTCP**: control de la transmisión de datos en tiempo real.
- **RTSP**: transmisión de flujo de datos multimedia.
- **RSVP**: reserva de recursos para la transmisión de datos en tiempo real.

10) Seguridad en las comunicaciones: hemos expuesto las ideas básicas sobre la seguridad, tanto por lo que respecta a las amenazas, como a los servicios y los mecanismos necesarios para proporcionar comunicaciones seguras.

Actividades

1. Si tenéis acceso a un cliente del protocolo DNS (por ejemplo, `nslookup`), utilizadlo para responder a estas preguntas:

a) ¿Cuál es la dirección del ordenador `www.uoc.es`?
b) ¿Qué ordenador u ordenadores tienen por alias `www.uoc.es`, `ftp.uoc.es` y `news.uoc.es`?
c) ¿Cuáles son los servidores DNS del dominio `uoc.es` y del dominio `uoc.edu`?
d) ¿Qué ordenador u ordenadores hacen de servidor del dominio de correo `campus.uoc.es`?
e) ¿Cuál es el ordenador que tiene por dirección `130.206.1.3`?

2. Con algunas implementaciones UNIX del cliente Telnet, se puede utilizar el comando `toggle netdata` para ver los bytes (en hexadecimal) intercambiados en una conexión. Intentad establecer una conexión Telnet aplicando este comando (llamad al programa telnet sin argumentos, haced `toggle netdata` y, a continuación, `open` *`host`*).

Asimismo, hay implementaciones que admiten el comando `toggle options`, que permite ver el diálogo de negociación. Intentad utilizar esta otra opción en una sesión interactiva (en el puerto por defecto 23) y en una conexión a otro puerto (por ejemplo, `echo`).

Utilizad el comando `send ayt` para enviar la función `AYT` (en una conexión al puerto por defecto). ¿Qué responde el servidor?

3. Estableced una conexión con un servidor NNTP (por ejemplo, con `telnet` *`servidor`* `nntp` o, si ello no funciona, con `telnet` *`servidor`* `119`). Efectuad algunas operaciones como listar los grupos que haya (¡atención: en algunos servidores la lista puede tener más de 10.000 líneas!), entrar en un grupo, ver la cabecera de algún artículo, leer alguno entero, obtener ayuda, etc., y observad los códigos numéricos de respuesta que envía el servidor.

Recordad lo que se ha explicado sobre Telnet en el subapartado 5.2.8 de este módulo didáctico.

4. Estableced una conexión con un servidor HTTP (con `telnet` *`servidor`* `http` o `telnet` *`servidor`* `80`) y mirad si podéis averiguar la fecha en que ha sido modificada por última vez la página principal, o alguna otra información referente al servidor (con un mensaje formado por la línea `HEAD / HTTP/1.0`, una cabecera vacía, una línea en blanco y un cuerpo vacío).

Ejercicios de autoevaluación

1. Un servidor de un protocolo (por ejemplo, rlogin) ha recibido una petición de conexión desde la dirección `213.73.40.10` y quiere saber el nombre del ordenador que tiene esta dirección. ¿Cuáles serán los valores de los campos del mensaje que debe enviar a su servidor DNS?

2. Queremos implementar el protocolo Telnet con un terminal que sólo posee las funciones de saltar de línea (dejando el cursor al principio de la línea siguiente) y retroceder un espacio. ¿Cómo debe actuar este terminal cuando reciba un carácter `LF`? ¿Y un carácter `CR`? ¿Cómo simplifica la implementación el requisito del protocolo de enviar siempre un `LF` o un `NUL` después de un `CR`?

3. En el FTP, ¿por qué creéis que es más eficiente el tipo de representación imagen para las transferencias entre ordenadores UNIX?

4. Enviaos un mensaje de correo por medio de una conexión Telnet al puerto 25 del servidor SMTP de la UOC (`tibet.uoc.es`) utilizando los comandos SMTP necesarios. Incluid algún campo de cabecera en el campo de datos.

5. Analizad los campos de cabecera del mensaje enviado en el ejercicio anterior y fijaos en qué campos han sido introducidos automáticamente por el sistema.

6. Enviaos, utilizando vuestro programa de correo, un mensaje que incluya palabras acentuadas (tanto en el cuerpo, como en alguna cabecera) y un fichero binario. Una vez recibido, analizad las características MIME de la cabecera y del cuerpo.

7. **a)** Este fragmento corresponde a un documento generado con una determinada versión de un conocido editor de páginas HTML 3.2:

```
<BODY>
<CENTER>
<P><B><FONT SIZE=+4>Pr&oacute;logo</FONT></B></P></CENTER>
```

Con independencia de la DTD utilizada, en estas líneas hay un error sintáctico según el estándar SGML. ¿Cuál es el error sintáctico de las líneas de código presentadas?

b) El mismo editor no sólo genera errores SGML, sino también errores respecto a la DTD HTML 3.2; por ejemplo, el siguiente:

```
<CENTER><P>
<HR WIDTH="100"></P></CENTER>
```

¿Qué error hay en este ejemplo?

8. Dibujad el modelo de un sistema que permita acceder al servicio de directorio con el LDAP y por medio de mensajes de correo SMTP. Explicad qué diferencia básica habrá con el acceso LDAP normal. Especificad también qué información deberían incluir los mensajes y cómo se podrían codificar.

9. Enumerad los pasos que se siguen para acceder a un recurso HTTP protegido.

Solucionario

Ejercicios de autoevaluación

1. Los valores de los campos del mensaje deben ser los siguientes:

- `ID`: un número cualquiera de 16 bits.
- Bit `QR`: consulta (0).
- `OPCODE`: `QUERY` (0).
- Bit `RD`: 1.
- `QDCOUNT`: 1.

Los campos de la cabecera `AA`, `TC`, `RA`, `RCODE`, `ANCOUNT`, `NSCOUNT` y `ARCOUNT` no son aplicables (es preciso ponerlos a 0).

- La sección de pregunta tendrá una entrada con los campos siguientes:

– `QNAME`: el nombre `10.40.73.213.IN-ADDR.ARPA`.
– `QTYPE`: `PTR (12)`.
– `QCLASS`: `IN (1)`.

- Las secciones de respuesta, autoridad e información adicional estarán vacías.
- La representación del campo `QNAME` estaría determinada por la secuencia de bytes (en hexadecimal) siguiente:

```
01 35 03 31 39 36 03 31 34 36 03 31 39 33 07 49 4E 2D 41 44 44 52 04 41 52 50 41 00
```

2. Cuando recibe un carácter `LF` (y el carácter anterior no era `CR`), el terminal puede saltar de línea y avanzar tantos espacios como sea preciso para llegar a la posición horizontal anterior.

Cuando recibe un carácter `CR`, debe esperar el carácter siguiente para decidir qué debe hacer. Si es un `LF`, hará un salto de línea normal y, si es `NUL`, puede simular el efecto del retorno de carro retrocediendo tantos espacios como sea preciso.

La especificación del protocolo permite ignorar el carácter que sigue a un `CR` cuando no es `LF` (una implementación simplificada se podría ahorrar la comparación con `NUL`).

3. Porque, con el tipo de representación ASCII, el emisor debe convertir los finales de línea, que en UNIX se representan con el carácter `LF`, en la secuencia `CR-LF`, y el receptor debe convertir los `CR-LF` de nuevo en `LF`. En cambio, con el tipo de representación imagen, los finales de línea se transmiten tal como se leen; es decir, directamente como caracteres `LF`.

4. Para enviar un mensaje, es preciso utilizar, como mínimo, los comandos SMTP que se especifican a continuación. Si se desea, se pueden incluir también algunos campos de cabecera en los datos.

```
HELO dominio
MAIL FROM: originador@dominio
RCPT TO: receptor@dominio
DATA
CampoCabecera1: Atributo1
...
CampoCabeceran: Atributon

Texto separado de la cabecera
por 1 línea en blanco, y acabado
por una línea con solo un punto.
.
QUIT
```

Algunos de los campos de cabecera que se pueden añadir dentro del campo Fecha son los que indicamos a continuación:

```
Subject: *texto
cc: 1#dirección
bcc: #dirección
```

5. Los campos de cabecera que habrá en el mensaje recibido serán los campos introducidos dentro del campo DATA más aquellos que haya introducido alguno de los sistemas por los que ha pasado el mensaje, que por norma general son los siguientes:

```
From: 1#dirección
To: 1#dirección
Date: fecha-hora
Message-ID: id-msg
Received: [from dominio][by dominio][via atom]
*(with atom)[id id-msg][for addr-spec]
```

6. En el mensaje, se podrán ver los campos de cabecera siguientes, que son los relacionados con MIME:

```
MIME-Version: 1*dígito.1*dígito
Content-Type: tipo / subtipo *(; parámetro)
Content-Transfer-Encoding: 7bit | 8bit | binary |
quoted-printable | base64
Content-ID: id-msg
Content-Description: texto
```

Asimismo, si se ha puesto algún acento en algún campo de cabecera, se podrá observar una codificación con la forma siguiente:

```
=? charset ? B | Q ? texto-codificado ?=
```

7.

a) El error está en <FONT SIZE=+4>: los valores de los atributos deben estar delimitados por los caracteres "'" o "". Sólo se pueden omitir estos delimitadores cuando el valor es un nombre SGML, pero +4 no lo es porque "+" no es un carácter válido en un nombre. La forma correcta sería <FONT SIZE="+4">.

b) El error es que un elemento P no puede contener un elemento HR. Si un analizador encuentra <HR> después de <P>, interpreta que hay una etiqueta </P> implícita (puesto que esta última es omisible). Pero, entonces, la etiqueta </P> que viene después de <HR> es incorrecta puesto que no corresponde a ningún elemento abierto.

8. El modelo del sistema del enunciado debe seguir la filosofía definida por el modelo cliente/servidor, en que para acceder a un servidor, el usuario necesita al cliente correspondiente. El sistema encargado del acceso al LDAP por medio de mensajes de correo debe ser capaz de acceder a su buzón para obtener las peticiones, al LDAP para realizar la búsqueda de información y al receptor SMTP para enviar las respuestas.

