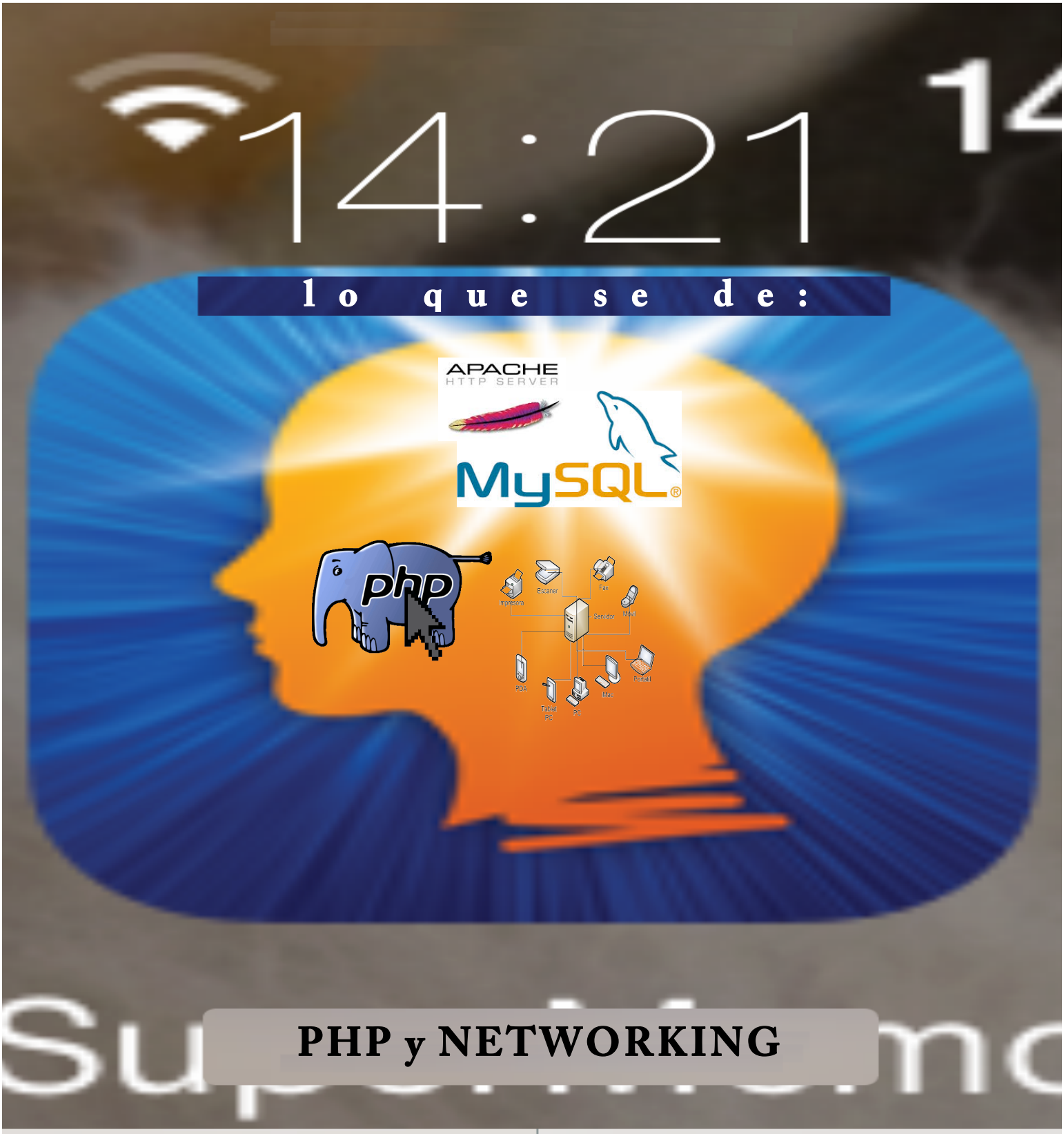


PHP y NETWORKING

ISBN: 978-607-9063-30-6

Coordinadores
Luis Manuel Martínez Hernández
Paula Elvira Ceceñas Torrero
Verónica Clementina Ontiveros Hernández

Autores

Luis Manuel Martínez Hernández
Universidad Juárez del Estado de Durango (UJED)
Instituto de Investigaciones Históricas - UJED
Facultad de Ciencias Exactas – UJED
Facultad de Psicología – UJED
Universidad Pedagógica de Durango
Red Durango de Investigadores Educativos, A. C.

Paula Elvira Ceceñas Torrero
Universidad Pedagógica de Durango

María Elizabeth Leyva Arellano
Facultad de Ciencias Químicas – UJED

Yareli Villalba Segovia
Alumna de la Facultad de Ciencias de la UJED

Yenifer Rivas García
Alumna de la Facultad de Ciencias de la UJED

Coordinadores

Luis Manuel Martínez Hernandez
Paula Elvira Ceceñas Torrero
Verónica Clementina Ontiveros Hernández

Revisión

Karla Campos Martínez

Nombre del libro

Lo que se de: PHP y Networking.

Primera Edición: noviembre de 2014
Editado en México
ISBN: 978-607-9063-30-6

Editor:
Red Durango de Investigadores Educativos, A. C.

Coeditores:
Universidad Juárez del Estado de Durango
Benemérita y Centenaria Escuela Normal del Edo. de Dgo.
Universidad Pedagógica de Durango
Centro de Actualización del Magisterio (Durango)
Instituto Universitario Anglo Español
Instituto de Investigaciones Históricas - UJED
Facultad de Ciencias Exactas – UJED
Facultad de Psicología - UJED
Facultad de Ciencias Químicas - Durango – UJED
Escuela de Lenguas - UJED
Área Básica – UJED

Diseño de portada
Dr. Luis Manuel Martínez Hernandez

Corrección de estilo
Mtra. Paula Elvira Ceceñas Torrero

PROLOGO

La vida actual se ha visto influenciada por el desarrollo de la tecnología, dentro de su desarrollo histórico algunos acontecimientos han sido producto de la capacidad creadora del hombre; como lo es sin duda la computación y la Internet, que han significado un salto gigantesco en la forma de cómo se observa el mundo y como este utiliza a la tecnología para su vida diaria.

En cuanto a la construcción del conocimiento, el desarrollo de modelos de enseñanza y la vida diaria la utilización de herramientas para la creación de nuevos productos computacionales como lo es el software es necesario la utilización de lenguajes de programación adecuados para estos entornos, ya sea en la computadora misma o si esta esta conectada al internet.

Las fronteras cada vez mas se han cancelado y las distancias y en el tiempo, acceso a la información para la toma de decisiones es inmediata, en donde a través del uso de estos nuevos servicios a través del internt se puede tener acceso inmediato a la las fuentes enciclopédicas de conocimiento, que anteriormente eran patrimonio de unos cuantos.

Uno de los grandes desafíos que se nos presentan es generar nuevo conocimiento y herramientas a través del uso de la computadora y el internet, que permita facilitar el acceso al conocimiento y a mejorar los niveles de preparación de los individuos. En este ámbito, a los docentes les corresponde un papel preponderante.

Este tipo de contribuciones se pueden llevar a cabo mediante programas y lenguajes de programación como lo es el lenguaje PHP, el cual es una herramienta de gran ayuda para poder crear nuevos ambientes y programas que se puedan utilizar desde un dispositibvo móvil hasta una gran computadora.

ÍNDICE

CAPÍTULO 1

CAPÍTULO 2

CAPÍTULO 1

Yareli Villalba Segovia
Alumna del Cuarto Semestre de Computación III
de la Facultad de Ciencias Exactas.

INTRODUCCIÓN

En la actualidad nos encontramos con una gran variedad de avances tecnológicos, comenzando por el internet. Pero lo importante no es lo que nos trae este gran avance sino lo que se encuentra detrás de ello. Por ejemplo, al encontrarnos una página de internet llamativa e interesante nos cuestionamos acerca de cómo se hace y comenzamos a imaginarnos sólo un montón de garabatos y numeritos detrás de aquel trabajo sin saber que está a nuestro alcance crear cosas así de interesantes e incluso más importantes de lo que acabamos de encontrar. Entonces comienza el cuestionamiento, "si está a mi alcance, ¿qué debo hacer?", y es donde nuestra curiosidad empieza a desarrollarse, lo que nos lleva a este manual, el cual convierte nuestro deseo de conocimiento al deseo de la creación, con la herramienta básica llamada PHP.

Aquí es donde nos preguntamos ¿qué es php? y ¿cómo lo utilizo?

El propósito de este manual no es otro más que contestar a estas preguntas que todo interesado se hace, definiendo conceptos básicos desde qué es una red, y empleando ejemplos y ejercicios fundamentales para lograr crear nuestra propia página web tan deseada.

Es por eso que, como estudiante, recomiendo el buen uso de este manual, ya que es una herramienta básica para las tecnologías existentes y las que cada día se siguen innovando.

CONCEPTOS BÁSICOS

RED

Conjunto de computadoras, equipos de comunicaciones y otros dispositivos que se pueden comunicar entre sí, a través de un medio en particular.

Parecida a su propia red de contactos, proveedores, partners y clientes, una red informática es simplemente una conexión unificada de sus ordenadores, impresoras, faxes, módems, servidores, y en ocasiones, también sus teléfonos. Las conexiones reales se realizan utilizando un cableado que puede quedar oculto detrás de las mesas de trabajo, bajo el suelo o en el techo. La red informática permite que sus recursos tecnológicos (y, por tanto, sus empleados) "hablen" entre sí; también permitirá conectar su empresa con la Internet y le puede aportar numerosos beneficios adicionales como teleconferencia, actividad multimedia, transferencia de archivos de vídeo y archivos gráficos a gran velocidad, servicios de información, de negocio en línea, etc.

Aplicación de las redes

El reemplazo de una máquina grande por estaciones de trabajo sobre una LAN no ofrece la posibilidad de introducir muchas aplicaciones nuevas, aunque podrían mejorarse la fiabilidad y el rendimiento. Sin embargo, la disponibilidad de una WAN (ya estaba antes) si genera nuevas aplicaciones viables, y algunas de ellas pueden ocasionar importantes efectos en la totalidad de la sociedad. Para dar una idea sobre algunos de los usos importantes de redes de ordenadores, veremos ahora brevemente tres ejemplos: el acceso a programas remotos, el acceso a bases de datos remotas y facilidades de comunicación de valor añadido.

Una compañía que ha producido un modelo que simula la economía mundial puede permitir que sus clientes se conecten usando la red y corran el programa para ver cómo pueden afectar a sus negocios las diferentes proyecciones de inflación, de tasas de interés y de fluctuaciones de tipos de cambio. Con frecuencia se prefiere este planteamiento que vender los derechos del programa, en especial si el modelo se está ajustando constantemente o necesita de una máquina muy grande para correrlo.

Todas estas aplicaciones operan sobre redes por razones económicas; el llamar a un ordenador remoto mediante una red resulta más económico que hacerlo directamente.

La posibilidad de tener un precio más bajo se debe a que el enlace de una llamada telefónica normal utiliza un circuito caro y en exclusiva durante todo el tiempo que dura la llamada, en tanto que el acceso a través de una red, hace que solo se ocupen los enlaces de larga distancia cuando se están transmitiendo los datos.

Una tercera forma que muestra el amplio potencial del uso de redes, es su empleo como medio de comunicación (INTERNET). Como por ejemplo, el tan conocido por todos, correo electrónico (e-mail), que se envía desde una terminal, a cualquier persona situada en cualquier parte del mundo que disfrute de este servicio. Además de textos, se pueden enviar fotografías e imágenes.

Usos de las Redes de Ordenadores

Las redes en general, consisten en "compartir recursos", y uno de su objetivo es hacer que todos los programas, datos y equipo estén disponibles para cualquiera de la red que así lo solicite, sin importar la localización física del recurso y del usuario. En otras palabras, el hecho de que el usuario se encuentre a 1000 km de distancia de los datos, no debe evitar que este los pueda utilizar como si fueran originados localmente.

Un segundo objetivo consiste en proporcionar una alta fiabilidad, al contar con fuentes alternativas de suministro. Por ejemplo todos los archivos podrían duplicarse en dos o tres máquinas, de tal manera que si una de ellas no se encuentra disponible, podría utilizarse una de las otras copias. Además, la presencia de múltiples CPU significa que si una de ellas deja de funcionar, las otras pueden ser capaces de encararse de su trabajo, aunque se tenga un rendimiento global menor.