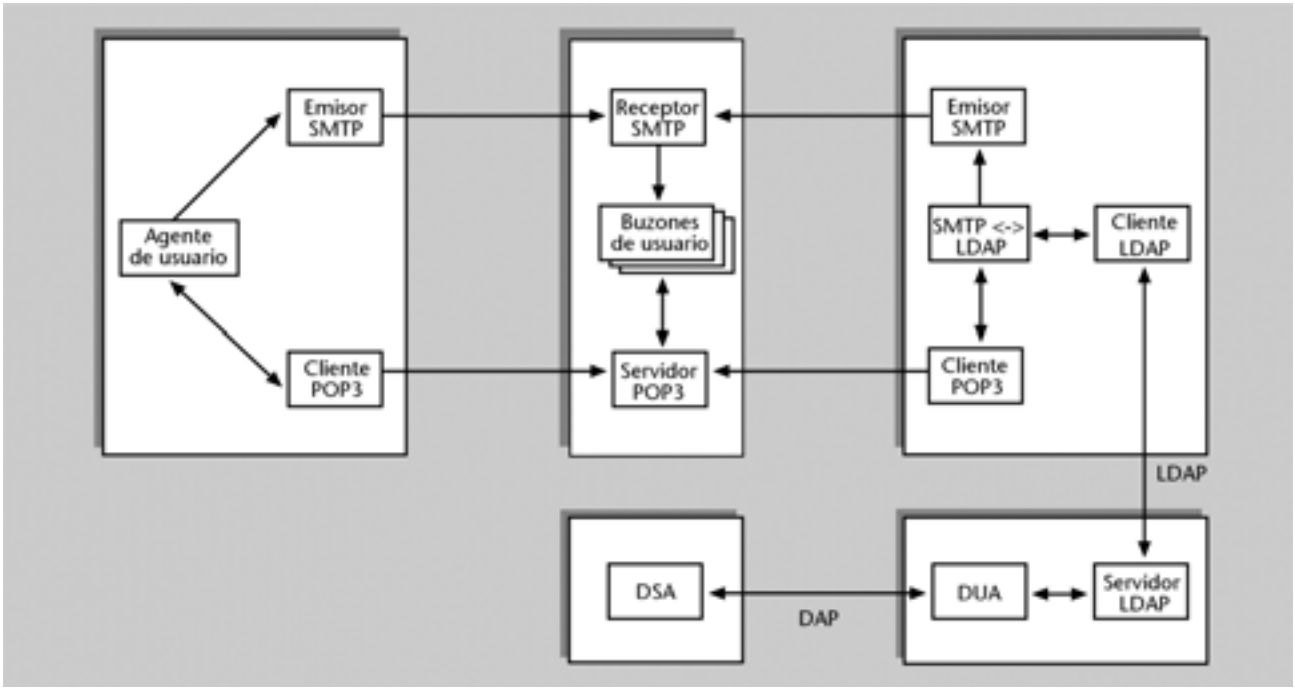
Por este motivo, necesita actuar como usuario de los clientes correspondientes:

- Cliente POP3: da acceso al buzón para recibir peticiones. También, sería válido un cliente IMAP4.
- Cliente LDAP: facilita el acceso al LDAP.
- Emisor SMTP: permite el envío de respuestas.

Por su parte, el usuario debe poder enviar y recibir mensajes, por lo que necesita lo siguiente:

- Emisor SMTP: para enviar peticiones.
- Cliente POP3: da acceso al buzón para recibir las respuestas.

Por estos motivos, en este caso es preciso utilizar los elementos del SMTP (para enviar mensajes), el POP3 (o el IMAP4) (para acceder a los mensajes que se encuentran en un buzón), el LDAP (para acceder al directorio) y el X.500 (el directorio).



La principal diferencia existente entre este método de acceso al X.500 por medio de correo y el acceso normal al LDAP es que este último protocolo se lleva a cabo mediante conexiones directas entre cliente y servidor, con lo que las respuestas se reciben en la misma conexión. En el caso de utilizar el correo para transportar las peticiones y las respuestas, como utiliza un protocolo de almacenamiento y reenvío, el sistema propuesto tendrá los retrasos típicos que implica este protocolo.

Para implementar el sistema utilizando mensajes, es preciso tener en cuenta que toda operación necesita codificar un identificador, un argumento y una respuesta (resultado, rechazos o errores). Por consiguiente, debe poderse incluir esta información dentro del mensaje. Una posible solución sería codificar, en algún campo de cabecera definido por nuestra aplicación, el identificador del tipo de operación (posiblemente también un identificador del "número de serie" de la operación) y, si se trata de una petición, un resultado, un rechazo o un error.

Por lo que respecta a los parámetros, se podrían incluir dentro del cuerpo del mensaje. Se podrían codificar siguiendo la codificación definida en el X.500 y podrían introducir la información dentro del mensaje mediante MIME. Otra opción sería codificar siguiendo la codificación definida en el LDAP, así como utilizar MIME para incluir la información dentro del mensaje. En este caso, en la cabecera sólo sería necesario indicar que se trata de un mensaje para acceder en el LDAP por medio del correo, puesto que los otros campos ya son codificados dentro del formato definido por el LDAP.

9. Una secuencia típica de acceso a un recurso protegido puede ser la siguiente:
a) El cliente envía una petición `GET` al servidor con el URI del recurso.
b) El servidor detecta que el recurso tiene el acceso restringido y envía una respuesta con el código 401 y el campo `WWW-Authenticate`.
c) El cliente pide al usuario que introduzca un nombre y una contraseña para el reino correspondiente al recurso.
d) El cliente construye el campo `Authorization` y envía una nueva petición `GET` del mismo recurso con este campo.
e) El servidor verifica la credencial, y si es válida, envía una respuesta 200 con el contenido del recurso y, si no lo es, vuelve a enviar una respuesta 401.
f) Si el cliente recibe la respuesta 401, pregunta al usuario si desea volver a introducir los datos de autenticación. Ahora bien, si recibe la respuesta 200, puede incluir automáticamente el campo `Authorization` calculado en las peticiones sucesivas.

Glosario

ancla *f* Cada uno de los dos extremos de un hiperenlace. Uno actúa como origen o cola del enlace (la referencia), y el otro, como destino o cabeza (el referenciado).

aplicación distribuida *f* Aplicación (programa) cuyas diferentes partes se encuentran en ordenadores distintos que utilizan las redes de comunicaciones para interaccionar entre sí.

clave *f* Parámetro que se utiliza, en un algoritmo criptográfico, a la hora de obtener los datos cifrados a partir de los datos originales o, en el algoritmo inverso, para volver a obtener los datos originales a partir de los datos cifrados.

definición de tipo de documento *f Veáse* DTD.

dominio *m* Cada uno de los nodos que, en el sistema DNS, forman el árbol del espacio de nombres.

DTD *f* Conjunto de reglas definidas en una aplicación SGML para determinar cómo se interpretan los documentos utilizados con esta última. La DTD debe incluir una declaración de tipo de documento (`DOCTYPE`) en la que se especifiquen los elementos, atributos y entidades que pueden aparecer en los documentos.

hipermedia *m* Generalización del concepto de hipertexto (texto que se puede recorrer siguiendo unos caminos predefinidos no secuenciales) que incluye contenidos de tipo no textual, como imágenes, audio o vídeo.

interfaz de usuario *f* Parte de un programa encargada de la presentación de la información al usuario y de la interacción con el mismo.

método *m* Cada una de las operaciones que, en el HTTP, se pueden aplicar a un recurso.

modelo cliente/servidor *m* Modelo para representar aplicaciones locales y distribuidas en que se define una parte servidora (que es quien proporciona los servicios) a la que accede una parte cliente siguiendo las indicaciones de un usuario humano por medio de una interfaz de usuario, u otra aplicación.

open systems interconnection *f Véase* OSI.

OSI *f* Arquitectura de comunicaciones definida por la ISO (*International Standards Organization*).

protocolo de acceso *m* Protocolo asimétrico que permite a los clientes acceder a los servidores.

protocolo de sistema *m* Protocolo simétrico que permite la interacción entre servidores de un mismo sistema servidor.

proxy *m* Servidor que recibe peticiones de los clientes, las retransmite a un servidor remoto, recibe las respuestas de este último y las devuelve a los clientes, con frecuencia después de aplicarle algún proceso o transformación.

recurso *m* Elemento de información al que se puede acceder por medio de una red de computadores. En el sistema DNS, la accesibilidad a un recurso (ordenador, buzón de correo, etc.) está determinada por su nombre de dominio y, en el protocolo HTTP, se accede a un recurso (documento, proceso generador de datos, etc.) por medio de su dirección, especificada en forma de URI (identificador uniforme de recurso).

request for comments *f Véase* RFC

RFC *f* Nombre genérico que reciben las normas que se suelen utilizar en Internet.

servidor esclavo *m* Servidor que proporciona (en el NNTP) a un grupo de clientes el acceso a las noticias Usenet por medio de otro servidor.

servidor primario *m* Servidor que contiene (en el DNS) la base de datos original de los nombres correspondientes a una determinada zona.

servidor secundario *m* Servidor que contiene (en el DNS) una copia actualizada de la base de datos de los nombres correspondientes a una determinada zona.

sistema servidor *m* Conjunto de servidores que interaccionan para proporcionar un servicio global.

terminal virtual *m* Dispositivo imaginario con un conjunto preestablecido de funciones de control canónicas. Conociendo sólo este conjunto, un servidor puede controlar cualquier tipo de terminal que efectúe una traducción de sus funciones reales en las canónicas, y viceversa.

Bibliografía

Comer, D. (1996). "Principles, protocols & architecture". *Internetworking with TCP/IP* (vol. I): Hertfortshire: Prentice Hall.

Anexos

Anexo 1

Notación

En este anexo se detalla la notación que se utiliza en todo el módulo para describir los campos de los mensajes y los comandos soportados por los diferentes protocolos.

Cadena literal

Las cadenas literales se definirán con letra de espaciado fijo.

```
cadena
```

Variable

Las variables se definirán con letra inclinada de espaciado fijo o letra cursiva.

```
variable
```

Alternativa

Para indicar que en una especificación se debe elegir un elemento de una lista de *n* elementos, se utiliza el carácter "|" para separar las diferentes alternativas.

```
elemento1 | elemento2 | ... | elementon
```

Repetición

Para los elementos que se pueden repetir entre *n* y *m* veces, se utiliza la construcción *n***m* como prefijo. Los valores *n* y *m* se pueden suprimir. En este caso, se toman por defecto los valores *n* = 0 y *m* = 8.

```
n*melemento
n*elemento
*melemento
*elemento
```

Opcional

Para indicar que una cadena o una variable son opcionales, es preciso cerrarlas entre los caracteres "[" y "]".

```
[elemento]
```

Repetición específica

Cuando se quiere indicar que un elemento debe aparecer con exactitud un número entero *n* de veces, debe ponerse el número *n* como prefijo.

```
nelemento
```

Lista

Si se quiere especificar una lista de entre *n* y *m* elementos separados por comas, se utiliza la construcción `n#m` como prefijo. Los valores *n* y *m* se pueden suprimir. En este caso se toman los valores por defecto, que son *n* = 0 y *m* = 8.

```
n#melemento
n#elemento
#melemento
#elemento
```

Agrupamiento

Cuando se quiere que uno o más elementos sean tratados como un único elemento a efectos de los operadores `|`, `*` y `#`, se indica cerrándolos con los caracteres "`(`" y "`)`". Se suele utilizar en especificaciones de repeticiones y listas.

```
(elemento1 elemento2 ... elementon)
```

Caracteres especiales

Algunos caracteres especiales se incluyen entre los caracteres "<" y ">".

```
<CR>: retorno de carro (CR, carriage return)
<LF>: salto de línea (LF, line feed)
<CRLF>: retorno de carro + salto de línea
```

Anexo 2

Códigos de respuesta

Las diferentes RFC definen los códigos de respuesta posibles para cada uno de sus comandos, así como su significado. Estos códigos están formados por tres dígitos y, para facilitar su interpretación, llevan asociado un significado concreto para cada dígito.

Primer dígito

Indica si la operación se ha efectuado o no.

Primer dígito	
1yz	Respuesta preliminar: se ha iniciado la operación y, cuando se haya completado, el servidor enviará una nueva respuesta.
2yz	Operación completada correctamente.
3yz	Respuesta intermedia: la operación se ha aceptado; sin embargo, el cliente debe enviar nuevos comandos con más información para completarla.
4yz	Respuesta negativa temporal: no se puede llevar a cabo la operación, pero se podrá efectuar si el cliente lo reintenta.
5yz	Respuesta negativa definitiva: no se puede efectuar la operación y probablemente tampoco se podrá llevar a cabo si el cliente lo reintenta.

Ejemplos de respuestas negativas

Cuando se quiere leer un fichero con acceso temporalmente bloqueado porque otro proceso está escribiendo datos en el mismo, el servidor envía una respuesta negativa temporal. En cambio, cuando se quiere leer un fichero que no existe, el servidor envía una respuesta negativa definitiva, aunque siempre existe la posibilidad de que mientras tanto otro proceso lo cree.

Segundo dígito

Indica el tipo de respuesta.

Segundo dígito	
x0z	Error de sintaxis, comando no implementado o, en general, respuesta no perteneciente a ninguna de las otras categorías.
x1z	Respuesta informativa.
x2z	Respuesta referente a las conexiones.
x3z	Respuesta referente al proceso de autenticación (nombre de usuario, contraseña, etc.).
x5z	Respuesta referente a la aplicación concreta (por ejemplo, al sistema de ficheros o de correo).

Tercer dígito

El tercer dígito complementa la información del segundo dígito para especificar de qué respuesta concreta se trata.

Anexo 3

El HTML

El HTML es una aplicación SGML que permite la representación y el intercambio de documentos hipermedia. La especificación HTML incluye una DTD que define un lenguaje basado en la sintaxis SGML y una serie de reglas que indican cómo debe interpretarse cada elemento de un documento. Es conveniente remarcar que todos los documentos HTML deben obedecer a la declaración del tipo de documento HTML; sin embargo, no todos los documentos SGML que siguen esta declaración constituyen documentos HTML.

La especificación HTML se originó en el marco del servicio WWW al principio de los años noventa y, desde entonces, ha experimentado un proceso continuado de evolución. La primera versión oficial, el HTML 2.0, se publicó en el documento *RFC 1866* y recogía las características comunes a las implementaciones que había a mediados de 1994. El hito siguiente fue la publicación de la especificación HTML 3.2 por parte del W3C, en que se describe la funcionalidad de uso común a principios de 1996. En la actualidad, el desarrollo continúa y ha dado lugar, entre otras, a las especificaciones siguientes:

- HTML 4.0: publicada al final de 1997.

- CSS (*Cascading Style Sheets*): mecanismo para asociar estilos de presentación a los elementos HTML.

- XML (*Extensible Markup Language*): representación de documentos más flexible que el HTML y que permite incorporar más funcionalidad del lenguaje SGML.

- DOM (*Document Objet Model*): permite acceder y modificar dinámicamente el contenido, la estructura y el estilo de los documentos.

Cambios en las implementaciones HTML

Aparte de tratarse de un desarrollo relativamente reciente, el principal motivo de los cambios que experimenta la especificación HTML es su enorme popularidad.

Existe un ingente número de implementaciones basadas en este lenguaje, y muchas no resisten la tentación de incluir características propias para satisfacer sus requerimientos particulares. Las extensiones se van añadiendo de manera un poco anárquica, hasta que alguna autoridad publica una especificación oficial que incorpora las más aceptadas.

Errores en las implementaciones HTML

Vale la pena comentar que, aunque las diferentes versiones de la especificación HTML siguen el formato SGML para definir elementos y atributos, por motivos históricos muchas implementaciones admiten algunas variantes sintácticamente incorrectas. El hecho es que, por lo que parece, en un inicio existía cierta confusión o ignorancia respecto al lenguaje SGML entre los implementadores de los sistemas WWW y, por ello, los documentos con que trabajaban no siempre eran documentos SGML correctos. Por consiguiente, en el diseño de visualizadores HTML, tradicionalmente, se seguía el criterio general de dar casi cualquier cosa por válida e intentar mostrar lo que se pudiera, por muchos errores que hubiera en el documento.

En la actualidad, la mayoría de las herramientas HTML generan documentos correctos; sin embargo, toleran ciertos tipos de errores a la hora de interpretar documentos externos para mejorar la compatibilidad con los sistemas antiguos. Sin embargo, todavía existen implementaciones que no es que admitan documentos incorrectos, sino que no son capaces de trabajar adecuadamente con documentos correctos.

Las confusiones sintácticas originales, junto con la obsesión de muchos implementadores de añadir funcionalidad no estándar al lenguaje, hacen que el HTML sea probablemente la especificación más transgredida de la historia de Internet.

Convenciones

Las DTD HTML incluyen, aparte de las declaraciones de tipo de documento, una serie de **convenciones sintácticas** que las implementaciones deben seguir. Algunas de dichas convenciones son las siguientes:

1) En el texto del documento, excepto en los elementos que representan texto formateado (`PRE`), una secuencia de dos espacios en blanco, o más, por norma general debe considerarse equivalente a un único espacio, y los saltos de línea también deben considerarse como espacios.

Presentación del texto

A la hora de presentar el texto, lo más normal es que los visualizadores inserten automáticamente un salto de línea después de la última palabra que quepa completa en cada línea de texto.

2) Algunos atributos, que aquí llamaremos ***univalentes***, están declarados en la DTD con un único valor posible que, asimismo, coincide con el nombre del atributo. Pero, en realidad se pueden utilizar dos valores: el valor declarado y el valor por defecto; es decir, el que se aplica cuando no se especifica el atributo (este valor por defecto, que paradójicamente no coincide con el único valor posible, está definido en la declaración del atributo como `#IMPLIED`).

Ejemplo de atributo univalente

El elemento `OPTION` tiene un atributo univalente denominado `SELECTED`:

```
<!ATTLIST OPTION
  SELECTED (SELECTED)  #IMPLIED
  ...>
```

Por tanto, una etiqueta de inicio de este elemento se puede escribir de la manera siguiente para indicar que se aplica el valor por defecto (¡qué no es `SELECTED`!),:

```
<OPTION>
```

Asimismo, se puede escribir de una de estas otras maneras para indicar que se aplica el valor `SELECTED`. La segunda forma, que utiliza la opción de minimización `SHORTTAG`, se emplea habitualmente, puesto que muchas implementaciones antiguas no reconocían la forma sin minimizar:

```
<OPTION SELECTED=SELECTED>
<OPTION SELECTED>
```

3) En los documentos HTML se utiliza la opción de minimización `OMITTAG`; sin embargo, en la mayoría de los elementos las etiquetas de inicio y final son obligatorias. No obstante, es conveniente exceptuar los elementos de contenido declarado vacío (`EMPTY`), que no pueden tener etiqueta de final.

De aquí en adelante, indicaremos explícitamente en qué elementos de contenido no vacío se puede omitir la etiqueta de inicio y/o final.

4) Las DTD proporcionan una serie de declaraciones de entidades que representan ciertos caracteres especiales para facilitar su inclusión en el texto del documento por medio de las referencias correspondientes.

La entidad `gt`...

... es innecesaria, porque el carácter ">" siempre se puede incluir en el texto sin que se confunda con un delimitador de marcado. En algunas implementaciones antiguas, sin embargo, sí que podía haber confusión porque interpretaban de manera incorrecta los valores de los atributos.

La tabla siguiente nos muestra algunos ejemplos de caracteres especiales y las referencias correspondientes:

Referencia	Carácter
`&`	&
`<`	<
`>`	>

También hay declaradas entidades para los caracteres del denominado **alfabeto latín1** que no tienen equivalente en el código ASCII. Los nombres de dichas entidades son fáciles de recordar, puesto que constan de un carácter y un sufijo como `grave`, `acute`, `uml`, `cedil` o `tilde` para indicar, respectivamente, que hay un acento grave, un acento agudo, una diéresis (*umlaut*), una vírgula (*cedilla*) o una tilde. Presentamos algunos de estos caracteres en la tabla siguiente.

El alfabeto latín1...

... está definido en la estándar ISO/IEC 8859-1 e incluye todos los caracteres y signos diacríticos utilizados en la mayoría de los idiomas de la Europa occidental.

Referencia	Carácter	Referencia	Carácter
`&agrave;`	á	`&Iuml;`	Ï
`&Agrave;`	Á	`&ccedil;`	ç
`&eacute;`	é	`&Ccedil;`	Ç
`&Eacute;`	É	`&ntilde;`	ñ
`&iuml;`	ï	`&middot;`	·

Inclusión de caracteres

Tanto si hay una declaración de entidad para un carácter dado como si no, siempre es posible incluirlo en el texto utilizando una referencia a carácter con su código numérico.

No obstante, si debemos crear un documento HTML y utilizamos el alfabeto latín1, podemos insertar directamente cualquier carácter con signos diacríticos, sin necesidad de recurrir a ninguna entidad.

El juego de caracteres de los documentos HTML

El juego de caracteres de los documentos HTML hasta la versión 3.2 era el alfabeto latín1 (en ello se nota el origen geográfico y cultural de la especificación HTML). Este chovinismo europeo occidental (que no es nada, sin embargo, comparado con el norteamericano, que durante tantos años ha impuesto el código ASCII como única norma *de facto* para representar textos) se ha corregido en la versión 4.0, que utiliza el juego de caracteres universal (UCS, *Universal Character Set*, definido en el estándar ISO/IEC 10646 y equivalente a la especificación Unicode 2.0). El UCS incluye caracteres de todo el mundo utilizados en sistemas de escritura muy diferentes.

A la hora de traducir una cadena de caracteres UCS en una secuencia de bytes, es preciso elegir una codificación. Dado que existen millones de caracteres UCS diferentes, se han definido distintos métodos de codificación adaptados a necesidades concretas: los hay que codifican sólo una parte del juego, los hay que codifican el juego completo, los hay que representan cada carácter con un número fijo de bytes y los hay que utilizan un número variable según cada carácter. De este modo, el alfabeto latín1 se puede considerar un caso particular de codificación.

Un visualizador que obtenga un documento HTML por medio del HTTP puede saber qué codificación se ha aplicado consultando el parámetro `charset` del campo `Content-Type` de la cabecera HTTP. De acuerdo con la especificación de este protocolo, en ausencia del parámetro `charset` debe interpretarse que la codificación es ISO-8859-1; es decir, el alfabeto latín1 (del que el código ASCII es un subconjunto).

Existen ciertas características del lenguaje SGML que, a pesar de no ser opcionales, no eran soportadas por las primeras implementaciones. Por ejemplo, la declaración de nuevas entidades aparte de las que proporcionan las DTD, o el uso de otras técnicas de minimización definidas por la opción `SHORTTAG` aparte de la abreviación de atributos (etiquetas vacías, nulas o sin cerrar).

En teoría, un documento HTML podría utilizar estas características; sin embargo, la probabilidad de que un visualizador las entienda correctamente es, todavía hoy, muy baja. Por ello, en la especificación HTML se recomienda simplemente no utilizarlas.