Otro objetivo es el ahorro económico. Los ordenadores pequeños tienen una mejor relación costo / rendimiento, comparada con la ofrecida por las máquinas grandes. Estas son, a grandes rasgos, diez veces más rápidas que el más rápido de los microprocesadores, pero su costo es miles de veces mayor. Este desequilibrio ha ocasionado que muchos diseñadores de sistemas construyan sistemas constituidos por poderosos ordenadores personales, uno por usuario, con los datos guardados una o más máquinas que funcionan como servidor de archivo compartido.

Este objetivo conduce al concepto de redes con varios ordenadores en el mismo edificio. A este tipo de red se le denomina LAN (red de área local), en contraste con lo extenso de una WAN (red de área extendida), a la que también se conoce como red de gran alcance.

Un punto muy relacionado es la capacidad para aumentar el rendimiento del sistema en forma gradual a medida que crece la carga, simplemente añadiendo más procesadores. Con máquinas grandes, cuando el sistema está lleno, deberá reemplazarse con uno más grande, operación que por lo normal genera un gran gasto y una perturbación inclusive mayor al trabajo de los usuarios.

Otro objetivo del establecimiento de una red de ordenadores, es que puede proporcionar un poderoso medio de comunicación entre personas que se encuentran muy alejadas entre sí. Con el ejemplo de una red es relativamente fácil para dos o más personas que viven en lugares separados, escribir informes juntos. Cuando un autor hace un cambio inmediato, en lugar de esperar varios días para recibirlos por carta.

Esta rapidez hace que la cooperación entre grupos de individuos que se encuentran alejados, y que anteriormente había sido imposible de establecer, pueda realizarse ahora.

A continuación se muestra la clasificación de sistemas multiprocesadores distribuidos de acuerdo con su tamaño físico. En la parte superior se encuentran las máquinas de flujo de datos, que son ordenadores con un alto nivel de paralelismo y muchas unidades funcionales trabajando en el mismo programa. Después vienen los multiprocesadores, que son sistemas que se comunican a través de memoria compartida. En seguida de los multiprocesadores se muestran verdaderas redes, que son ordenadores que se comunican por medio del intercambio de mensajes. Finalmente, a la conexión de dos o más redes se le denomina interconexión de redes.

Tipos De Redes

Los principales tipos de redes son:

Clasificación de las redes según su tamaño y extensión:

- **Redes LAN**. Las redes de área local (Local Área Network) son redes de ordenadores cuya extensión es del orden de entre 10 metros a 1 kilómetro. Son redes pequeñas, habituales en oficinas, colegios y empresas pequeñas, que generalmente usan la tecnología de broadcast, es decir, aquella en que a un sólo

cable se conectan todas las máquinas. Como su tamaño es restringido, el peor tiempo de transmisión de datos es conocido, siendo velocidades de transmisión típicas de LAN las que van de 10 a 100 Mbps (Megabits por segundo).

- Redes MAN. Las redes de área metropolitana (Metropolitan Area Network) son redes de ordenadores de tamaño superior a una LAN, soliendo abarcar el tamaño de una ciudad. Son típicas de empresas y organizaciones que poseen distintas oficinas repartidas en un mismo área metropolitana, por lo que, en su tamaño máximo, comprenden un área de unos 10 kilómetros.

- Redes WAN. Las redes de área amplia (Wide Area Network) tienen un tamaño superior a una MAN, y consisten en una colección de host o de redes LAN conectadas por una subred. Esta subred está formada por una serie de líneas de transmisión interconectadas por medio de routers, aparatos de red encargados de rutear o dirigir los paquetes hacia la LAN o host adecuado, enviándose éstos de un router a otro. Su tamaño puede oscilar entre 100 y 1000 kilómetros.

- Redes internet. Un internet es una red de redes, vinculadas mediante ruteadores gateways. Un gateway o pasarela es un computador especial que puede traducir información entre sistemas con formato de datos diferentes. Su tamaño puede ser desde 10000 kilómetros en adelante, y su ejemplo más claro es Internet, la red de redes mundial.

Redes inalámbricas. Las redes inalámbricas son redes cuyos medios físicos no son cables de cobre de ningún tipo, lo que las diferencia de las redes anteriores. Están basadas en la transmisión de datos mediante ondas de radio, microondas, satélites o infrarrojos.

Clasificación de las redes según la tecnología de transmisión:

- Redes de Broadcast. Aquellas redes en las que la transmisión de datos se realiza por un sólo canal de comunicación, compartido entonces por todas las máquinas de la red. Cualquier paquete de datos enviado por cualquier máquina es recibido por todas las de la red.

Redes Point-To-Point. Aquellas en las que existen muchas conexiones entre parejas individuales de máquinas. Para poder transmitir los paquetes desde una máquina a otra a veces es necesario que éstos pasen por máquinas intermedias, siendo obligado en tales casos un trazado de rutas mediante dispositivos routers.

Clasificación de las redes según el tipo de transferencia de datos que soportan:

- Redes de transmisión simple. Son aquellas redes en las que los datos sólo pueden viajar en un sentido.

- Redes Half-Duplex. Aquellas en las que los datos pueden viajar en ambos sentidos, pero sólo en uno de ellos en un momento dado. Es decir, sólo puede haber transferencia en un sentido a la vez.

- **Redes Full-Duplex.** Aquellas en las que los datos pueden viajar en ambos sentidos a la vez.

Razones para instalar una red

Las redes, entre otras cosas, sirven para:

Compartir recursos y ahorrar dinero.
Aumentar la disponibilidad de la información.

Permitir el acceso a información a una gran cantidad de usuarios (Internet).

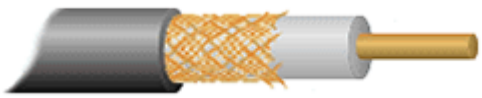
Objetivos principales:

1. La información debe ser entregada de manera confiable y sin daños en los datos.
2. La información debe entregarse de manera consistente.
3. Los equipos que forman la red deben ser capaces de identificarse entre sí.
4. Debe existir una manera estandarizada de nombrar e identificar las partes de la red.

Pese a toda la publicidad y la jerga que rodea a la informática, los beneficios reales que proporciona a la empresa son sencillos. Los ordenadores aportan una mayor velocidad, precisión y fiabilidad al proceso de negocio. Por consiguiente, ahorran tiempo y dinero, y mejoran la calidad de los servicios y productos que se ofrecen a los clientes. Las redes añaden un valor incluso mayor que estas ventajas fundamentales haciendo que los ordenadores (y otras tecnologías) trabajen rápida y fácilmente entre sí. Por ello, para seguir siendo competitivo en el actual entorno de negocio, una red eficaz es una necesidad crítica.

Forma de conexión

Cable Coaxial

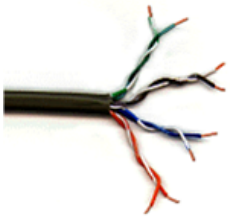


Consiste en un cable conductor interno cilíndrico separado de otro cable conductor externo por anillos aislantes o por un aislante macizo. Esto se recubre por otra capa aislante que es la funda del cable. Este medio físico, es más caro que el par

trenzado, pero se puede utilizar a más larga distancia, con velocidades de transmisión superiores, menos interferencias y permite conectar más estaciones.

Se suele utilizar para televisión, telefonía a larga distancia, LAN, conexión de periféricos a corta distancia, etc. Se utiliza para transmitir señales analógicas o digitales. Sus inconvenientes principales son: atenuación, ruido térmico, ruido de intermodulación. Para señales analógicas, se necesita un amplificador cada pocos kilómetros y para señales digitales un repetidor cada kilómetro.

Par Trenzado



Se trata de dos hilos de cobre aislados y trenzados entre sí, y envueltos por una cubierta protectora. Los hilos están trenzados para reducir las interferencias electromagnéticas con respecto a los pares cercanos que se encuentran a su alrededor (dos pares paralelos constituyen una antena simple, en tanto que un par trenzado no).

Se pueden utilizar tanto para transmisión analógica como digital, y su ancho de banda de ende de la sección de cobre utilizado y de la distancia que tenga que recorrer. Se trata del cableado más económico y la mayoría del cableado telefónico es de este tipo.

Presenta una velocidad de transmisión que depende del tipo de cable de par trenzado que se esté utilizando. Está dividido en categorías por el EIA/TIA:

Categoría 1. Hilo telefónico trenzado de calidad de voz no adecuado para las transmisiones de datos. Velocidad de transmisión inferior a 1 Mbps.

Categoría 2. Cable de par trenzado sin apantallar. Su velocidad de transmisión es de hasta 4 Mbps.

Categoría 3. Velocidad de transmisión de 10 Mbps. Con este tipo de cables se implementa las redes Ethernet 10BaseT.

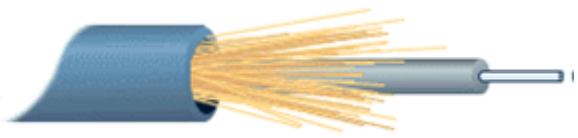
Categoría 4. La velocidad de transmisión llega a 16 Mbps.

Categoría 5. Puede transmitir datos hasta 100 Mbps. Tiene una longitud máxima limitada y, a pesar de los aspectos negativos, es una opción a tener en cuenta debido a que ya se encuentra instalado en muchos edificios como cable telefónico y esto permite utilizarlo sin necesidad de cambiar el cableado. Además, resulta fácil de combinar con otros tipos de cables para la extensión de redes.

Existen dos tipos de pares trenzados, los *apantallados o STP* y los *sin apantallar o UTP*.

Los pares sin apantallar son los más baratos aunque menos resistentes a interferencias. A velocidades de transmisión bajas, los pares apantallados son menos susceptibles a interferencias, aunque son más caros y más difíciles de instalar.

Fibra Óptica



Se trata de un medio muy flexible y muy fino que conduce energía de naturaleza óptica. Su forma es cilíndrica con tres secciones radiales: núcleo, revestimiento y cubierta. El núcleo está formado por una o varias fibras muy finas de cristal o plástico. Cada fibra está rodeada por su propio revestimiento que es un cristal o plástico con diferentes propiedades ópticas distintas a las del núcleo. Alrededor de esto está la cubierta, constituida de material plástico o similar, que se encarga de aislar el contenido de aplastamientos, abrasiones, humedad, etc.

Sus beneficios frente a cables coaxiales y pares trenzados son:

Permite mayor ancho de banda
Menor tamaño y peso
Menor atenuación
Aislamiento electromagnético
Mayor separación entre repetidores.

Su rango de frecuencias es todo el espectro visible y parte del infrarrojo. El método de transmisión es el siguiente: los rayos de luz inciden con una gama de ángulos diferentes posibles en el núcleo del cable, entonces solo una gama de ángulos conseguirán reflejarse en la capa que recubre el núcleo. Son precisamente esos rayos que inciden en un cierto rango de ángulos los que irán rebotando a lo largo del cable hasta llegar a su destino. A este tipo de propagación se le llama *multimodal*. Si se reduce el radio del núcleo, el rango de ángulos disminuye hasta que solo sea posible la transmisión de un rayo, el rayo axial, y a este método de transmisión se le llama *mono modal*.

Los inconvenientes del modo multimodal es que debido a que dependiendo al Ángulo de incidencia de los rayos, estos tomaran caminos diferentes y tardaran más o menos tiempo en llegar al destino, con lo que se puede producir una distorsión (rayos que salen antes pueden llegar después). Debido a esto, se limita la velocidad de transmisión posible. Hay un tercer modo de transmisión que es un

paso intermedio entre los anteriormente comentados y que consiste en cambiar el índice de refracción del núcleo.

A este modo se le llama *multimodo de índice gradual*. Los emisores de luz utilizados son: LED (de bajo costo, con utilización en un amplio rango de temperaturas y con larga vida media) e ILD (más caro, pero más eficaz y permite una mayor velocidad de transmisión).

Protocolos

Los protocolos de comunicación son grupos de reglas que definen los procedimientos, convenciones y métodos utilizados para transmitir datos entre dos o más dispositivos conectados a la red. La definición tiene dos partes importantes:

* Una especificación de las secuencias de mensajes que se han de intercambiar.
* Una especificación del formato de los datos en los mensajes.

La existencia de protocolos posibilita que los componentes software separados pueden desarrollarse independientemente e implementarse en diferentes lenguajes de programación sobre computadores que quizás tengan diferentes representaciones internas de datos.