Con el objetivo de facilitar la extensibilidad del lenguaje y la compatibilidad entre implementaciones de diferentes versiones, se recomienda que los visualizadores ignoren las etiquetas de inicio y de final correspondientes a elementos desconocidos y que traten su contenido como si perteneciera al elemento inmediatamente superior. Asimismo, se recomienda que ignoren las especificaciones de atributos desconocidas y que traten las referencias a entidades desconocidas como si fueran datos (caracteres normales).

Elementos estructurales

Todo documento HTML, como documento SGML, debe empezar con la declaración de tipo de documento. En este caso, el tipo de documento tiene por nombre `HTML`.

No es necesario incluir la declaración SGML en los documentos HTML, puesto que se supone que es conocida por todas las implementaciones, dado que forma parte de la especificación HTML. La declaración SGML varía según la versión HTML; sin embargo, en general especifica, entre otras cosas, que la sintaxis concreta utilizada es la de referencia, que se utilizan las opciones de minimización `OMITTAG` y `SHORTTAG` y que los nombres SGML no están limitados a ocho caracteres, sino a una longitud mayor (que depende de la versión). En el HTML 4.0, además, los caracteres "`:`" y "`_`" también pueden formar parte de un nombre.

Según cuál sea la versión HTML, la declaración de tipo de documento puede tener una de las formas siguientes:

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0 Strict//EN">
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN"
     "http://www.w3.org/TR/REC-html40/strict.dtd">
```

HTML 2.0 y HTML 2.0 Strict

La declaración HTML 2.0 Strict es preferible al HTML 2.0 porque representa una definición más estructurada de los documentos, mientras que la segunda se diseñó para facilitar la interoperabilidad con implementaciones antiguas.

En los documentos HTML, el **elemento raíz** (elemento HTML) contiene una **cabecera** (elemento HEAD) y un **cuerpo** (elemento BODY), en este orden. Ambos elementos son obligatorios, aunque tanto en uno como en otro las etiquetas de inicio y de final son omisibles (así como en el elemento HTML).

Documento mínimo HTML

Éste es un ejemplo de documento mínimo HTML (sólo contiene los elementos obligatorios, pero con todas las etiquetas):

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0 Strict//EN">
<HTML>
<HEAD>
<TITLE> Documento mínimo</TITLE>
</HEAD>
<BODY>
</BODY>
</HTML>
```

En la **cabecera del documento** (elemento HEAD), pueden encontrarse los elementos siguientes:

- TITLE: representa el título del documento. Por norma general, los visualizadores HTML muestran esta información en la parte superior del área de presentación del documento.

- STYLE: contiene una lista de reglas de estilo para controlar la presentación del documento. La sintaxis depende del lenguaje de estilos utilizado (por ejemplo, CSS), que debe especificarse en el atributo TYPE. Este elemento se ha introducido en la versión 4.0.

- BASE: un atributo de este elemento, llamado HREF, indica la dirección base que debe utilizarse para interpretar las direcciones relativas que haya en el documento.

- META: este elemento permite especificar metadatos; es decir, propiedades o datos generales sobre el documento que son útiles, por ejemplo, en las búsquedas de información. Un atributo denominado NAME indica el nombre de la propiedad, y otro, denominado CONTENT, su valor. Asimismo, se puede utilizar el atributo HTTP-EQUIV en lugar de NAME si se quiere especificar un campo de la cabecera HTTP para cuando se acceda al documento por medio de este protocolo.

  Los valores de que puede disponer el atributo NAME del elemento META dependen de la aplicación que deba utilizarlos y, por norma general, están recogidos, junto con sus significados, en una lista predefinida denominada *perfil*. A partir de la versión 4.0, se pueden especificar los perfiles utilizados en los elementos META (así como LINK) por medio de un atributo del elemento HEAD llamado PROFILE.

- LINK: sirve para especificar cómo está relacionado el documento con otro recurso WWW. El atributo HREF contiene la dirección de este otro recurso, y el atributo REL indica de qué tipo de relación se trata (por ejemplo, top si el recurso es la raíz de la jerarquía en que se encuentra el documento, contents si en el recurso hay una tabla de contenidos del documento, etc.). Asimismo, se puede utilizar el atributo REV para indicar una "relación inversa"; es decir, del recurso hacia el documento (por ejemplo, made para identificar al autor del documento, etc.).

El elemento TITLE es obligatorio, y sólo puede haber uno. Todos los demás son opcionales y, salvo el elemento BASE, se pueden repetir. El contenido de TITLE y STYLE son caracteres, y los otros elementos tienen contenido declarado vacío (la información asociada está contenida en sus atributos).

Ejemplo de cabecera de un documento

```
<HEAD>
<TITLE>Documento ejemplo</TITLE>
<BASE HREF="http://www.acme.com/principal.html">
<META NAME="keywords" CONTENT="hipermedia, recurso, marcado">
<META NAME="copyright" CONTENT="© 1998 ACME S.A.">
<META HTTP-EQUIV="Expires" CONTENT="Tue, 12 Jun 2003 14:00:00 GMT">
<LINK REL=contents HREF="index.html">
<LINK REV=made HREF="mailto:autor@acme.com">
</HEAD>
```

En el **cuerpo del documento** (elemento BODY) es donde se encuentra el contenido principal.

Marcos

En el HTML 4.0, se ha introducido la posibilidad de sustituir el elemento BODY por otro tipo de elemento denominado FRAMESET. El contenido de un FRAMESET consta de una combinación de elementos FRAME y/u otros elementos FRAMESET y, opcionalmente, de un elemento NOFRAMES. Cuando se utilizan estos elementos, la declaración de tipo de documento debe tener esta forma:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Frameset//EN"
                 "http://www.w3.org/TR/REC-html40/frameset.dtd">
```

El área de presentación de los documentos puede estar dividida horizontal y/o verticalmente, formando una cuadrícula de zonas rectangulares en las que se visualizan diferentes recursos de manera independiente. Cada una de estas zonas se denomina ***marco***. Cualquier marco, a su vez, puede estar subdividido en otros marcos. Cada elemento FRAME representa un marco, y cada elemento FRAMESET es un conjunto de marcos que están contenidos en una misma zona.

En un elemento FRAMESET, el atributo COLS indica los tamaños horizontales (absolutos o relativos) de cada una de las columnas de marcos que contiene, y el atributo ROWS es análogo referido a los tamaños verticales de las filas. En ausencia de los atributos COLS y ROWS, se entiende que en la zona sólo hay una columna y una fila de marcos, respectivamente.

Orden de los encabezamientos

Las DTD no imponen ninguna restricción sobre el orden en que deben aparecer los encabezamientos: un H1 puede ir seguido de un H3, puede haber un H2 sin que haya ningún H1 antes, etc. Por norma general, sin embargo, será preferible utilizarlos en el orden lógico.

El elemento FRAME no tiene contenido, y su atributo SRC indica la dirección del recurso que se visualiza en el marco correspondiente. El elemento NOFRAMES consta de un elemento BODY en el que se encuentra el contenido que debe mostrar el visualizador en caso de que no soporte la presentación de marcos o esté configurado para no presentarlos.

Es habitual, pero no obligatorio, empezar cada parte del cuerpo con un **encabezamiento**. El HTML proporciona seis tipos de elementos para representar los encabezamientos: H1 se puede utilizar, por ejemplo, para los capítulos del documento; H2, para las secciones de cada capítulo; H3, para las subsecciones, y así hasta H6.

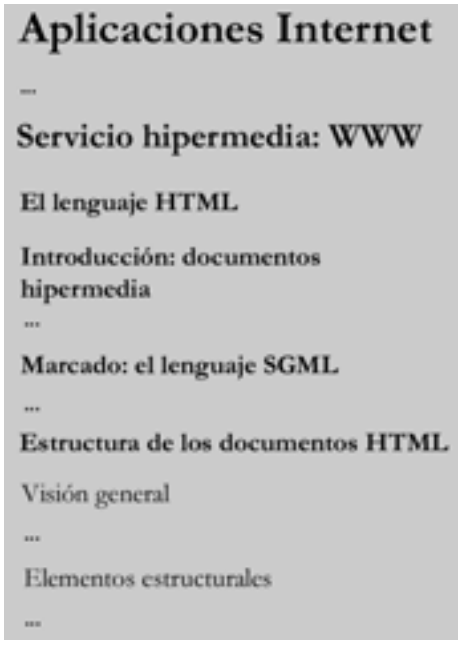
Un visualizador debe mostrar el contenido de cada encabezamiento con un formato más o menos destacado según su nivel, por ejemplo con la letra más grande en los elementos H1 y más pequeña en los H6.

Ejemplo de cuerpo de documento con encabezamientos

Código

```
<BODY>
<H1>Aplicaciones Internet</H1>
...
<H2>Servicio hipermedia: WWW</H2>
<H3>El lenguage HTML</H3>
<H4>Introducción: documentos hipermedia</H4>
...
<H4>Marcado: el lenguage SGML</H4>
...
<H4>Estructura de los documentos HTML</H4>
<H5>Visión general</H5>
...
<H5>Elementos estructurales</H5>
...
</BODY>
```

Resultado



Nuevos atributos en el HTML 4.0

En el HTML 4.0 se han introducido una serie de atributos opcionales comunes a casi todos los elementos del cuerpo (incluido el cuerpo mismo) que son los siguientes:

- `ID` y `CLASS`: sirven para asociar un identificador y una lista de clases a un elemento. Cada identificador de elemento debe ser único en todo el documento; sin embargo, puede haber diferentes elementos pertenecientes a la misma clase. Así, se puede referenciar un elemento sin ambigüedad (por ejemplo, desde otro elemento o durante el procesado del documento) por medio del valor de su atributo `ID`, mientras que, para referenciar un grupo de elementos, se puede utilizar el valor del atributo `CLASS`.

- `TITLE`: permite asociar un título descriptivo a cada elemento.

- `STYLE`: permite especificar información de estilo para presentar un elemento particular. La sintaxis de esta información, como en el caso del elemento `STYLE` de la cabecera, depende del lenguaje de estilos utilizado.

  Según la especificación HTML 4.0, el lenguaje de estilos por defecto se especifica en un campo de la cabecera HTTP (o por medio del elemento `META` correspondiente) llamado `Content-Style-Type`. Si este campo no se encuentra presente, el valor por defecto es `CSS`. Análogamente, el lenguaje de secuencias (*scripts*) por defecto se especifica en el campo `Content-Script-Type`. La especificación del HTTP no define ninguno de estos dos campos; sin embargo, permite la extensión de las cabeceras con campos no estándar.

- Hay un grupo de atributos que indican acciones que debe efectuar el visualizador cuando se producen determinados hechos. La sintaxis depende del lenguaje utilizado para especificar *scripts* o secuencias de acciones (por ejemplo, `JavaScript`).

  La lista de atributos relacionados con las acciones incluye `ONCLICK`, `ONDBLCLICK`, `ONMOUSEDOWN`, `ONMOUSEUP`, `ONMOUSEOVER`, `ONKEYDOWN`, `ONKEYUP`, `ONLOAD` y `ONUNLOAD` (los dos últimos sólo para los elementos `BODY` y `FRAMESET`).

Asimismo, existen dos atributos opcionales introducidos en la versión 4.0 que son comunes a casi todos los elementos del documento, incluyendo `HTML`, `HEAD` y `BODY`:

- `LANG`: indica el idioma utilizado en el contenido o en los atributos del elemento, de acuerdo con la especificación RFC 1766.

- `DIR`: indica en qué dirección debe presentarse el texto de un elemento o los componentes de una tabla. Su valor puede ser `LTR` (de izquierda a derecha) o `RTL` (de derecha a izquierda).

A continuación, veremos otros elementos que pueden aparecer en el cuerpo de un documento, aparte de los encabezamientos:

1) Elementos de párrafo

Los elementos de párrafo, junto con los encabezamientos (H1-H6) son los que por norma general podemos encontrar en el contenido del elemento BODY. Tenemos los elementos siguientes:

- `P`: representa un párrafo. La etiqueta de inicio es obligatoria y la de final es omisible. Su contenido es una combinación de caracteres y elementos de frase, que veremos a continuación.

Elemento vacío

Algunas implementaciones antiguas consideraban `P` como un elemento vacío que representaba un cambio de párrafo y, por consiguiente, no admitían la etiqueta `</P>`.

- `UL`: representa una lista. Su contenido es una secuencia de elementos `LI` que representan los componentes de la lista. Cada componente puede contener caracteres, elementos de frase y la mayoría de los elementos de párrafo.

- OL: representa una lista ordenada. Es como UL, excepto que el visualizador debería mostrar los componentes de la lista numerados automáticamente.

- DL: representa una lista de definiciones. Los componentes de la lista son elementos DT, que representa un término, y DD, que representa su definición. El contenido de DT es como el de P, y el de DD, como el de LI.

En los componentes de las listas (LI, DT y DD), la etiqueta de final es omisible. El contenido de un elemento LI o DD puede incluir, en casos particulares, otras listas (es decir, elementos UL, OL o DL, que entonces representan sublistas).

Antes de la versión 4.0...

... las listas (UL, OL y DL) podían tener un atributo univalente denominado COMPACT. Este atributo indicaba que el visualizador debía mostrar la lista de una manera más compacta. En el HTML 4.0, en cambio, se aconseja el uso de estilos para esta finalidad.

Ejemplo de párrafos y listas

Código