Un protocolo está implementado por dos módulos software ubicados en el emisor y el receptor. Un proceso transmitirá unos mensajes a otro efectuando una llamada al módulo pasándole el mensaje en cierto formato. Se transmitirá el mensaje a su destino, dividiéndolo en paquetes de tamaño y formato determinado. Una vez recibido el paquete de su módulo realiza transformaciones inversas para regenerar el mensaje antes de dárselo al proceso receptor.

Protocolos Internet

Internet surgió después de dos décadas de investigación y desarrollo de redes de área amplia en los Estados Unidos, comenzando en los primeros años setenta con ARPANET, la primera red de computadoras a gran escala desarrollada. Una parte importante de esa investigación fue el desarrollo del conjunto de protocolos TCP/IP. TCP es el acrónimo de *Transmisión Control Protocol* (protocolo de control de la transmisión), e IP se refiere a *Internet Protocol* (protocolo de Internet).

Servicios de aplicación y protocolos de nivel de aplicación basados en TCP/IP, incluyendo el Web (http), el correo electrónico (SMTP, POP), las redes de noticias (TNP), la transferencia de archivos (FTP), y la conexión remota (TELNET). TCP es un protocolo de transporte; puede ser utilizado para soportar aplicaciones directamente sobre él, o se le puede superponer capas adicionales de protocolos para proporcionar características adicionales (el protocolo Secure Sockerts Layer (SSL) es para conseguir canales seguros sobre los que enviar los mensajes http).

Existen dos protocolos de transporte, TCP (Transport Control Protocol) y UDP (User Datagram Protocol). TCP es un protocolo fiable orientado a conexión, mientras que UDP es un protocolo de datagramas que no garantiza fiabilidad en la transmisión. El protocolo Interred IP (Internet Protocol) es el protocolo de red subyacente de la red virtual Internet; esto es, los datagramas proporcionan un mecanismo de trasmisión básico para Internet y otras redes TCP/IP.

Ethernet proporciona una capa de red física que posibilita que las computadoras conectadas a la misma red intercambien datagramas.

IP se encuentra implementado sobre líneas serie y circuitos telefónicos vía el protocolo PPP, haciendo posible su utilización en las comunicaciones con módem y otros enlaces serie.

El éxito de TCP/IP se basa en su independencia de la tecnología de transmisión subyacente, haciendo posible construir interredes a partir de varias redes y enlaces de datos heterogéneos.

Los usuarios y los programas de aplicación perciben una única red virtual que soporta TCP y UDP, y los constructores de TCP y UDP ven una única red IP virtual, ocultando la diversidad de medios de transmisión.

Redes Inalámbricas

La conexión de los dispositivos portátiles y de mano necesita redes de comunicaciones inalámbricas (wireless networks). Algunos de ellos son la IEEE802.11 (wave lan) son verdaderas redes LAN inalámbricas (wireless local área networks; WLAN) diseñados para ser utilizados en vez de los LAN. También se encuentran las redes de área personal inalámbricas, incluida la red europea mediante el Sistema Global para Comunicaciones Móviles, GSM (global system for mobile communication). En los Estados Unidos, la mayoría de los teléfonos móviles están actualmente basados en la análoga red de radio celular AMPS, sobre la cual se encuentra la red digital de comunicaciones de Paquetes de Datos Digitales Celular, CDPD (Cellular Digital Packet Data).

Dado el restringido ancho de banda disponible y las otras limitaciones de los conjuntos de protocolos llamados Protocolos de Aplicación Inalámbrica WAP (Wireless Aplication Protocol).

Redes de Radio

Las ondas de radio tienen como principales características que son fáciles generar, pueden viajar distancias largas, y penetran edificios fácilmente. Además, son omnidireccionales, lo que significa que ellas viajan en todas las direcciones

desde la fuente, para que el transmisor y receptor no tengan que estar físicamente alineados con cuidado.

Las propiedades de ondas son dependientes de la frecuencia. A frecuencias bajas, atraviesan bien obstáculos, pero el poder baja grandemente cuando se aleja de la fuente. A frecuencias altas, las ondas tienden a viajar en líneas rectas y rebotar cuando consiguen obstáculos. Ellas también son absorbidas por la lluvia. A cualquier frecuencia, las ondas están sujetas a interferencia de los motores y otros equipos eléctricos. El problema principal que se presenta al usar estas bandas para comunicación de datos es el ancho de banda relativamente bajo que ellas ofrecen.

Debido a la habilidad de radio de viajar grandes distancias, la interferencia entre los usuarios es un problema. Por esta razón, todos los gobiernos licencian al usuario de transmisores de radio.

Redes de Microondas

Por encima de los 100 MHz, las ondas viajan en líneas rectas y pueden por consiguiente enfocarse estrechamente. Concentrando toda la energía en una haz pequeño, usando una antena parabólica se obtiene una razón señal a ruido bastante alta, permitiendo la comunicación, pero las antenas transmisoras y receptoras deben alinearse con precisión entre sí. Además, esta direccionalidad permite que múltiples transmisores sean alineados seguidamente para comunicarse con múltiples receptores seguidos sin interferencia.

Puesto que las microondas viajan en una línea recta, si las torres están demasiado separadas, la Tierra estará en el camino (recordar la curvatura del planeta). Por consiguiente, se necesitan repetidoras periódicamente. Mientras más altas sean las torres, más distantes pueden estar. La distancia entre las repetidoras sube muy bruscamente con la raíz cuadrada de la altura de la torre. Para torres con

altura de 100 metros, las repetidoras pueden estar separadas entre sí unos 80 km. Este hecho las hace ser relativamente baratas.

A diferencia de las ondas a bajas frecuencias, las microondas no atraviesan bien los edificios.

Más aun, aunque el haz pueda enfocarse bien al transmisor, hay todavía alguna divergencia en el espacio. Algunas ondas pueden refractarse por capas atmosféricas bajas y pueden tomar ligeramente más tiempo en llegar que las ondas directas. Las ondas retrasadas pueden llegar fuera de fase con la onda directa y por lo tanto cancelar la señal.

La comunicación por microondas se usa ampliamente para la comunicación de teléfono a larga distancia, teléfonos celulares y distribución de la televisión.

Redes Infrarrojos y Ondas Milimétricas

Estos medios de transmisión son ampliamente usados en la comunicación de corto rango, por ejemplo, controles remotos de televisores, VCRs, etc. Son relativamente direccionales, baratos, y fáciles de construir, pero su mayor inconveniente es que no atraviesan objetos sólidos. Por otro lado, el hecho que las ondas infrarrojas no atraviesen paredes sólidas también es una ventaja. Significa que un sistema infrarrojo en un cuarto de un edificio no interferirá con un sistema similar en oficinas adyacentes.

Además, la seguridad de sistemas infrarrojos contra escuchar detrás de las puertas es mejor que el de sistemas de radio precisamente por esta razón. Por esto, ninguna licencia gubernamental se necesita para operar un sistema infrarrojo, en contraste con sistemas de radio que deben ser autorizados.

Estas propiedades han hecho del infrarrojo un candidato interesante para LANs inalámbricas interiores. Por ejemplo, pueden equiparse computadoras y oficinas en un edificio con transmisores y receptores infrarrojos sin necesidad de enfocar.

Redes Satelitales

Un satélite de comunicación puede ser pensado como un repetidor de microondas en el cielo. Contiene diversas transmisiones, cada uno de los cuales escuchan alguna porción del espectro, amplifica la señal entrante, y hace una difusión de vuelta en otra frecuencia para evitar interferencia con la señal que entra. Los rayos que bajan son anchos o angostos, pudiendo cubrir grandes o pequeñas superficies de la tierra, respectivamente.

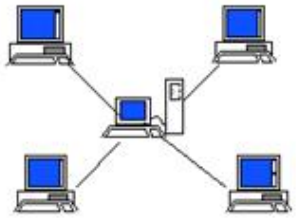
Los enlaces satelitales se diferencian de los enlaces punto a punto terrestres en que los retardos producto de las distancias involucradas son considerables, típicamente 270 ms. Esto es bastante en comparación con los 3 μs/km de los enlaces de microondas y los 5 μs/km del coaxial o la fibra. Otra diferencia es que los satélites son por naturaleza elementos de difusión, lo que es útil en algunos casos, pero en otros, como la seguridad, no lo es. Otras características son que el costo de una transmisión es independiente de la distancia y que tienen una tasa de error bajísima.

Topología De Redes

Cuando hablamos de topología de una red, hablamos de su configuración. Esta configuración recoge tres campos: físico, eléctrico y lógico. El nivel físico y eléctrico se puede entender como la configuración del cableado entre máquinas o dispositivos de control o conmutación. Cuando hablamos de la configuración lógica tenemos que pensar en cómo se trata la información dentro de nuestra red, como se dirige de un sitio a otro o como la recoge cada estación.

Topología en estrella

Todos los elementos de la red se encuentran conectados directamente mediante un enlace punto a punto al nodo central de la red, quien se encarga de gestionar las transmisiones de información por toda la estrella. Evidentemente, todas las tramas de información que circulen por la red deben pasar por el nodo principal, con lo cual un fallo en él provoca la caída de todo el sistema. Por otra parte, un fallo en un determinado cable sólo afecta al nodo asociado a él; si bien esta topología obliga a disponer de un cable propio para cada terminal adicional de la red. La topología de Estrella es una buena elección siempre que se tenga varias unidades dependientes de un procesador, esta es la situación de una típica mainframe, donde el personal requiere estar accesando frecuentemente esta computadora. En este caso, todos los cables están conectados hacia un solo sitio, esto es, un panel central.

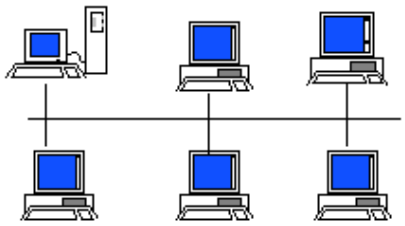


Topología en bus

En esta topología, los elementos que constituyen la red se disponen linealmente, es decir, en serie y conectados por medio de un cable; el bus. Las tramas de información emitidas por un nodo (terminal o servidor) se propagan por todo el bus (en ambas direcciones), alcanzado a todos los demás nodos. Cada nodo de la red se debe encargar de reconocer la información que recorre el bus, para así determinar cuál es la que le corresponde, la destinada a él.

Es el tipo de instalación más sencillo y un fallo en un nodo no provoca la caída del sistema de la red. Por otra parte, una ruptura del bus es difícil de localizar (dependiendo de la longitud del cable y el número de terminales conectados a él) y provoca la inutilidad de todo el sistema.

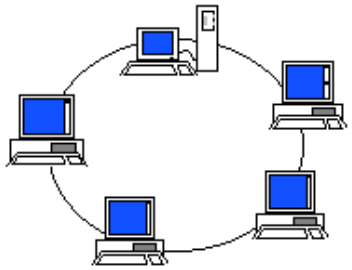
Como ejemplo más conocido de esta topología, encontramos la red *Ethernet* de Xerox. El método de acceso utilizado es el *CSMA/CD*, método que gestiona el acceso al bus por parte de las terminales y que por medio de un algoritmo resuleve los conflictos causados en las colisiones de información. Cuando un nodo desea iniciar una transmisión, debe en primer lugar escuchar el medio para saber si está ocupado, debiendo esperar en caso afirmativo hasta que quede libre. Si se llega a producir una colisión, las estaciones reiniciarán cada una su transmisión, pero transcurrido un tiempo aleatorio distinto para cada estación. Esta es una breve descripción del protocolo de acceso *CSMA/CD*, pues actualmente se encuentran implementadas cantidad de variantes de dicho método con sus respectivas peculiaridades. El bus es la parte básica para la construcción de redes Ethernet y generalmente consiste de algunos segmentos de bus unidos ya sea por razones geográficas, administrativas u otras.



Topología en anillo

Los nodos de la red se disponen en un anillo cerrado, conectados a él mediante enlaces punto a punto. La información describe una trayectoria circular en una única dirección y el nodo principal es quien gestiona conflictos entre nodos al evitar la colisión de tramas de información. En este tipo de topología, un fallo en

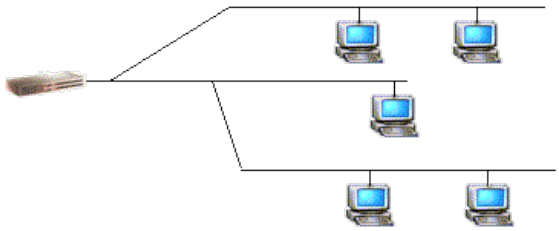
un nodo afecta a toda la red aunque actualmente hay tecnologías que permiten mediante unos conectores especiales, la desconexión del nodo averiado para que el sistema pueda seguir funcionando. La topología de anillo está diseñada como una arquitectura circular, con cada nodo conectado directamente a otros dos nodos. Toda la información de la red pasa a través de cada nodo hasta que es tomado por el nodo apropiado. Este esquema de cableado muestra alguna economía respecto al de estrella. El anillo es fácilmente expandido para conectar más nodos, aunque en este proceso interrumpe la operación de la red mientras se instala el nuevo nodo. Así también, el movimiento físico de un nodo requiere de dos pasos separados: desconectar para remover el nodo y otra vez reinstalar el nodo en su nuevo lugar.