```
<P>En un documento HTML podemos encontrar:
<DL>
<DT>Párrafos.<DD>Suele haber unos cuantos.
<DT>Listas.<DD>Hay listas de tres tipos:
<UL>
<LI>ordenadas
<LI>sin ordenar
<LI>listas de definiciones, formadas por:
<OL>
<LI>un término
<LI>una definición.
</OL>
</UL>
<DT>Otros elementos.<DD>También suele haber otros elementos.
</DL>
```

En las primeras versiones del HTML,...

... había dos elementos más para representar listas (DIR y MENU) que hoy se consideran obsoletos.

Resultado

En un documento HTML podemos encontrar:

Párrafos.
Suele haber unos cuantos.

Listas
Hay listas de tres tipos:
- Ordenadas.
- Sin ordenar.
- Listas de definiciones formadas por:
  1. un término.
  2. una definición.

Otros elementos.
También suele haber otros elementos.

- PRE: representa un párrafo con contenido preformateado. El visualizador debe presentarlo respetando los saltos de línea y las secuencias de espacios que pueda haber. Éste es, pues, el único elemento en que no se aplica la regla general de considerar dos o más espacios como uno solo y los saltos de línea como espacios. Asimismo, debe presentarse con un tipo de letra monoespaciada, es decir, con todos los caracteres igual de anchos. Su contenido es como el del elemento P, excepto que no debería incluir ciertos elementos especiales, como los que cambian el tamaño de la letra.

Ancho de área

En las versiones 2.0 y 3.2, se podía utilizar el atributo WIDTH para especificar el ancho deseado del área de presentación de un elemento PRE, pero la versión 4.0 desaconseja su uso.

- DIV: este elemento, introducido en la versión 3.2, permite estructurar el documento en divisiones, con frecuencia de manera jerárquica. También sirve para agrupar una parte del documento en una división para poder aplicar a toda esta parte un determinado atributo. El contenido del elemento DIV puede incluir caracteres y/o elementos de párrafos y de frase.

- INS y DEL: estos elementos se han introducido en el HTML 4.0. Sirven para representar, respectivamente, una parte del documento que se ha insertado o suprimido respecto a una versión anterior. Pueden incluir un atributo CITE, que es la dirección de otro documento con información relacionada con estas modificaciones, y/o un atributo DATETIME, expresado según el estándar ISO 8601, que indica cuándo se produjo la modificación. Su contenido es como el del elemento DIV. Los elementos INS y DEL también se pueden utilizar como elementos de frase. En este caso no pueden contener ningún elemento de párrafo.

- BLOCKQUOTE: sirve para representar una cita textual. Por norma general, se presenta con los márgenes izquierdo y derecho más anchos que en el resto del documento. En el HTML 4.0, se puede especificar el atributo CITE, que es la dirección de un documento con información sobre el texto original. El contenido de este elemento es como el del elemento BODY.

- ADDRESS: este elemento puede representar, por ejemplo, el nombre y la dirección del autor, que por lo general se ubican al inicio o al final del documento. Su contenido es igual que el del elemento P*.

*** En el HTML 2.0 y 3.2, un elemento ADDRESS también puede contener elementos P; en el HTML 4.0, no.**

- HR: es un elemento vacío que representa una línea horizontal. Se puede utilizar, por ejemplo, para separar párrafos o secciones de un documento.

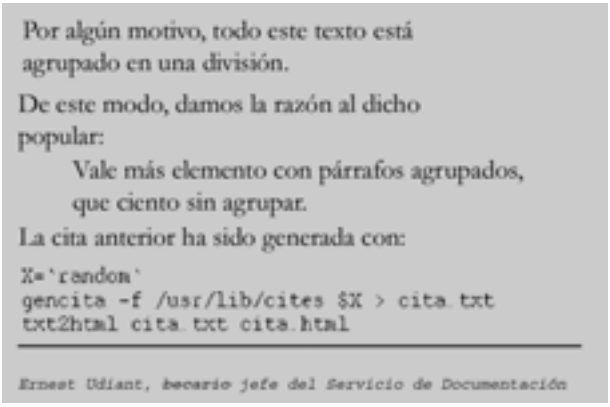
Ejemplo de elementos de párrafo

A continuación, presentamos un ejemplo de diferentes elementos de párrafo (aquí se utilizan INS y DEL, pero como elementos de frase):

Código

```
<DIV><P>Por algún motivo, todo este texto
está agrupado en una división.
<P>De este modo damos la razón al dicho popular:
<BLOCKQUOTE>
<P>Vale más elemento con párrafos agrupados
que ciento sin agrupar.
</BLOCKQUOTE>
<P>La cita anterior ha sido generada con:
<PRE>
X='random'
gencita -f /usr/lib/cites $X > cita.txt
txt2html cita.txt cita.html
</PRE>
</DIV>
<HR>
<ADDRESS>
Ernest Udiant,
<DEL>becario</DEL><INS>jefe</INS> del Servicio de Documentación
</ADDRESS>
```

Resultado



- TABLE: este elemento, que se introdujo en la versión 3.2, se utiliza para representar una tabla; es decir, una matriz de celdas dispuestas en filas y columnas. Puede tener los atributos siguientes:

– WIDTH: es el ancho de la tabla (en píxeles o, si acaba en "%", en porcentaje de la dimensión horizontal del área de presentación).

– BORDER: es el ancho en píxeles del filete en el borde la tabla (por defecto es cero, es decir, sin filete).

– CELLSPACING: indica la separación en píxeles existente entre los filetes de los bordes de celdas adyacentes, y entre los de las celdas de los extremos y el borde de la tabla.

– CELLPADDING: es la distancia entre el filete del borde de cada celda y su contenido.

El contenido del elemento TABLE consta de un elemento CAPTION (cuyo contenido es como el de un elemento P), que es opcional y representa el título de la tabla, seguido de una secuencia de un elemento TR o más, que representan las filas de la tabla.

Un elemento TR puede tener los atributos siguientes:

– ALIGN: indica el tipo de alineación horizontal por defecto de las celdas (puede ser LEFT, CENTER o RIGHT).

– VALIGN: indica la alineación vertical (puede ser TOP, MIDDLE o BOTTOM).

El contenido de este elemento consta de una combinación de elementos TH (celdas cabecera) y/o elementos TD (celdas de datos); un visualizador puede aplicar diferentes estilos de presentación a cada tipo de celda.

Los atributos que se pueden especificar en los elementos TH y TD son ALIGN y VALIGN, que equivalen a los del elemento TR, y ROWSPAN y COLSPAN, que indican cuántas filas y columnas ocupa la celda, respectivamente (por defecto, 1). El contenido de dichos elementos puede ser como el del elemento DIV.

Las etiquetas de final de los elementos TR, TH y TD son omisibles.

Ejemplo de tabla

Código