Topología Árbol

En esta topología que es una generalización del tipo bus, el árbol tiene su primer nodo en la raíz y se expande hacia fuera utilizando ramas, en donde se conectan las demás terminales.

Esta topología permite que la red se expanda y al mismo tiempo asegura que nada más exista una ruta de datos entre dos terminales cualesquiera.

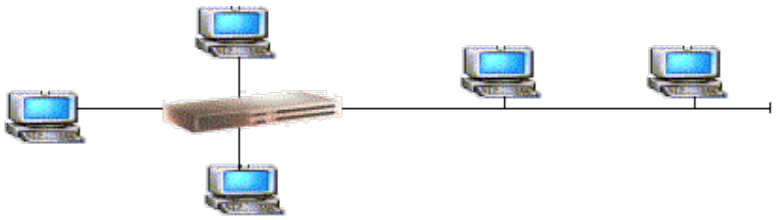


Topología Mesh

Es una combinación de más de una topología, como podría ser un bus combinado con una estrella.

Este tipo de topología es común en lugares en donde tenían una red bus y luego la fueron expandiendo en estrella.

Son complicadas para detectar su conexión por parte del servicio técnico para su reparación.



Qué se necesita para montar una red de ordenadores

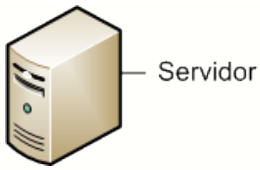
Para montar una red ya sea Ethernet o Inalámbrica, necesitamos unos elementos comunes para su interconexión. Estos elementos son los denominados adaptadores de red, los cuales harán de intérpretes entre las señales electrónicas que circulan por los cables de red, u ondas de radio, y el ordenador. Otro elemento prácticamente imprescindible es el denominado concentrador, que será el encargado de gestionar los paquetes que circulan por los cables, de forma que estos lleguen a su destino. En redes inalámbricas necesitaremos de un dispositivo

adicional denominado Punto de Acceso (Access Point, en inglés), el cual realizará las funciones del concentrador, asignando un canal de radio a cada adaptador de red. Aunque no es imprescindible, su uso es muy recomendado en este tipo de redes. En el caso de querer conectar en red solo dos equipos, este concentrador puede ser sustituido por un cable de red cruzado, llamado Ethernet Crossover. Esta solución podríamos denominarla punto a punto, ya que el cable cruzado conecta directamente a los dos ordenadores. Siempre que se quieran poner en red más de dos ordenadores es necesario el uso de **un concentrador**.

Actualmente existen dos tipos de concentradores, los concentradores convencionales y los conmutados, llamados **Hub** y **Switch** respectivamente. Al igual que es necesario el uso de componentes de hardware para que se comuniquen entre ellos, también es imprescindible el uso de unos componentes de software como lo son, el cliente, los servicios, y el protocolo. De estos componentes el único imprescindible es el protocolo, el cual permitirá a los equipos entenderse entre ellos. Adicionalmente, dependiendo de la topología de red que escojamos, harán falta más o menos componentes, tanto de hardware como de software.

SERVIDOR

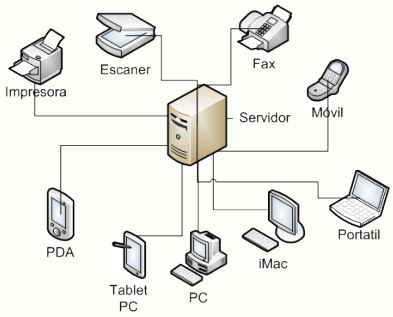
Un servidor, como la misma palabra indica, es un ordenador o máquina informática que está al “servicio” de otras máquinas, ordenadores o personas llamadas clientes y que le suministran a estos, todo tipo de información. A modo de ejemplo, imaginemos que estamos en nuestra casa, y tenemos una despensa. Pues bien a la hora de comer necesitamos unos ingredientes por lo cual vamos a la despensa, los cogemos y nos lo llevamos a la cocina para cocinarlos.



Así en nuestro ejemplo, nuestra máquina servidor sería la despensa, y los clientes somos nosotros como personas que necesitamos unos ingredientes del servidor o despensa. Pues bien con este ejemplo podemos entender ahora un poco mejor qué es un servidor.

Por tanto un servidor en informática será un ordenador u otro tipo de dispositivo que suministra una información requerida por unos clientes (que pueden ser personas, o también pueden ser otros dispositivos como ordenadores, móviles, impresoras, etc.).

Por tanto básicamente tendremos el siguiente esquema general, en el denominado esquema “cliente-servidor” que es uno de los más usados ya que en él se basa gran parte de internet.



Como vemos, tenemos una máquina servidora que se comunica con variados clientes, todos demandando algún tipo de información. Esta información puede ser desde archivos de texto, video, audio, imágenes, emails, aplicaciones, programas, consultas a base de datos, etc.

Por regla general, las máquinas servidoras suelen ser algo más potentes que un ordenador normal. Sobre todo suelen tener más capacidad tanto de almacenamiento de información como de memoria principal, ya que tienen que dar servicio a muchos clientes. Pero como todo, también depende de las necesidades, ya que podemos tener un servidor de menores prestaciones si vamos a tener pocos clientes conectados, o si los servicios que queramos en el servidor no requieren una gran capacidad servidora. A modo de ejemplo, podríamos hacer funcionar un ordenador en nuestra casa como si fuera un servidor, aunque esto no es lo más habitual. Por general, los servidores suelen estar situados en centros de datos de empresas (edificios con grandes salas dedicadas a alojar a los servidores).

TÉRMINOS

Vamos ahora a introducir algunos términos que son muy usados cuando nos referimos a servidores. Estos términos suelen usarse para definir lo que hace un servidor. Por ejemplo, se suele llamar servidor web a aquél cuya actividad principal es enviar páginas web a los usuarios que las solicitan cuando se conectan a internet. Veamos los términos usados habitualmente cuando se habla de servidores:

Proxy.- Es un programa u ordenador que hace de intermediario entre dos ordenadores. Supongamos que nosotros nos identificamos como “juanito” y queremos hacer una petición al servidor llamado “pepito”. Si la petición la hacemos directamente, “pepito” sabe que “juanito” le hizo una petición. En cambio, si usamos un proxy que sería un intermediario que por ejemplo podemos llamar “manolito”, la petición se la haríamos a manolito y éste se la haría a pepito. De esta manera, pepito no sabe que quien realmente ha hecho la petición es juanito. A su vez, el intermediario puede bloquear determinadas peticiones. Por ejemplo, si pedimos a un proxy que tiene bloqueadas las extensiones .xxx, que nos muestre

la página web “amanecer.xxx”, dicha página web no se nos mostrará porque el proxy actúa bloqueándola.

DNS.- Son las siglas de Domain Name System. Es un sistema por el que se asocia una información con un nombre de dominio. El ejemplo más claro es cuando introducimos una ruta url en nuestro navegador de internet del tipo http://www.aprenderaprogramar.com. Una vez hemos introducido esta ruta, dicha información es enviada a un servidor DNS que lo que hace es determinar en qué lugar se encuentra esa página web alojada y nos conecta con ella.

WEB.- El término web va asociado a internet, donde los usuarios utilizan sus navegadores web para visitar sitios web, que básicamente se componen de páginas web donde los usuarios pueden acceder a informaciones con texto, videos, imágenes, etc. y navegan a través de enlaces o hipervínculos a otras webs.

FTP.- Acrónimo de File Transfer Protocol o Protocolo de transferencia de archivos. Es un protocolo utilizado para la transferencia de archivos entre un cliente y un servidor, permitiendo al cliente descargar el archivo desde el servidor o al servidor recibir un archivo enviado desde un cliente. Por defecto FTP no lleva ningún tipo de encriptación permitiendo la máxima velocidad en la transferencia de los archivos, pero puede presentar problemas de seguridad, por lo que muchas veces se utiliza SFTP que permite un servicio de seguridad encriptada.

Dedicación.- Normalmente al ser los servidores equipos más potentes y por tanto más caros, se suelen compartir entre varias personas o empresas, permitiéndoles a todos tener un servicio de gran calidad y a un mínimo precio. En este caso se dice que se trata de un servidor compartido. Pero en otros casos puede haber servidores dedicados exclusivamente a una sola persona o empresa si esta puede hacer frente al gasto económico que supone. En este caso se dice que el servidor es “dedicado”.

POP3 y SMTP.- Hay servidores especializados en correos electrónicos o e-mails. Estos utilizan los protocolos POP3 y SMTP para recibir los correos de nuestro servidor en nuestro cliente, o para enviar desde nuestro cliente un correo al servidor de otro cliente. Aunque hay diversos tipos de protocolos estos son los más utilizados. Un protocolo no es otra cosa que "una forma de hacer algo".

DHCP y TCP/IP.- Cuando un cliente se conecta a un servidor, éste tiene que identificar a cada cliente y lo hace con una dirección IP. Es decir, cuando desde casa entramos en una página web estamos identificados por una serie de dígitos que son nuestra IP. Esta dirección IP son 4 pares de números y es única para cada cliente. Así el protocolo TCP/IP permite que cuando nos conectamos a internet se nos asigne una dirección IP que nos identifica. Cada ordenador conectado a internet tiene su dirección IP, aunque en el caso de usuarios de una empresa que da acceso a internet como "Telefónica", varios usuarios de la empresa pueden tener la misma IP porque utilizan un mismo servidor para canalizar sus peticiones en internet. Por otro lado, DHCP es un protocolo de asignación dinámica de host que permite asignar una IP dinámicamente a cada cliente cuando este se conecta con el servidor que le da acceso a internet. Esto significa que si nos conectamos el lunes a internet, nuestra IP, que nos asigna Telefónica, puede ser 82.78.12.52. En cambio, si nos conectamos el jueves nuestra IP podría ser 212.15.23.88. ¿Por qué cambia nuestra IP? Porque la empresa que nos da conexión nos asigna una de sus IPs disponibles. En cambio, los servidores al ser máquinas más potentes e importantes suelen tener una IP fija.

Una vez introducido estos conceptos, vamos a ver los tipos de servidores, que básicamente se basan en el uso de estos términos.

TIPOS DE SERVIDORES

En esta tabla podemos ver los tipos de servidores más habituales.

DENOMINACIÓN DEL SERVIDOR	DESCRIPCIÓN
Servidor de Correo	Es el servidor que almacena, envía, recibe y realiza todas las operaciones relacionadas con el e-mail de sus clientes.
Servidor Proxy	Es el servidor que actúa de intermediario de forma que el servidor que recibe una petición no conoce quién es el cliente que verdaderamente está detrás de esa petición.
Servidor Web	Almacena principalmente documentos HTML (son documentos a modo de archivos con un formato especial para la visualización de páginas web en los navegadores de los clientes), imágenes, videos, texto, presentaciones, y en general todo tipo de información. Además se encarga de enviar estas informaciones a los clientes.
Servidor de Base de Datos	Da servicios de almacenamiento y gestión de bases de datos a sus clientes. Una base de datos es un sistema que nos permite almacenar grandes cantidades de información. Por ejemplo, todos los datos de los clientes de un banco y sus movimientos en las cuentas.
Servidores Clúster	Son servidores especializados en el almacenamiento de la información teniendo grandes capacidades de almacenamiento y permitiendo evitar la pérdida de la información por problemas en otros servidores.
Servidores Dedicados	Como ya expresamos anteriormente, hay servidores compartidos si hay varias personas o empresas usando un mismo servidor, o dedicados que son exclusivos para una sola persona o empresa.
Servidores de imágenes	Recientemente también se han popularizado servidores especializados en imágenes, permitiendo alojar gran cantidad de imágenes sin consumir recursos de nuestro servidor web en almacenamiento o para almacenar fotografías personales, profesionales, etc. Algunos gratuitos pueden ser: www.imageshack.us, www.theimagehosting.com,www.flickr.com de Yahoo, o picasaweb.google.com de Google.