```
<TABLE BORDER=2>
<CAPTION>Tabla 9.2.1</CAPTION>
<TR><TH ROWSPAN=2>Articulo<TH COLSPAN=2>Cantidad
<TR><TH>Bruta<TH>Neta
<TR><TD>Primero<TD ALIGN=RIGHT>4<TD ALIGN=RIGHT>3
<TR><TD>Segundo<TD ALIGN=RIGHT>75<TD ALIGN=RIGHT>50
</TABLE>
```

Resultado

Tablas en el HTML 4.0

En la versión 4.0 del HTML se han introducido una serie de extensiones al mecanismo de representación de tablas siguiendo la especificación RFC 1942.

Se han añadido los atributos siguientes al elemento TABLE:

- SUMMARY es un resumen de la tabla para los visualizadores que no la pueden mostrar.
- FRAME indica en cuáles de los cuatro lados de la tabla debe haber filetes.
- RULES indica entre qué celdas debe haber filetes.

Después del elemento CAPTION pueden aparecer una serie de elementos COL, de contenido vacío, que sirven para especificar propiedades de las columnas de celdas:

- Alineamiento horizontal y vertical (como en los elementos TR, TH y TD).
- Ancho por defecto de la columna (atributo WIDTH).
- Número de columnas a las que se aplica el elemento COL (atributo SPAN).

Alternativamente, los elementos COL pueden estar agrupados en una serie de elementos denominados COLGROUP, a los que se pueden aplicar los mismos atributos que a los elementos COL. La etiqueta de final de los elementos COLGROUP es omisible.

El conjunto de filas de una tabla puede estar estructurado en filas de la cabecera, filas del cuerpo y filas del pie. Así, el elemento TABLE no contiene directamente una secuencia de elementos TR, sino un elemento THEAD opcional, seguido de un elemento TFOOT también opcional y, por último, uno o más elementos TBODY. El contenido de cada uno de

estos tres elementos nuevos constituye una secuencia de elementos TR. Y, en cualquiera de ellos, se pueden especificar los mismos atributos que en un elemento TR. En los elementos THEAD y TFOOT se puede omitir la etiqueta de final y, en el TBODY, tanto la de inicio como la de final (por tanto, una tabla HTML 3.2 también puede ser formalmente una tabla HTML 4.0 válida).

En la especificación de la alineación horizontal, se han añadido los valores JUSTIFY y CHAR al atributo ALIGN de los elementos TR, TH, TD (así como COL, COLGROUP, THEAD, TBODY y TFOOT). Cuando el atributo ALIGN vale CHAR, un nuevo atributo denominado CHAR indica cuál es el carácter respecto al cual se alinean las celdas, y otro que se llama CHAROFF permite indicar la posición del carácter de alineación. Y se ha añadido el valor BASELINE al atributo VALIGN (para alinear la primera línea de las celdas de una fila).

- FORM: representa un formulario que el usuario puede rellenar con datos y enviar a un servidor HTTP.

- SCRIPT: este elemento se introdujo en la versión 3.2, pero no se ha definido su semántica hasta la versión 4.0. Su contenido es una secuencia de caracteres que representa un *script* o lista de acciones que es preciso ejecutar cuando se visualiza el documento. La sintaxis depende del lenguaje de especificación de *scripts* utilizado. Alternativamente, el *script* puede estar contenido en un recurso externo; en este caso, se ignora el contenido del elemento SCRIPT.

Los elementos SCRIPT se pueden utilizar tanto en los párrafos como en las frases, incluso se pueden encontrar en la cabecera del documento. Si como resultado de la ejecución del *script* se genera una cadena de caracteres, a la hora de interpretar el documento, el visualizador debe sustituir el elemento SCRIPT por la cadena generada.

En un elemento SCRIPT pueden encontrarse los atributos siguientes:

– TYPE (obligatorio): indica en qué lenguaje está escrito el *script*.

– DEFER (univalente): indica que el *script* no genera contenido y, por tanto, no es preciso ejecutarlo antes de interpretar el documento.

– SRC: si el *script* se encuentra en un recurso externo, indica su dirección o URI (en este caso, otro atributo denominado CHARSET indica la codificación de caracteres utilizada en el recurso externo).

Interpretación de scripts con visualizadores antiguos

Un visualizador que no reconozca el elemento SCRIPT intentará interpretar directamente su contenido como texto HTML, siguiendo la regla general de ignorar las etiquetas desconocidas.

Para evitar que esto suceda, se puede tener el *script* en un recurso externo (y entonces incluir algún texto informativo en el contenido del elemento SCRIPT), o bien, si el lenguaje utilizado lo permite, delimitar el *script* con <!-- y --> para que sea tratado como un comentario si se interpreta como texto HTML. Un visualizador que reconozca el elemento SCRIPT, y se percate de que su contenido está declarado como CDATA, no interpretará los símbolos <!-- y --> que encuentre como delimitadores de comentario, sino como caracteres normales.

- NOSCRIPT: un visualizador puede no soportar la ejecución de *scripts*, estar configurado para no ejecutarlos o no entender un lenguaje particular de *scripts*. En este caso, debe mostrar el contenido del elemento NOSCRIPT (si no, debe ignorarlo). Este elemento se ha introducido en la versión 4.0, y su contenido es como el del elemento BODY (excepto que, como es obvio, no debe contener elementos SCRIPT).

2) Elementos de frase

El contenido de los elementos de párrafo de tipo P y similares consta de una combinación de caracteres y/o elementos de frase. Ni los caracteres ni los elementos de frase deberían aparecer directamente en el elemento BODY, deberían hacerlo formando parte de un elemento de párrafo. La versión 3.2 lo permite; sin embargo, las versiones anteriores (2.0) y posteriores (4.0) lo desaconsejan.

A su vez, el contenido de los elementos de frase en general puede ser el mismo que en los de párrafo; es decir, una combinación de caracteres y/u otros elementos de frase. Los elementos posibles de frase que tendremos son los siguientes:

- A: representa un extremo de un hiperenlace.

Observad cómo funcionan los mapas en la parte dedicada a los enlaces en este anexo.

- IMG: es un elemento vacío que representa una imagen que debe visualizarse en la posición actual del documento. Admite los atributos siguientes:

– SRC (obligatorio): es la dirección que el visualizador debe utilizar para obtener el contenido de la imagen.

– ALT (obligatorio a partir de la versión 4.0): es un texto alternativo que el visualizador debe mostrar si no soporta la presentación de imágenes o está configurado para no presentarlas.

– ISMAP (univalente): indica que la imagen es un mapa controlado por el servidor.

Observad cómo funcionan los mapas en la parte dedicada a los enlaces en este anexo.

En la versión 3.2 se añadió al elemento IMG los atributos WIDTH y HEIGHT, que indican los tamaños horizontal y vertical deseados (en píxeles o, en la versión 4.0, en porcentaje), y USEMAP, la dirección de un elemento MAP, que se utiliza en las imágenes que son mapas controlados por el cliente. En el HTML 4.0 se ha añadido un nuevo atributo, LONGDESC, que es la dirección de un recurso que contiene una descripción más detallada de la imagen como complemento al atributo ALT.

ALIGN

Tanto la versión 2.0 como la 3.2 permiten incluir en los elementos IMG un atributo llamado ALIGN, lo que se desaconseja en la versión 4.0.

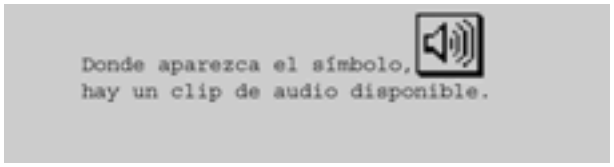
- BR: es un elemento vacío que representa un salto de línea. Por tanto, indica al visualizador que debe dejar de escribir texto en la línea actual y continuar en la siguiente.

Ejemplo de elementos IMG y BR

Código

```
<P>Donde aparezca el símbolo
<IMG SRC="sound.xbm" ALT="[AUDIO]"><BR>
hay un clip de audio disponible.
```

Resultado



Algunos elementos de frase representan partes de un párrafo que deben mostrarse con cierto tipo de letra. Las especificaciones posibles de estos elementos son las siguientes:

– B: representa texto en letra negrita.

– I: representa texto en letra cursiva.

– TT: representa texto en letra mecanográfica (monoespaciada, como la de las máquinas de escribir tradicionales).

– BIG y SMALL: representan texto en letra grande y pequeña, respectivamente.

– SUB y SUP: representan texto que debe mostrarse como subíndice y superíndice, respectivamente.

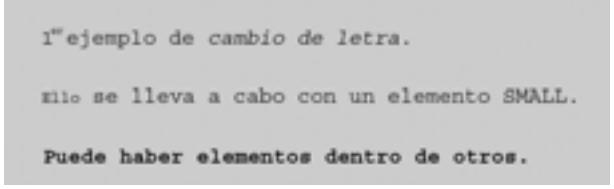
Los elementos BIG, SMALL, SUB y SUP se introdujeron en la versión 3.2.

Ejemplo de cambios de tipo de letra

Código

```
<P>1<SUP>er</SUP> ejemplo de <I>cambio de letra</I>.
<P><SMALL>Ello</SMALL> se lleva a cabo con un elemento
<TT>SMALL</TT>.
<P><B>Puede haber elementos <BIG>dentro de otros</BIG></B>.
```

Resultado



Asimismo, existen elementos que representan partes de un párrafo que deben mostrarse con cierto estilo de presentación, como los siguientes:

– EM: representa un texto enfatizado (que se puede mostrar, por ejemplo, con letra cursiva o subrayada).

– STRONG: representa un texto fuertemente enfatizado (que se puede mostrar, por ejemplo, con letra negrita).

– DFN: su contenido es un término o frase que se quiere destacar para indicar que es en esta parte del texto donde se da su definición (este elemento se introdujo en la versión 3.2).

– SAMP: representa una secuencia de caracteres literales.

– CODE: representa un fragmento de código fuente de un programa.

– KBD: representa un texto que debe teclearse literalmente (por norma general, se utiliza en los manuales de los programas).

– VAR: representa un texto variable o que debe sustituirse por otro (por lo general, se visualiza con letra cursiva).

– CITE: representa algunos tipos de citas; por ejemplo, una referencia al título de una obra (se suele visualizar con letra cursiva).

– ABBR y ACRONYM son dos nuevos elementos introducidos en la versión 4.0 que sirven para indicar que su contenido es una abreviatura y una sigla, respectivamente.

Nota

El contenido de los elementos **SAMP**, **CODE** y **KBD** por lo general se visualiza con letra monoespaciada.

ABBR y ACRONYM...

... no tienen por qué visualizarse con ningún estilo especial; sin embargo, pueden ser útiles en ciertas aplicaciones que procesen el documento (correctores ortográficos, traductores, sintetizadores de voz, etc.). Asimismo, su atributo TITLE (común a casi todos los elementos en HTML 4.0) puede servir para indicar el texto completo que representan.

Ejemplo de cambios de estilo de presentación

Código