A modo de resumen, un servidor es un ordenador de gran capacidad que atiende las peticiones de cientos o miles de ordenadores a los que envía información u

ofrece un servicio. El mundo de los servidores es muy complejo. No te preocupes si algunos términos no te resultan del todo claros pues hay profesionales que llevan muchos años trabajando con servidores y realmente es difícil conocer la gran variedad de tipos y nomenclatura que se utiliza para referirse a todos ellos.

DIFERENCIA ENTRE COMPUTADORA Y SERVIDOR

La PC (computadora personal) es la que utilizas como terminal en cada puesto de trabajo y el servidor es al que se conectan en la red, ya sea privada o pública (Internet), en él se configuran los permisos: usuario y contraseña, antivirus, cortafuegos etc.

El PC común y corriente

Tiene un sistema operativo para trabajar en Red dentro de una empresa ejemplo: Windows XP Professional, la versión Home no puede trabajar con redes además que no tiene los protocolos instalados necesarios para trabajar en una red corporativa. O de una empresa... en una empresa nunca hallaras un PC con Windows XP HOME o Windows ME o Windows 98..... pueden tener Windows NT, Windows 2000 profesional o Windows XP Professional. La memoria RAM a una PC común siempre tendrá un Procesador Pentium 4, Celeron, Core 2 Duo, AMD SEmpron, AMD Atlhon a 1.7, 2.2, hasta 3.3 Ghz..... La RAM de una PC normal puede ser desde los 256 MB hasta 1 GB de RAM. El disco duro de una PC normal puede ser desde 80 Gb hasta 300 GB IDE y SATA.

EN UN SERVIDOR

El sistema operativo depende de para que sea el servidor, puede ser muchas veces Windows 2000 Server o Windows 2003 Server (no hay Windows XP Server). En un servidor el procesador puede ser un Intel Core duo, o 2 Intel Core duo, es decir es una motherborad especial que soporta dos procesadores (solo los

Windows servidores pueden soportar más procesadores. Incluso hasta 16 procesadores), con velocidades arriba de los 3.4 Ghz, o AMD más avanzados igual arriba de 3.4 Ghz. En Memoria RAM dependiendo de cuantos equipos de la red se conecten al servidor, varia la RAM, que esta empieza desde 1 GB e incluso llega a 16 GB en sistemas muy grandes (Windows 2000 Server y Windows 2003 Server soportan un máximo de 16 y 32 Gb respectivamente). El disco duro de un servidor es arriba de 300 Gb, llegando a 500 GB o hasta 1 TB (terabyte) o cómo se diga xD, por supuesto son discos SATA en caso de no ser tan grandes pero en servidores se manejan más los SCSI.

APACHE

Apache es el servidor web más usado en sistemas Linux. Los servidores web se usan para servir páginas web solicitadas por equipos cliente. Los clientes normalmente solicitan y muestran páginas web mediante el uso de navegadores web como Firefox, Opera o Mozilla.

Los usuarios introducen un Localizador de Recursos Uniforme (Uniform Resource Locator, URL) para señalar a un servidor web por medio de su nombre de Dominio Totalmente Cualificado (Fully Qualified Domain Name, FQDN) y de una ruta al recurso solicitado. Por ejemplo, para ver la página web del sitio web de Ubuntu, un usuario debería introducir únicamente el FQDN. Para solicitar información específica acerca del soporte de pago, un usuario deberá introducir el FQDN seguido de una ruta.

El protocolo más comunmente utilizado para ver páginas Web es el Hyper Text Transfer Protocol (HTTP). Protocolos como el Hyper Text Transfer Protocol sobre Secure Sockets Layer (HTTPS), y File Transfer Protocol (FTP), un protocolo para subir y descargar archivos, también son soportados.

Los servidores web Apache a menudo se usan en combinación con el motor de bases de datos MySQL, el lenguaje de scripting PHP, y otros lenguajes de scripting populares como Python y Perl. Esta configuración se denomina LAMP (Linux, Apache, MySQL y Perl/Python/PHP) y conforma una potente y robusta plataforma para el desarrollo y distribución de aplicaciones basadas en la web.

BASE DE DATOS

Una base de datos es un conjunto de información que se liga de alguna forma y se encuentra catalogada para tener mejor acceso de la misma.

Inicialmente eran llevadas en registros por medio de fichas, y es por esto que existe una amplia variedad de fichas.

Hoy en día es mediante servidores y computadoras que se acumula la información y dichos datos se catalogan en forma ordenada y de fácil acceso por medio de un sistema que está definido como “SGBD” (sistema de gestión de base de datos), un software creado para tal fin.

Son muchas plataformas las creadas para este fin y muchos programas que logran leer y manejar este tipo de información, la cual con el tiempo se ha unificado en formatos compatibles.

¿Para qué sirven las bases de datos?

Las bases de datos son utilizadas en infinidad de circunstancias:

Una base de datos permite acomodar, ordenar y tener libre acceso de la información, sea cual fuere.

- En los hospitales para catalogar medicamentos y a los pacientes
- En la administración para catalogar los diferentes temas a tratar
- En el gobierno para catalogar los temas y obligaciones a resolver
- En la escuela para matricular a los alumnos
- En el comercio para controlar la información.

Es muy utilizada por los administradores, quienes entre sus funciones tienen la de ordenar y catalogar al personal, las mercancías, los gastos, los ingresos, etc.

Por ello se encuentran en la eminente necesidad de catalogar los datos para que al ser buscados puedan ser encontrados en forma adecuada.

En los colegios, las bibliotecas se encuentran dotadas de bases de datos simples y sencillos, que permiten a los alumnos y maestros encontrar la información deseada en forma rápida y precisa.

En el comercio, los comerciantes ingresan en una base de datos las entradas y salidas para que en base a esos datos el contador o administrador, resuelva al momento de responder por las obligaciones.

COMPILADOR

Para traducir las instrucciones de un programa escrito en un lenguaje de alto nivel a instrucciones de un lenguaje máquina, hay que utilizar un programa llamado **compilador**. Así pues, el compilador es un programa que recibe como datos de entrada el código fuente de un programa escrito por un programador, y genera como salida un conjunto de instrucciones escritas en el lenguaje binario de la computadora donde se van a ejecutar.

Figura - Funcionamiento de un compilador.

INTÉRPRETE

Un **intérprete** es un software que recibe un programa en lenguaje de alto nivel, que lo analiza y lo ejecuta. Para analizar el programa completo, va traduciendo sentencias de código y ejecutándolas si están bien, así hasta completar el programa origen.

Los intérpretes informáticos, al contrario que los compiladores, no generan un fichero ejecutable u otro programa equivalente en otro lenguaje, por lo que cada vez que se ejecuta el programa original debe pasar por la fase de análisis. Esto hace de los intérpretes más lentos que los compiladores, donde las fases de análisis y ejecución son independientes, por lo que solo se compila una vez y se ejecuta cuantas veces se quiera.

Comparando su actuación con la de un ser humano, un compilador equivale a un traductor profesional que, a partir de un texto, prepara otro independiente traducido a otra lengua, mientras que un intérprete corresponde al intérprete humano, que traduce de viva voz las palabras que oye, sin dejar constancia por escrito.

Así, mientras un intérprete toma las instrucciones del programa fuente y las traduce y ejecuta a lenguaje máquina una a una, un compilador realiza la traducción completa del programa fuente a codigo máquina, sin ejecutarlo, siendo posteriormente cuando se ejecute el programa una vez compilado.

Ventajas del intérprete frente al compilador:

- El programa se puede ejecutar de inmediato, sin esperar a ser compilado.
- Puede ser interrumpido con facilidad.
- Puede ser rápidamente modificado y ejecutado nuevamente.
- Resultan muy apropiados durante la fase de desarrollo de un programa, ya que la compilación no permite la ejecución paso a paso del programa y con ello impide la edición seguimiento y depuración del programa.

Desventajas del intérprete frente al compilador:

- La ejecución es más lenta, pues cada instrucción debe ser traducida a código máquina tantas veces como sea ejecutada.
- No son adecuados en la fase de explotación del programa ya que el proceso de interpretación se ha de repetir cada vez que se ejecuta el programa, mientras que con la compilación, una vez obtenido el programa en lenguaje máquina, éste puede ser ejecutado sin necesidad de compilarlo de nuevo.

Son lenguajes interpretados:

- PHP
- ASP (hasta la versión 3)
- HTML
- JavaScript
- Haskell
- Prolog

MySQL

MySQL es un sistema de gestión de bases de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones. MySQL AB —desde enero de 2008 una subsidiaria de Sun Microsystems y ésta a su vez de Oracle Corporation desde abril de 2009— desarrolla MySQL como software libre en un esquema de licenciamiento dual.

Por un lado se ofrece bajo la GNU GPL para cualquier uso compatible con esta licencia, pero para aquellas empresas que quieran incorporarlo en productos privativos deben comprar a la empresa una licencia específica que les permita este uso. Está desarrollado en su mayor parte en ANSI C.

Al contrario de proyectos como Apache, donde el software es desarrollado por una comunidad pública y los derechos de autor del código están en poder del autor individual, MySQL es patrocinado por una empresa privada, que posee el copyright de la mayor parte del código. Esto es lo que posibilita el esquema de licenciamiento anteriormente mencionado. Además de la venta de licencias privativas, la compañía ofrece soporte y servicios. Para sus operaciones contratan trabajadores alrededor del mundo que colaboran vía Internet. MySQL AB fue fundado por David Axmark, Allan Larsson y Michael Widenius.

MySQL es usado por muchos sitios web grandes y populares, como Wikipedia, Google (aunque no para búsquedas), Facebook, Twitter, Flickr y Youtube.

PHYTON

Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis muy limpia y que favorezca un código legible.

Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma.

Es administrado por la Python Software Foundation. Posee una licencia de código abierto, denominada Python Software Foundation License,[1]que es compatible con la Licencia pública general de GNU a partir de la versión 2.1.1, e incompatible en ciertas versiones anteriores.

ACTION SCRIPT

Adobe Action Script (fecha de lanzamiento en 1997) es el lenguaje de programación de la plataforma Adobe Flash. Originalmente desarrollado como una forma para que los desarrolladores programen de forma más interactiva. La programación con Action Script permite mucha más eficiencia en las aplicaciones de la plataforma Flash para construir animaciones de todo tipo, desde simples a complejas, ricas en datos e interfases interactivas.

La versión más extendida actualmente es Action Script 3.0, que significó una mejora en el manejo de programación orientada a objetos al ajustarse mejor al estándar ECMA-262 y es utilizada en las últimas versiones de Adobe Flash y Flex y en anteriores versiones de Flex. Desde la versión 2 de Flex viene incluido Action Script 3, el cual mejora su rendimiento en comparación de sus antecesores,

además de incluir nuevas características como el uso de expresiones regulares y nuevas formas de empaquetar las clases.

C

C es un lenguaje de programación creado en 1972 por Dennis M. Ritchieen y los Laboratorios Bell como evolución del anterior lenguaje B, a su vez basado en BCPL.

Al igual que B, es un lenguaje orientado a la implementación de Sistemas Operativos, concretamente Unix. C es apreciado por la eficiencia del código que produce y es el lenguaje de programación más popular para crear software de sistemas, aunque también se utiliza para crear aplicaciones.

Se trata de un lenguaje de tipos de datos estáticos, débilmente tipificado, de medio nivel pero con muchas características de bajo nivel. Dispone de las estructuras típicas de los lenguajes de alto nivel pero, a su vez, dispone de construcciones del lenguaje que permiten un control a muy bajo nivel. Los compiladores suelen ofrecer extensiones al lenguaje que posibilitan mezclar código en ensamblador con código C o acceder directamente a memoria o dispositivos periféricos.

La primera estandarización del lenguaje C fue en ANSI, con el estándar X3.159-1989. El lenguaje que define este estándar fue conocido vulgarmente como ANSI C. Posteriormente, en 1990, fue ratificado como estándar ISO (ISO/IEC 9899:1990). La adopción de este estándar es muy amplia por lo que, si los programas creados lo siguen, el código es portátil entre plataformas y/o arquitecturas.

HTML

HTML, siglas de ***HyperText Markup Language*** («lenguaje de marcas de hipertexto»), hace referencia al lenguaje de marcado para la elaboración de páginas web. Es un estándar que sirve de referencia para la elaboración de páginas web en sus diferentes versiones, define una estructura básica y un código (denominado código HTML) para la definición de contenido de una página web, como texto, imágenes, etc. Es un estándar a cargo de la W3C, organización dedicada a la estandarización de casi todas las tecnologías ligadas a la web, sobre todo en lo referente a su escritura e interpretación. Es el lenguaje con el que se definen las páginas web.

El lenguaje HTML basa su filosofía de desarrollo en la referenciación. Para añadir un elemento externo a la página (imagen, vídeo, *script*, etc.), este no se incrusta directamente en el código de la página, sino que se hace una referencia a la ubicación de dicho elemento mediante texto. De este modo, la página web contiene sólo texto mientras que recae en el navegador web (interpretador del código) la tarea de unir todos los elementos y visualizar la página final. Al ser un estándar, HTML busca ser un lenguaje que permita que cualquier página web escrita en una determinada versión, pueda ser interpretada de la misma forma (estándar) por cualquier navegador web actualizado. Sin embargo, a lo largo de sus diferentes versiones, se han incorporado y suprimido diversas características, con el fin de hacerlo más eficiente y facilitar el desarrollo de páginas web compatibles con distintos navegadores y plataformas (PC de escritorio, portátiles, teléfonos inteligentes, tabletas, etc.). Sin embargo, para interpretar correctamente una nueva versión de HTML, los desarrolladores de navegadores web deben incorporar estos cambios y el usuario debe ser capaz de usar la nueva versión del navegador con los cambios incorporados.

Usualmente los cambios son aplicados mediante parches de actualización automática (Firefox, Chrome) u ofreciendo una nueva versión del navegador con

todos los cambios incorporados, en un sitio web de descarga oficial (Internet Explorer). Un navegador no actualizado no será capaz de interpretar correctamente una página web escrita en una versión de HTML superior a la que pueda interpretar, lo que obliga muchas veces a los desarrolladores a aplicar técnicas y cambios que permitan corregir problemas de visualización e incluso de interpretación de código HTML. Así mismo, las páginas escritas en una versión anterior de HTML deberían ser actualizadas o reescritas, lo que no siempre se cumple. Es por ello que ciertos navegadores aún mantienen la capacidad de interpretar páginas web de versiones HTML anteriores. Por estas razones, aún existen diferencias entre distintos navegadores y versiones al interpretar una misma página web.

PERL

Antes de nada, un poco de filosofía. Al Perl se le conoce a menudo como "lenguaje de scripting", a diferencia de los lenguajes de programación, ya que originariamente se pensó no para grandes trabajos, sino para la automatización de determinados tasks. Con los años, el Perl ha ido siendo más y más potente, utilizándose para proyectos cada día más relevantes, aunque este no era su cometido original.

El Perl ("Processing Estraction Report Language", para quien lo aprecia; "Pathologically Electric Rubbish Lister", para quien le fascina, como sugiere el autor) es un lenguaje dirigido principalmente al tratamiento de stringhe y file de texto. Lo que en Perl resulta absolutamente natural hacer es efectuar búsquedas de secuencias de caracteres dentro de stringhe (pattern matching), sustituciones de substringhe (pattern substitution), operaciones sobre file de textos estructurados en campos y record o bien no estructurados. Por estas razones, el Perl se utiliza con fuerza en la escritura de procesos CGI instalados en un servidor web, o para el desarrollo de procesos de mantenimiento de las actividades de un servidor. Por los mismos motivos, el Perl no resulta, sin embargo, indicado para

desarrollar procesos de puro cálculo científico o programas que necesitan una gran velocidad y precisión de cálculo o elaboraciones numéricas complejas.

Pero el Perl se convierte en una necesidad para quien tenga que manejar un sitio web que no esté compuesto sólo de texto e imagen: con este lenguaje, es posible automatizar algunas operaciones útiles, desde las más simples solicitud de una dirección e-mail, a las más complejas (como registrar datos de usuarios, etc.). ¿Quién no se ha topado alguna vez con un sitio en el que se nos pedía la dirección e-mail para inscribirnos en una mailing-list, o quién no ha visto en algunas páginas algo como "Visitante número nnnnnnn", o a quién no le han pedido que rellenase módulos de inscripción o registro? ¡Creo que todos! Pues bien, por debajo de todo esto, hay líneas y líneas de Perl (¡en la mayor parte de los casos!) que realizan lo que el usuario ha tecleado y lo restituyen al webmaster en formato inteligible o mejor (sobre todo para registros e inscripciones, en los que hay bastantes campos que rellenar: nombre, apellidos, etc.) formateado en diversos campos (de base de datos) o, en el mejor de los casos, haciéndolo todo solo: tomemos por ejemplo a un usuario que quiere tener un espacio web gratis y que tiene que registrarse: se encontrará ante muchas preguntas que responder, y al final tendrá que pulsar la tecla "envía" y pasar el asunto al webmaster, que se ocupará de confirmar la inscripción.

Quien haya hecho esto al menos una vez se habrá dado cuenta de que la confirmación de inscripción realizada llega inmediatamente (¡hablamos de algunos minutos como máximo!): pues bien, aquí no es el webmaster el que trabaja a la velocidad de la luz, sino un complicado script que recibe las informaciones y las elecciones del ususario, asignándole un nombre de usuario y una clave, preparando un directorio en el servidor que pueda recibir los file del usuario, file que, por otra parte, quedará protegido de otros usuarios que pretendiesen modificarlo, etc.; en definitiva, se preocupa de enviarle al usuario un mensaje (que normalmente es un modelo en el que se incluyen sus datos) para advertirlo de que la operación se ha completado con éxito. Y no nos olvidemos de que el mismo

script controla que las elecciones del usuario sean sensatas: por ejemplo, controlando que la dirección e-mail sea válida (en el sentido de que esté en la forma usuario@dominio y no usuario_dominio, por ejemplo).

En definitiva, el Perl es sin duda es una mina de oro para el webmaster que tenga determinadas exigencias, y aprender las bases es útil para poder iniciar a escribir los propios script de automatización (pero no sólo, se pueden escribir también auténticos programas) y sucesivamente implementarlos en el propio servidor, sea o no local.

El intérprete Perl no compila nunca el códice en códice-máquina, como hacen los compiladores C: cada vez que se ejecuta un script en Perl, se interpreta "al vuelo". Hay dos versiones de Perl: el perl4 y su sucesor, el perl5. Éste es, en general, compatible con el perl4, y tiene además features adicionales. En este documento se hará referencia sobre todo al perl5, ya que debería ser la release más utilizada.

JAVA

El lenguaje de programación Java fue originalmente desarrollado por James Gosling de Sun Microsystems (la cual fue adquirida por la compañía Oracle) y publicado en 1995 como un componente fundamental de la plataforma Java de Sun Microsystems. Su sintaxis deriva mucho de C y C++, pero tiene menos facilidades de bajo nivel que cualquiera de ellos. Las aplicaciones de Java son generalmente compiladas a bytecode (clase Java) que puede ejecutarse en cualquier máquina virtual Java (JVM) sin importar la arquitectura de la computadora subyacente.

Es un lenguaje de programación de propósito general, concurrente, orientado a objetos y basado en clases que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo

ejecuten en cualquier dispositivo (conocido en inglés como WORA, o "write once, run anywhere"), lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra. Java es, a partir de 2012, uno de los lenguajes de programación más populares en uso, particularmente para aplicaciones de cliente-servidor de web, con unos 10 millones de usuarios reportados.

La compañía Sun desarrolló la implementación de referencia original para los compiladores de Java, máquinas virtuales, y librerías de clases en 1991 y las publicó por primera vez en 1995. A partir de mayo de 2007, en cumplimiento con las especificaciones del Proceso de la Comunidad Java, Sun volvió a licenciar la mayoría de sus tecnologías de Java bajo la Licencia Pública General de GNU. Otros también han desarrollado implementaciones alternas a estas tecnologías de Sun, tales como el Compilador de Java de GNU y el GNU Classpath.

Entornos de funcionamiento

El diseño de Java, su robustez, el respaldo de la industria y su fácil portabilidad han hecho de Java uno de los lenguajes con un mayor crecimiento y amplitud de uso en distintos ámbitos de la industria de la informática.

- En dispositivos móviles y sistemas empotrados.

Desde la creación de la especificación J2ME (Java 2 Platform, Micro Edition), una versión del entorno de ejecución Java reducido y altamente optimizado, especialmente desarrollado para el mercado de dispositivos electrónicos de consumo se ha producido toda una revolución en lo que a la extensión de Java se refiere.

Es posible encontrar microprocesadores diseñados para ejecutar bytecode Java y software Java para tarjetas inteligentes (JavaCard), teléfonos móviles,

buscapersonas, set-top-boxes, sintonizadores de TV y otros pequeños electrodomésticos.

El modelo de desarrollo de estas aplicaciones es muy semejante a las applets de los navegadores salvo que en este caso se denominan MIDlets.

- En el navegador web

Desde la primera versión de Java existe la posibilidad de desarrollar pequeñas aplicaciones (Applets) en Java que luego pueden ser incrustadas en una página HTML para que sean descargadas y ejecutadas por el navegador web. Estas mini aplicaciones se ejecutan en una JVM que el navegador tiene configurada como extensión (plug-in) en un contexto de seguridad restringido configurable para impedir la ejecución local de código potencialmente malicioso.

El éxito de este tipo de aplicaciones (la visión del equipo de Gosling) no fue realmente el esperado debido a diversos factores, siendo quizás el más importante la lentitud y el reducido ancho de banda de las comunicaciones en aquel entonces que limitaba el tamaño de las applets que se incrustaban en el navegador. La aparición posterior de otras alternativas (aplicaciones web dinámicas de servidor) dejó un reducido ámbito de uso para esta tecnología, quedando hoy relegada fundamentalmente a componentes específicos para la intermediación desde una aplicación web dinámica de servidor con dispositivos ubicados en la máquina cliente donde se ejecuta el navegador.

Las applets Java no son las únicas tecnologías (aunque sí las primeras) de componentes complejos incrustados en el navegador. Otras tecnologías similares pueden ser: ActiveX de Microsoft, Flash, Java Web Start, etc.

- En sistemas de servidor

En la parte del servidor, Java es más popular que nunca, desde la aparición de la especificación de Servlets y JSP (Java Server Pages).

Hasta entonces, las aplicaciones web dinámicas de servidor que existían se basaban fundamentalmente en componentes CGI y lenguajes interpretados. Ambos tenían diversos inconvenientes (fundamentalmente lentitud, elevada carga computacional o de memoria y propensión a errores por su interpretación dinámica).

Los servlets y las JSPs supusieron un importante avance ya que:

- El API de programación es muy sencilla, flexible y extensible.
- Los servlets no son procesos independientes (como los CGIs) y por tanto se ejecutan dentro del mismo proceso que la JVM mejorando notablemente el rendimiento y reduciendo la carga computacional y de memoria requeridas.
- Las JSPs son páginas que se compilan dinámicamente (o se pre-compilan previamente a su distribución) de modo que el código que se consigue una ventaja en rendimiento substancial frente a muchos lenguajes interpretados.

La especificación de Servlets y JSPs define un API de programación y los requisitos para un contenedor (servidor) dentro del cual se puedan desplegar estos componentes para formar aplicaciones web dinámicas completas. Hoy día existen multitud de contenedores (libres y comerciales) compatibles con estas especificaciones.

A partir de su expansión entre la comunidad de desarrolladores, estas tecnologías han dado paso a modelos de desarrollo mucho más elaborados con frameworks (pe Struts, Webwork) que se sobreponen sobre los servlets y las JSPs para

conseguir un entorno de trabajo mucho más poderoso y segmentado en el que la especialización de roles sea posible (desarrolladores, diseñadores gráficos,...) y se facilite la reutilización y robustez de código. A pesar de todo ello, las tecnologías que subyacen (Servlets y JSPs) son substancialmente las mismas.

Este modelo de trabajo se ha convertido en uno de los estándar de-facto para el desarrollo de aplicaciones web dinámicas de servidor.

- En aplicaciones de escritorio

Hoy en día existen multitud de aplicaciones gráficas de usuario basadas en Java. El entorno de ejecución Java (JRE) se ha convertido en un componente habitual en los PC de usuario de los sistemas operativos más usados en el mundo. Además, muchas aplicaciones Java lo incluyen dentro del propio paquete de la aplicación de modo que se ejecuten en cualquier PC.