```
<P>Un <DFN>estilo</DFN> sirve para controlar el <EM>aspecto</EM>,
como se explica en el libro <CITE>¿Qué es un estilo?</CITE>.
<P>Haced <KBD>^C</KBD> para dejar de leer esto,
o <SAMP>rm -f <VAR>fichero</VAR></SAMP> para
no volverlo a ver <STRONG>nunca más</STRONG>.
```

Resultado

Un **estilo** sirve para controlar el *aspecto*,
como se explica en el libro *¿Qué es un estilo?*

Haced ^C para dejar de leer esto, o rm -f
fichero para no volverlo a ver **nunca más.**

Asimismo, encontramos elementos que son equivalentes a otros de párrafo. Aparte de los elementos INS, DEL y SCRIPT, que se pueden utilizar tanto en párrafos como en frases, en el HTML 4.0 se han introducido nuevos elementos de frase que poseen el mismo significado que ciertos elementos de párrafo (y que admiten los mismos atributos):

– SPAN: permite agrupar una parte del contenido del documento. Es equivalente al elemento DIV, pero en las frases.

– Q: representa una cita. Es equivalente al elemento BLOCKQUOTE, pero en las frases.

– IFRAME: indica que su contenido debe ser mostrado sólo por los visualizadores que no soportan los elementos FRAME; de lo contrario, dicho contenido se obtiene a partir del atributo SRC. Es equivalente al elemento FRAME, pero en las frases.

Otros elementos introducidos en el HTML 4.0

En la versión 4.0 del lenguaje HTML se han introducido los dos elementos siguientes, que se utilizan en las frases:

– BDO: representa un texto que debe mostrarse en el sentido indicado por el atributo DIR (de izquierda a derecha o de derecha a izquierda). Se utiliza cuando el documento contiene textos multilingües en idiomas con sentidos de escritura diferentes.

– OBJECT: representa un objeto y permite incluir en el documento datos de cualquier tipo (un caso particular de las cuales pueden ser las imágenes). Para mostrar un objeto, el visualizador puede necesitar hasta tres clases de información: la implementación del objeto, los datos que lo componen y los valores iniciales de ciertos parámetros.

  La implementación del objeto es, por ejemplo, el código ejecutable necesario para visualizarlo. El atributo CLASSID es un URI del que se puede obtener la implementación y CODETYPE indica de qué tipo es. Otro atributo, ARCHIVE, contiene una lista de URI que también pueden ser necesarios (librerías, etc.). El atributo DATA es otro URI del que se pueden obtener los datos del objeto y el atributo TYPE indica de qué tipo son. Otros atributos que se pueden especificar en un elemento OBJECT son los siguientes: CODEBASE (la dirección base para interpretar los URI anteriores, si son relativos), STANDBY (mensaje que es preciso mostrar mientras se carga el código), HEIGHT y WIDTH.

  Los parámetros del objeto se especifican en elementos PARAM contenidos dentro del elemento OBJECT. Los atributos del elemento PARAM son NAME, VALUE, y VALUETYPE, que indica cómo debe interpretarse el valor: directamente o como una referencia (en el segundo caso, otro atributo denominado TYPE indica de qué tipo es el valor referenciado).

  Además de elementos PARAM, en un elemento OBJECT puede haber contenido "normal" (que puede ser como el de un elemento DIV). Dicho contenido sólo debe presentarse si el visualizador no puede mostrar el objeto; en caso contrario, debe ignorarse. Ello se puede aplicar de manera recursiva: en el contenido de un elemento OBJECT, puede haber otro elemento OBJECT alternativo por si no se puede mostrar el primero, y así sucesivamente.

Enlaces

El mecanismo básico en HTML para representar un hiperenlace o enlace hipermedia (es decir, una referencia a otro recurso) son los elementos A. Por norma general, el visualizador presenta de manera destacada el contenido de un elemento A para que se pueda ver que corresponde a un enlace.

Cada hiperenlace posee dos extremos, en ocasiones denominados **anclas**. Se suele decir que uno es la **cabeza** o destino del enlace y el otro es la **cola** u origen. Si un documento contiene anclas origen, el visualizador debe permitir que el usuario seleccione una, de manera que, una vez la haya seleccionado, se muestre el contenido del recurso al que apunta. Ésta es la base de la navegación hipermedia.

El elemento A sirve para representar un ancla y puede tener los atributos siguientes:

- HREF: es la referencia a la cabeza del enlace y se representa por medio de su dirección. La presencia de este atributo indica que el ancla se puede utilizar como origen de un enlace.

- NAME: es el nombre del ancla cuando se utiliza como cabeza del enlace (no puede haber nombres repetidos en un mismo documento). Este atributo es obligatorio si el atributo HREF no se encuentra presente.

  Cuando el recurso al que apunta un ancla es un documento HTML, por norma general el visualizador lo muestra desde el principio. Si en el documento hay algún ancla con el atributo NAME, es posible especificar su nombre como parte de la dirección, de manera que el visualizador muestre directamente el fragmento del documento en que se encuentra esta ancla.

- REL y REV: permiten expresar una relación directa o inversa con el recurso referenciado (como en el elemento LINK de la cabecera).

Otros atributos añadidos en la versión 4.0 son los siguientes:

En el HTML 2.0,..

... el elemento A podía tener dos atributos más (URN y METHODS) que se suprimieron a partir de la versión 3.2.

– TYPE: indica el tipo de contenido del recurso referenciado.

– CHARSET: informa de la codificación de los caracteres en caso de que sea de tipo texto.

– HREFLANG: indica el idioma del recurso.

– ACCESSKEY: es un carácter que el usuario puede teclear para seleccionar directamente el enlace.

– TABINDEX: sirve para indicar el orden en que se ofrecerán los enlaces cuando el usuario utilice una sola tecla para cambiar la selección.

– SHAPE y COORDS: se utilizan en los mapas controlados por el cliente.

Las versiones 2.0 y 3.2 permiten especificar en los elementos A otro atributo, denominado TITLE, que en la versión 4.0 se ha generalizado a cualquier otro tipo de elemento.

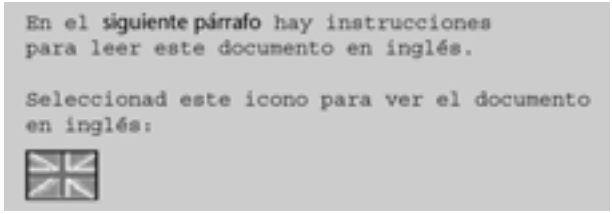
El **contenido de un elemento A**, en general, es como el de cualquier elemento de frase, excepto que no puede contener otros elementos A.

Ejemplo de enlaces

Código

```
<P>En el <A HREF="#ejemplo2">siguiente párrafo</A> hay
instrucciones para leer este documento en inglés.
<P>Seleccionad <A NAME="ejemplo2">este icono</A> para ver
el documento en inglés:
<P><AHREF="ejemplo.html"><IMGSRC="ingles.gif"ALT="[INGLÉS]"></A>
```

Resultado



Si un elemento IMG aparece en el contenido de un ancla, puede incluir el atributo ISMAP para indicar que se trata de un **mapa controlado por el servidor** y que, por tanto, el usuario puede seleccionar un punto cualquiera de la imagen. En este caso, el visualizador añadirá automáticamente un campo denominado ***consulta*** a la dirección del recurso. El valor de este campo consiste en dos números, separados por una coma, que representan las coordenadas *x* e *y* del punto seleccionado.

En este tipo de mapas, el servidor decide qué recurso enviará al cliente según las coordenadas que reciba. Existe otro tipo de mapas, introducidos a partir de la versión 3.2 del HTML, en los que es el cliente quien decide qué recurso solicitará según las coordenadas seleccionadas por el usuario. Este tipo se denomina *mapas controlados por el cliente*, y se representan por medio de los elementos MAP.

Un elemento MAP consta de una lista ordenada de elementos AREA que delimitan áreas dentro del mapa, a cada una de las cuales se puede asociar la dirección de un recurso. Si dos o más áreas se encabalgan, se aplica la que aparezca primero en la lista. El contenido del elemento AREA está vacío, y sus posibles atributos son los que indicamos a continuación:

- SHAPE: indica la forma del área, y su valor puede ser RECT, CIRCLE o POLY (rectangular, circular o poligonal, respectivamente), siendo RECT el valor por defecto.

- COORDS: es una secuencia de números separados por comas, 4 números en las áreas rectangulares (coordenadas *x* e *y* de los vértices superior izquierdo e inferior derecho), 3 números en las circulares (coordenadas del centro y radio) y 2*n* números en las poligonales (coordenadas de cada vértice).

- HREF: es la dirección del recurso al que debe accederse cuando el usuario seleccione un punto del área.

- NOHREF (univalente): indica que es un "área muerta" (no contiene ningún enlace).

- ALT (obligatorio): es una representación alternativa del mapa, en forma de texto, para los visualizadores que no soportan o no tienen configurada su presentación.

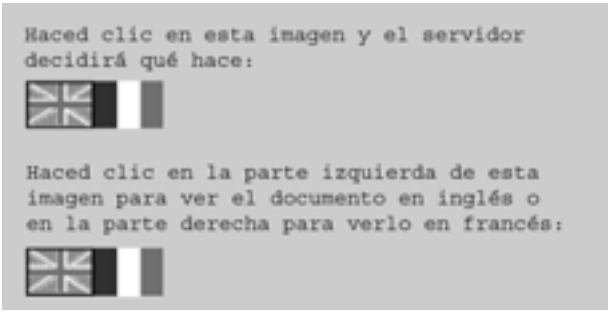
Por su parte, el elemento MAP debe tener un atributo NAME que se puede utilizar de la misma manera que en los elementos A. El elemento IMG que contenga la imagen correspondiente al mapa deberá incluir un atributo USEMAP, cuyo valor será la dirección que identifica el elemento MAP asociado.

Ejemplo de mapa controlado por el servidor y de mapa controlado por el cliente

Código

```
<P>Haced clic en esta imagen y el servidor decidirá qué hace:
<P><A HREF="programa.cgi"><IMG SRC="icono.gif" ALT="[ICONO]"
ISMAP></A>
<P>Haced clic en la parte izquierda de esta imagen para ver el
documento en inglés o en la parte derecha para verlo en francés:
<P><MAP NAME="mapa2">
<AREA SHAPE=RECT COORDS="0,0,50,100"
HREF="something.html" ALT="[INGLÉS]">
<AREA SHAPE=RECT COORDS="50,0,100,100"
HREF="quelquechose.html" ALT="[FRANCÉS]">
</MAP>
<IMG SRC="icona.gif" ALT="[ICONO]" USEMAP="#mapa2">
```

Resultado



Mapas controlados por el cliente en el HTML 4.0

En el HTML 4.0, se han introducido algunas modificaciones en la especificación de los mapas controlados por el cliente, las tres más importantes son las siguientes:

a) El contenido del elemento `MAP` puede ser una secuencia de elementos `AREA`, como en el HTML 3.2, o bien elementos de párrafo (como en el elemento `BODY`) dentro de los que se pueden encontrar elementos `A` con los atributos `SHAPE` y `COORDS` para definir las áreas.

b) El atributo `SHAPE` puede tener otro valor, `DEFAULT`, para indicar el área de la imagen entera.

c) El elemento `MAP` puede incluir atributos `ACCESSKEY` y `TABINDEX` como los de los elementos `A`.

Formularios

Los formularios permiten al usuario introducir datos para que el visualizador los envíe al servidor, por norma general mediante el HTTP. Los datos que se envían constan de un conjunto de campos, cada uno formado por un nombre y un valor.

Los formularios se representan con elementos `FORM`, a los que se les aplican los atributos siguientes:

- `ACTION`: es la dirección del recurso al que deben enviarse los datos (por norma general, será un recurso HTTP).

- METHOD: es el método HTTP que es preciso utilizar para enviar los datos, y puede ser GET (el valor por defecto) o POST. No obstante, como norma general conviene considerar que el método GET debería utilizarse en los formularios que no implican modificaciones en el recurso (por ejemplo, una consulta en una base de datos) y el método POST en los que las implican (por ejemplo, añadir un registro a una base de datos).

- ENCTYPE: es la codificación que se aplica a los datos del formulario para enviarlos al servidor. El atributo ENCTYPE tiene como valor por defecto application/x-www-form-urlencoded.

El contenido de un formulario es como el de un elemento BODY, exceptuando que no puede haber un elemento FORM dentro de otro, y que pueden aparecer en el mismo los elementos específicos de los formularios que indicamos a continuación:

- INPUT: sirve para representar un campo del formulario. Es un elemento vacío con un atributo NAME que indica el nombre del campo y otro atributo TYPE que se utiliza para indicar cómo debe introducir su valor el usuario. Según el atributo TYPE, puede haber otros atributos en el elemento INPUT:

– TYPE=TEXT: el valor del campo se introduce en una línea de texto (éste es el valor por defecto del atributo TYPE). En este caso, el elemento INPUT puede incluir los atributos siguientes: MAXLENGTH es la longitud máxima del valor, SIZE es la longitud de la línea en que el usuario debe introducir el valor (si es menor que MAXLENGTH, pueden ser necesarios desplazamientos horizontales del texto) y VALUE es un valor inicial por defecto.

– TYPE=PASSWORD: este caso es equivalente al caso TYPE=TEXT, exceptuando que el visualizador no debe mostrar el valor del campo cuando se introduzca.

– TYPE=FILE: este caso es como el elemento TYPE=TEXT; sin embargo, el valor introducido por el usuario representa el nombre de un fichero, cuyo contenido se utilizará como valor del campo (este valor del atributo TYPE no estaba definido en la versión 2.0).

– TYPE=CHECKBOX: el visualizador muestra un botón que el usuario puede activar o desactivar. Un atributo VALUE debe especificar el valor del campo que se enviará si el botón está activado (si este último no está activado, no se envía el campo), y un atributo opcional denominado CHECKED (univalente) indica que en un inicio el botón estará activado. En un formulario, puede haber más de un elemento INPUT de este tipo con el mismo nombre.

– TYPE=RADIO: este caso es equivalente al caso TYPE=CHECKBOX, salvo que, de todos los botones con el mismo nombre de campo, es preciso

que haya uno, y sólo uno, que esté activado (si no hay ninguno que incluya el atributo CHECKED, por defecto será el primero que se encuentre en el formulario).

– TYPE=IMAGE: el visualizador muestra una imagen, dada por el atributo SRC (como en los elementos IMG), y el usuario selecciona un punto del mismo. A este elemento INPUT le corresponden dos campos del formulario: uno tendrá el nombre dado por el atributo NAME pero acabado en .x y el otro, acabado en .y, y sus valores serán las coordenadas *x* e *y* del punto seleccionado.

– TYPE=HIDDEN: el visualizador no muestra nada; sin embargo, añade automáticamente al formulario un campo con el nombre y el valor proporcionados por los atributos NAME y VALUE, respectivamente.

– TYPE=SUBMIT: el visualizador muestra un botón que sirve para enviar el formulario. El valor del atributo VALUE se utiliza como etiqueta del botón. En este caso, el atributo NAME es opcional. Si se encuentra presente, el visualizador añadirá el campo correspondiente en el formulario, con el valor dado por el campo VALUE.

– TYPE=RESET: el visualizador muestra un botón que sirve para volver a poner todos los campos del formulario en sus valores iniciales. El valor del atributo VALUE se utiliza como etiqueta del botón. En este caso, no es preciso el atributo NAME.

• SELECT: representa un campo del formulario, cuyo valor debe elegirse en un menú o lista de valores enumerados. Este elemento puede tener los atributos siguientes:

– NAME: indica el nombre del campo.

– SIZE: indica el número de opciones que el visualizador debe mostrar simultáneamente al usuario (si es menor que el número total de opciones, será preciso ir desplazando el menú).

– MULTIPLE (univalente): indica que se puede seleccionar más de una opción.

El contenido del elemento SELECT es una serie de elementos OPTION, cada uno correspondiente a una opción del menú. Los posibles atributos del elemento OPTION son los siguientes:

– SELECTED (univalente): avisa que la opción está inicialmente seleccionada.

– VALUE: indica el valor del campo si se selecciona esta opción.

El contenido del elemento OPTION es una secuencia de caracteres que se muestra al usuario para identificar la opción y que se utiliza como valor del campo en caso de que el atributo VALUE no se encuentre presente. Las etiquetas de final de los elementos OPTION son omisibles.

- TEXTAREA: sirve para representar un campo del formulario cuyo valor es una serie de líneas de texto. El nombre del campo es determinado por el atributo NAME. Existen dos atributos más, ROWS y COLS, que indican cuántas líneas y columnas debe utilizar el visualizador para permitir que el usuario introduzca el valor (si el texto introducido tiene más, puede ser necesario desplazarlo). El contenido de este elemento, que también es una secuencia de caracteres, es el valor inicial que tendrá este campo.

Ejemplo de formulario

Código

```
<FORM ACTION="alta.cgi" METHOD=POST>
<P>Introducid vuestros datos:
<TABLE>  <TR><TD>Nombre:<TD><INPUT  TYPE=TEXT  NAME=nombre  VA-
LUE="anónimo">
<TR><TD>Contraseña:<TD><INPUT TYPE=PASSWORD NAME=contraseña>
</TABLE>
<P>Sexo:
<P><INPUT TYPE=RADIO NAME=sexo VALUE=h> Hombre
<INPUT TYPE=RADIO NAME=sexo VALUE=m> Mujer
<P>Continente:
<SELECT NAME=cont><OPTION VALUE=eur SELECTED> Europa
<OPTION VALUE=afr>África<OPTION VALUE=amer> América
<OPTION VALUE=asia>Asia<OPTION VALUE=ocean> Oceanía
</SELECT>
<P>Servicios a los que queréis acceder:
<P><INPUT TYPE=CHECKBOX CHECKED NAME=serv VALUE=www> WWW
<INPUT TYPE=CHECKBOX NAME=serv VALUE=correo> Correo
<INPUT TYPE=CHECKBOX NAME=serv VALUE=news> News
<P>Aficiones: <TEXTAREA NAME=afic ROWS=3 COLS=20></TEXTAREA>
<P><INPUT TYPE=HIDDEN NAME=código VALUE=ABC>
<INPUT TYPE=SUBMIT VALUE="Enviar datos">
<P>Haced clic <INPUT TYPE=RESET VALUE="aquí"> para borrar los da-
tos y volver a empezar.
</FORM>
```

Resultado

Cuando el usuario selecciona un elemento `INPUT` con `TYPE=SUBMIT` o `TYPE=IMAGE` (o cuando introduce el valor de un elemento `INPUT` con `TYPE=TEXT` y no hay ningún otro campo en el formulario), el visualizador debe enviar los campos del formulario al servidor después de aplicarle la codificación especificada en el atributo `ENCTYPE`.

Cuando este atributo vale `application/x-www-form-urlencoded`, la codificación consiste en efectuar los pasos que presentamos a continuación:

1) En los nombres y valores de cada campo, se sustituye cada espacio en blanco por el carácter "+", y cada carácter especial por la secuencia "`%HH`", donde `HH` son dos dígitos hexadecimales correspondientes al código del carácter (un salto de línea se representa con "`%0D%0A`").

2) En cada campo, se concatena el nombre y el valor separándolos con "=".

3) Se concatenan los campos del formulario, en el orden en que aparecen, separándolos con el símbolo "&".

Ejemplo de codificación de campos de formularios

Ésta sería una posible codificación de los campos del formulario del ejemplo anterior:

```
nombre=Ernest+Udiant&contraseña=Secreto&sexo=h&cont=eur&serv=
www&serv=correo&afic=M%FAsica%0D%0AFutbol&codigo=ABC
```

Una vez codificados los campos, tenemos lo siguiente:

a) Si el valor del atributo `METHOD` es `GET`, el visualizador utiliza la cadena de caracteres resultante como parte de la dirección a la que enviará el formulario. En concreto, esta cadena se especifica como una consulta (lo que significa que se concatena con la dirección separándola con el símbolo "?"). El cuerpo del mensaje HTTP de petición, en este caso, está vacío.

b) Si el valor del atributo `METHOD` es `POST`, los campos codificados se envían en el cuerpo del mensaje HTTP de petición. En la cabecera del mensaje es preciso especificar que el tipo del contenido es el proporcionado por el atributo `ENCTYPE` (`application/x-www-form-urlencoded`).

En ocasiones, puede suceder que un formulario no produzca ninguna modificación en el recurso asociado, pero que la lista de campos sea tan larga que resulte poco conveniente añadirla a la dirección. En este caso, por norma general será más adecuado utilizar el método `POST` que el método `GET`.

Formularios en el HTML 4.0

Como en otros tipos de elemento, la versión 4.0 ha introducido algunos cambios en la especificación de los formularios; los más destacados son los siguientes:

- Un nuevo valor posible del atributo `ENCTYPE` es `multipart/form-data`, más adecuado para transmitir datos de formularios que contengan ficheros arbitrarios.

- El elemento FORM puede tener un nuevo atributo, denominado ACCEPT-CHARSET, que contiene una lista de codificaciones de caracteres que el servidor debe aceptar.
- Un nuevo tipo de elemento INPUT que tiene el atributo TYPE igual a BUTTON.
- Un nuevo elemento denominado BUTTON que equivale al elemento INPUT con TYPE=BUTTON, pero ofrece más funcionalidad.
- Un campo puede incluir los atributos DISABLED o READONLY (univalentes) para indicar que está inhabilitado o que no se puede cambiar su valor inicial.
- Los atributos ALT, USEMAP, ACCESSKEY y TABINDEX se utilizan en los campos de un formulario.
- Las opciones de un elemento SELECT pueden estar agrupadas en elementos OPTGROUP y pueden incluir un atributo LABEL.
- También hay un elemento LABEL para asociar una etiqueta a un campo cualquiera.
- Los diferentes campos se pueden agrupar en elementos FIELDSET, y a cada grupo se le puede asociar un título con un elemento LEGEND.

Elementos y atributos obsoletos

Como hemos visto en los puntos anteriores, existen algunas características del HTML que, a partir de la versión 4.0, se consideran obsoletas y, por tanto, se desaconseja su uso en los documentos. Uno de los motivos principales suele ser que la funcionalidad que ofrecían se puede lograr de manera más flexible con los estilos. En otros casos, se han eliminado características que se implementaban de manera inconsistente o incompatible entre los sistemas más utilizados.

Características añadidas en el HTML 3.2 no recomendadas en el HTML 4.0

En la versión 3.2 de la especificación HTML se introdujo un número importante de nuevos elementos y atributos, no presentes en la versión 2.0, algunos de los cuales se han "retirado" (se recomienda no utilizarlos) en la versión 4.0; entre estos últimos podemos destacar los siguientes:

a) Atributos del elemento BODY: BACKGROUND es la dirección de un recurso que el visualizador debe utilizar como imagen de fondo en el área de presentación, y BGCOLOR, TEXT, LINK, VLINK y ALINK indican con qué color deben mostrarse el fondo, el texto, los enlaces no visitados, los ya visitados y el enlace que el usuario selecciona, respectivamente.

b) El atributo ALIGN en los encabezamientos y en elementos como P, DIV, HR y TABLE se utiliza para especificar la alineación horizontal de las líneas (por la izquierda, por la derecho o centrado).

c) Atributos aplicables a las listas: TYPE (aplicable a UL, OL y LI) indica qué tipo de símbolo debe mostrarse al inicio de los componentes o bien, en las listas ordenadas, el tipo de numeración; START (aplicable a OL) informa del número del primer componente, que por defecto es 1, y VALUE (aplicable a los elementos LI de un elemento OL) indica el número de un componente.

d) El elemento CENTER es equivalente a un elemento DIV con su atributo ALIGN igual a CENTER.

e) Atributos de presentación del elemento HR (además del atributo ALIGN): NOSHADE, SIZE y WIDTH.

f) El atributo ALIGN en el elemento CAPTION de una tabla, que permite indicar si se debe presentar sobre la tabla o bajo la misma.

g) Atributos de los elementos `TH` y `TD`: `NOWRAP` (un atributo univalente que indica que el texto de la celda no debe partirse en líneas), y `WIDTH` y `HEIGHT` (los tamaños horizontal y vertical).

h) El elemento `APPLET`, en la versión 4.0 ha sido sustituido por `OBJECT`, más general.

i) Atributos del elemento `IMG`: `ALIGN` (que ya estaba definido en la versión 2.0), `BORDER`, `HSPACE` y `VSPACE`.

j) Un atributo del elemento `BR` denominado `CLEAR`, que indica al visualizador que avance las líneas necesarias para llegar hasta el final de las imágenes que pueda haber a un lado u otro del texto.

k) Elementos que representan frases con un determinado tipo de letra: `U` (texto subrayado), `STRIKE` (texto rayado), `FONT` (texto con letra del tamaño y color indicados por los atributos correspondientes) y `BASEFONT` (elemento vacío que indica un cambio en el tamaño de la letra).