En las primeras versiones de la plataforma Java existían importantes limitaciones en las APIs de desarrollo gráfico (AWT). Desde la aparición de la biblioteca Swing la situación mejoró substancialmente y posteriormente con la aparición de bibliotecas como SWT hacen que el desarrollo de aplicaciones de escritorio complejas y con gran dinamismo, usabilidad, etc. sea relativamente sencillo.

Plataformas soportadas

Una versión del entorno de ejecución Java JRE (Java Runtime Environment) está disponible en la mayoría de equipos de escritorio. Sin embargo, Microsoft no lo ha incluido por defecto en sus sistemas operativos. En el caso de Apple, éste incluye una versión propia del JRE en su sistema operativo, el Mac OS. También es un producto que por defecto aparece en la mayoría de las distribuciones de GNU/Linux. Debido a incompatibilidades entre distintas versiones del JRE, muchas aplicaciones prefieren instalar su propia copia del JRE antes que confiar

su suerte a la aplicación instalada por defecto. Los desarrolladores de applets de Java o bien deben insistir a los usuarios en la actualización del JRE, o bien desarrollar bajo una versión antigua de Java y verificar el correcto funcionamiento en las versiones posteriores.

XCODE

Xcode es el entorno de desarrollo integrado (IDE, en sus siglas en inglés) de Apple Inc. y se suministra gratuitamente junto con Mac OS X. Xcode trabaja conjuntamente con Interfase Builder, una herencia de NeXT, una herramienta gráfica para la creación de interfases de usuario.

Xcode incluye la colección de compiladores del proyecto GNU (GCC), y puede compilar código C, C++, Objective-C, Objective-C++, Java y AppleScript mediante una amplia gama de modelos de programación, incluyendo, pero no limitado a Cocoa, Carbón y Java. Otras compañías han añadido soporte para GNU Pascal, Free Pascal, Ada y Perl.

Entre las características más apreciadas de Xcode está la tecnología para distribuir el proceso de construcción a partir del código fuente entre varios ordenadores, utilizando Bonjour.

PÁGINA WEB

Una **página web** o **página electrónica** es un documento o información electrónica capaz de contener texto, sonido, vídeo, programas, enlaces, imágenes, y muchas otras cosas, adaptada para la llamada World Wide Web, y que puede ser accedida mediante un navegador. Esta información se encuentra generalmente en formato HTML o XHTML, y puede proporcionar navegación (acceso) a otras páginas web mediante enlaces de hipertexto. Las páginas web frecuentemente también

incluyen otros recursos como ser hojas de estilo en cascada, guiones (*scripts*), imágenes digitales, entre otros.

Las páginas web pueden estar almacenadas en un equipo local o un servidor webremoto. El servidor web puede restringir el acceso únicamente a redes privadas, por ejemplo, en una intranet corporativa, o puede publicar las páginas en la World Wide Web. El acceso a las páginas web es realizado mediante una transferencia desde servidores, utilizando el protocolo de transferencia de hipertexto (HTTP).

Características

Una página web está compuesta principalmente por información (sólo texto y/o módulos multimedia) así como por hiperenlaces; además puede contener o asociar hoja de estilo, datos de estilo para especificar cómo debe visualizarse, y también aplicaciones embebidas para así permitir interactividad.

Las páginas web son escritas en un lenguaje de marcado que provee la capacidad de manejar e insertar hiperenlaces, generalmente HTML.

El contenido de la página puede ser predeterminado (página web estática) o generado al momento de visualizarla o solicitarla a un servidor web (página web dinámica). Las páginas dinámicas que se generan al momento de la visualización, se especifican a través de algún lenguaje interpretado, generalmente JavaScript, y la aplicación encargada de visualizar el contenido es la que realmente debe generarlo. Las páginas dinámicas que se generan, al ser solicitadas, son creadas por una aplicación en el servidor web que alberga las mismas.

Respecto a la estructura de las páginas web, algunos organismos, en especial el World Wide Web Consortium (W3C), suelen establecer directivas con la

intención de normalizar el diseño, y para así facilitar y simplificar la visualización e interpretación del contenido.

Una página web es en esencia una tarjeta de presentación digital, ya sea para empresas, organizaciones, o personas, así como una tarjeta de presentación de ideas y de informaciones y de teorías. Así mismo, la nueva tendencia orienta a que las páginas web no sean sólo atractivas para los internautas, sino también optimizadas (preparadas) para los buscadores a través del código fuente. Forzar esta doble función puede, sin embargo, crear conflictos respecto de la calidad del contenido.

Si hablamos de posicionamiento web, una página web es la base para optimizar todo un sitio web el cual es un conjunto de páginas web.

PHP

PHP (acrónimo recursivo de *PHP: Hypertext Preprocessor*) es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML.

Ejemplo #1 Un ejemplo introductorio

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">
<html>
   <head>
      <title>Ejemplo</title>
   </head>
   <body>
      <?php
```

```
        echo "¡Hola, soy un script de PHP!";
    ?>
  </body>
</html>
```

En lugar de usar muchos comandos para mostrar HTML (como en C o en Perl), las páginas de PHP contienen HTML con código incrustado que hace "algo" (en este caso, mostrar "¡Hola, soy un script de PHP!). El código de PHP está encerrado entre las etiquetas especiales de comienzo y final <?php y ?> que permiten entrar y salir del "modo PHP".

Lo que distingue a PHP de algo como Javascript del lado del cliente es que el código es ejecutado en el servidor, generando HTML y enviándolo al cliente. El cliente recibirá el resultado de ejecutar el script, aunque no se sabría el código subyacente que era. El servidor web puede ser incluso configurado para que procese todos los ficheros HTML con PHP, por lo que no hay manera de que los usuarios puedan saber qué se tiene debajo de la manga.

Lo mejor de usar PHP es que es extremadamente simple para el principiante, pero a su vez ofrece muchas características avanzadas para los programadores profesionales. No sienta miedo de leer la larga lista de características de PHP. En unas pocas horas podrá empezar a escribir sus primeros scripts.

Aunque el desarrollo de PHP está centrado en programación de scripts del lado del servidor, se puede utilizar para muchas otras cosas.

FUNCIONAMIENTO DE PHP

PHP está enfocado principalmente a la programación de scripts del lado del servidor, por lo que se puede hacer cualquier cosa que pueda hacer otro programa

CGI, como recopilar datos de formularios, generar páginas con contenidos dinámicos, o enviar y recibir cookies. Aunque PHP puede hacer mucho más.

Existen principalmente tres campos principales donde se usan scripts de PHP.

- Scripts del lado del servidor. Este es el campo más tradicional y el foco principal. Se necesitan tres cosas para que esto funcione. El analizador de PHP (módulo CGI o servidor), un servidor web y un navegador web. Es necesario ejecutar el servidor, con una instalación de PHP conectada. Se puede acceder al resultado del programa PHP con un navegador, viendo la página de PHP a través del servidor. Todo esto se puede ejecutar en su máquina si está experimentado con la programación de PHP. Véase la sección sobre las instrucciones de instalación para más información.

- Scripts desde la línea de comandos. Se puede crear un script de PHP y ejecutarlo sin necesidad de un servidor o navegador. Solamente es necesario el analizador de PHP para utilizarlo de esta manera. Este tipo de uso es ideal para scripts ejecutados regularmente usando cron (en *nix o Linux) o el Planificador de tareas (en Windows). Estos scripts también pueden usarse para tareas simples de procesamiento de texto. Véase la sección Uso de PHP en la línea de comandos para más información.

- Escribir aplicaciones de escritorio. Probablemente PHP no sea el lenguaje más apropiado para crear aplicaciones de escritorio con una interfaz gráfica de usuario, pero si se conoce bien PHP, y se quisiera utilizar algunas características avanzadas de PHP en aplicaciones del lado del cliente, se puede utilizar PHP-GTK para escribir dichos programas. También es posible de esta manera escribir aplicaciones independientes de una plataforma. PHP-GTK es una extensión de PHP, no disponible en la distribución principal.

PHP puede usarse en todos los principales sistemas operativos, incluyendo Linux, muchas variantes de Unix (incluyendo HP-UX, Solaris y OpenBSD), Microsoft Windows, Mac OS X, RISC OS y probablemente otros más. PHP admite la mayoría de servidores web de hoy en día, incluyendo Apache, IIS, y muchos otros. Esto incluye cualquier servidor web que pueda utilizar el binario de PHP FastCGI, como lighttpd y nginx. PHP funciona tanto como módulo como procesador de CGI.

De modo que con PHP se tiene la libertad de elegir el sistema operativo y el servidor web. Además, se tiene la posibilidad de utilizar programación por procedimientos o programación orientada a objetos (POO), o una mezcla de ambas.

Con PHP no se está limitado a generar HTML. Entre las capacidades de PHP se incluyen la creación de imágenes, ficheros PDF e incluso películas Flash (usando libswf y Ming) generadas sobre la marcha. También se puede generar fácilmente cualquier tipo de texto, como XHTML y cualquier otro tipo de fichero XML. PHP puede autogenerar estos ficheros y guardarlos en el sistema de ficheros en vez de imprimirlos en pantalla, creando una caché en el lado del servidor para contenido dinámico.

Una de las características más potentes y destacables de PHP es su soporte para un amplio abanico de bases de datos. Escribir una página web con acceso a una base de datos es increíblemente simple utilizando una de las extensiones específicas de bases de datos (p.ej., para mysql), o utilizar una capa de abstracción como PDO, o conectarse a cualquier base de datos que admita el estándar de Conexión Abierta a Bases de Datos por medio de la extensión ODBC. Otras bases de datos podrían utilizar cURL osockets, como lo hace CouchDB.

PHP también cuenta con soporte para comunicarse con otros servicios usando protocolos tales como LDAP, IMAP, SNMP, NNTP, POP3, HTTP, COM (en Windows) y muchos otros. También se pueden crear sockets de red puros e

interactuar usando cualquier otro protocolo. PHP tiene soporte para el intercambio de datos complejos de WDDX entre virtualmente todos los lenguajes de programación web. Y hablando de interconexión, PHP posee soporte para la instalación de objetos Java y usarlos de forma transparente como objetos de PHP. PHP tiene útiles características de procesamiento de texto, las cuales incluyen las expresiones regulares compatibles con Perl (PCRE), y muchas extensiones y herramientas para el acceso y análisis de documentos XML. PHP estandariza todas las extensiones XML sobre el fundamento sólido de libxml2, y amplía este conjunto de características añadiendo soporte para Simple XML, XML Reader y XL Write.

Existen otras extensiones interesantes, las cuales están categorizadas alfabéticamente y por categoría. Y hay extensiones adicionales de PECL que podrían estar documentadas o no dentro del manual de PHP, tal como » XDebug.

INSTALACIÓN DE PHP EN WINDOWS 7 Y 8

Instalaremos un programa llamado AppServ el cual contiene:

Apache

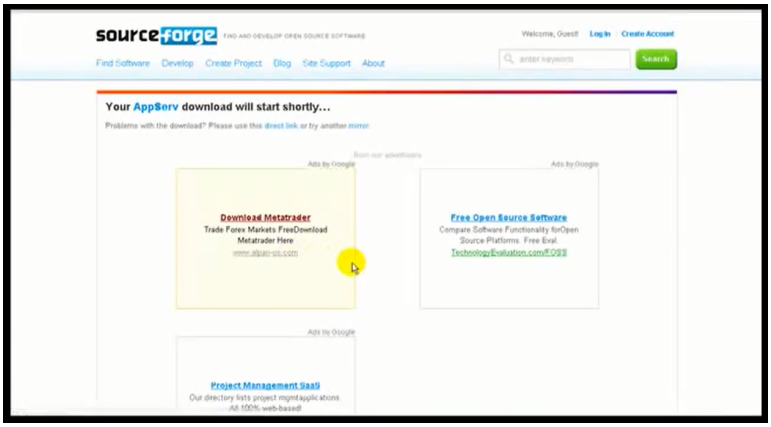
Php

MySQL

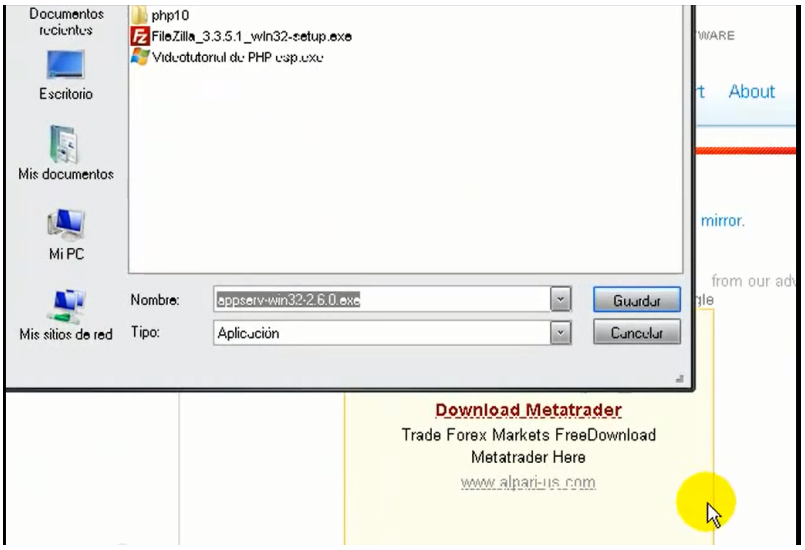
PHPMyAdmin.

Nos iremos a la página de appserv y escogeremos según si es nuestra máquina de 32 bits o 64 bist y también dependiendo de qué gama queremos los programas, en este caso descargaremos el AppServ de 2.6.0.

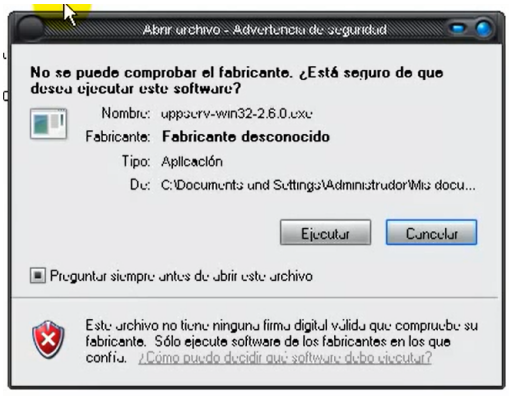
Proseguiremos a descargarlo y aparecerá lo siguiente. Y le daremos clic en el cuadro amarillo.



Después aparecerá otro cuadro y le pondremos donde lo queremos guardar.

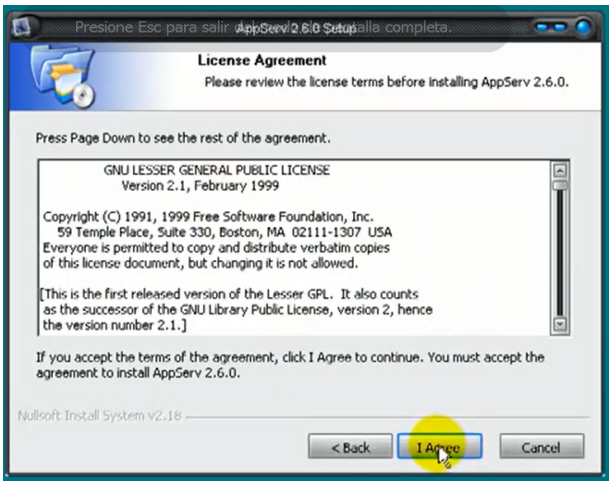


Después de descargarse completo iremos a la carpeta donde lo guardamos y le damos doble clic derecho apareciendo un cuadro después y le damos clic en ejecutar.

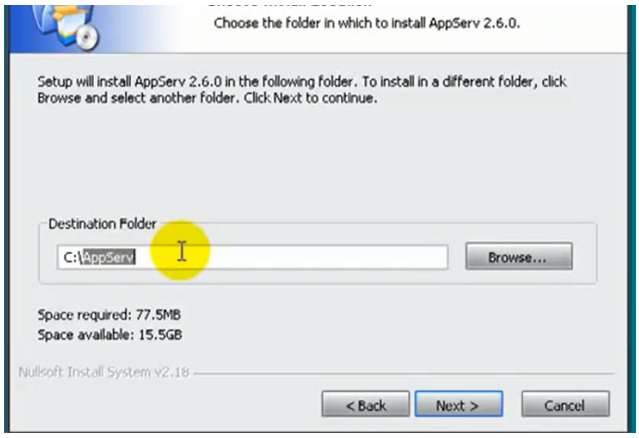


Aparecerá una ventana como la siguiente y le damos en next.

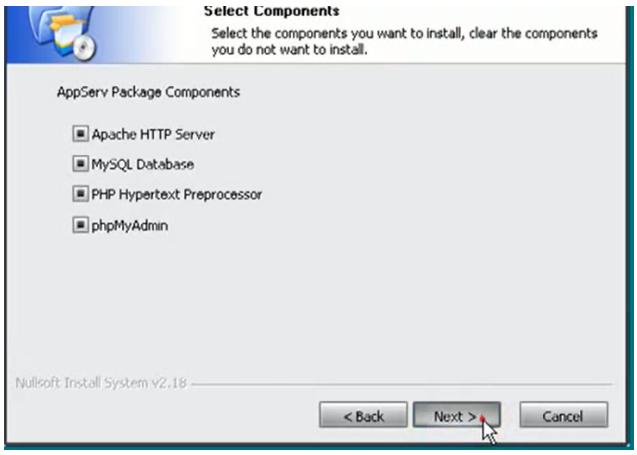
Después aceptamos los términos y condiciones dándole clic en I Agree



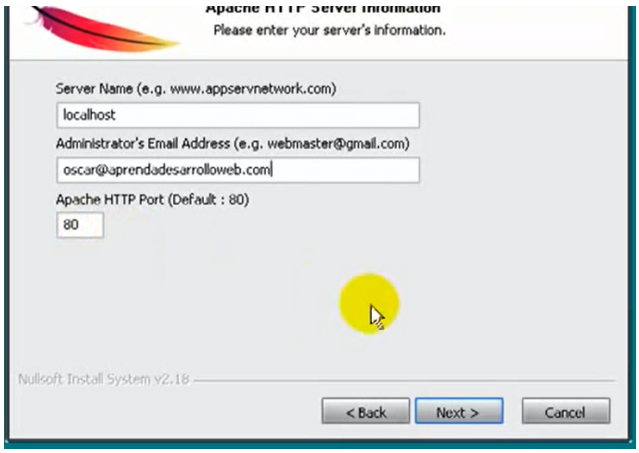
Después aparecerá un cuadro donde nos dirá en qué parte guardará lo instalado que será en el disco C en una carpeta que creo llamada AppServ y le damos clic en next.



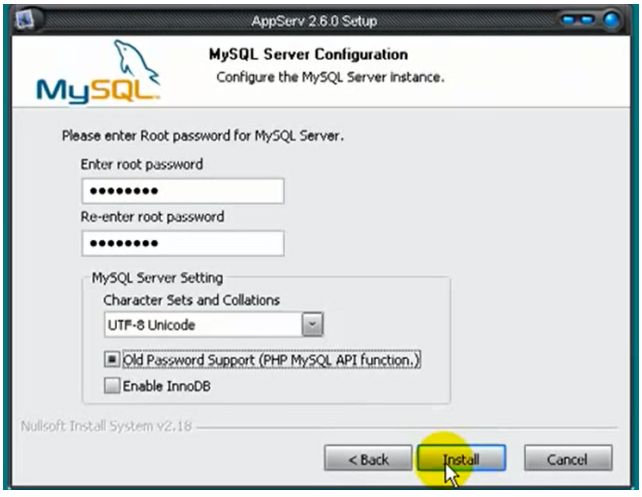
Luego aparecerá un cuadro donde saldrán los distintos programas de paquete que vamos utilizar en este caso los dejamos todos y le damos clic en next.



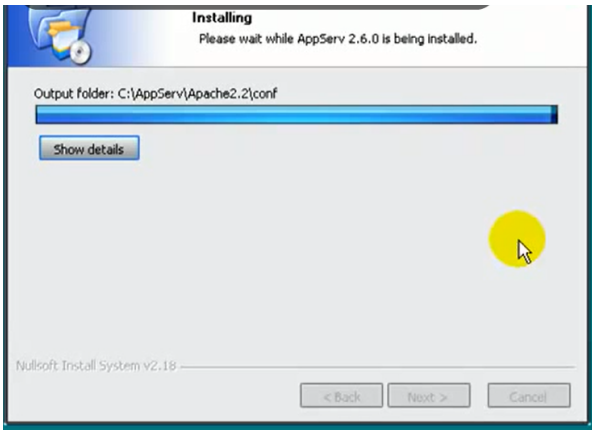
Después nos pedirá el nombre del servidor, al que le pondremos localhost y un correo de servidor que puede ser cualquiera y le damos clic en next.



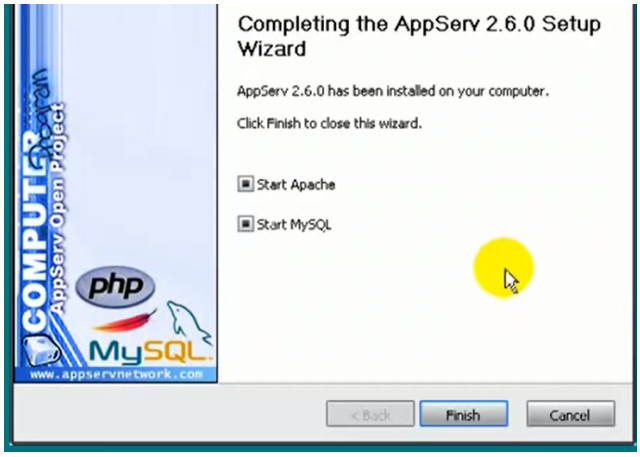
Después nos pedirá el password. Debemos tener mucho cuidado con esto, ya que para entrar a alguna página nos lo pedirá, lo ponemos 2 veces y le seleccionamos en la primera de las 2 opciones que se nos da. Le damos clic en instalar.



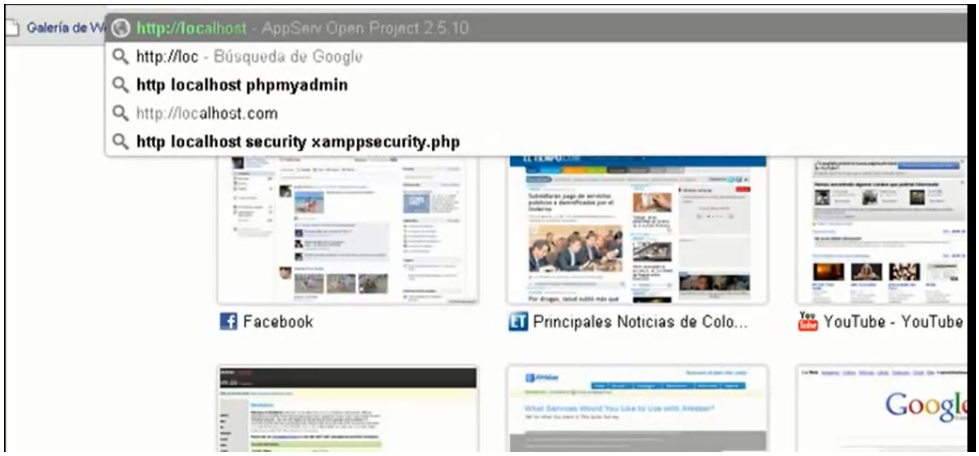
Prosiguiendo a la instalación del paquete.



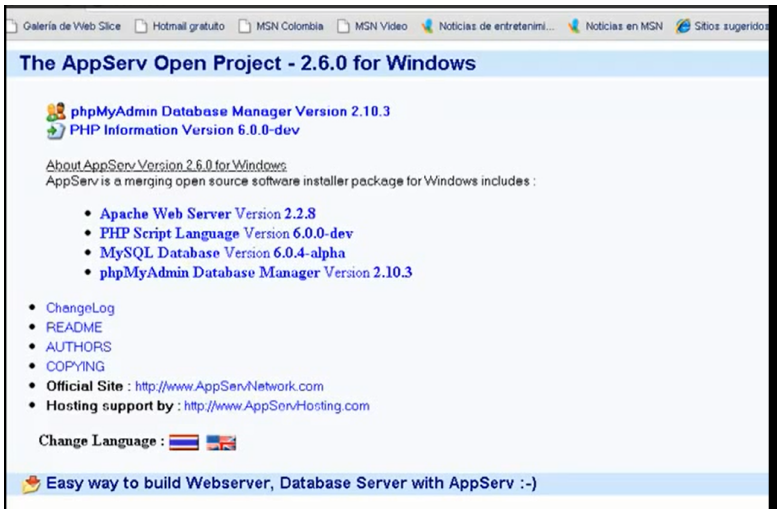
Nos mostrará en la pantalla que ya se instaló el paquete y le damos en finish



Y listo, está instalado php. Para verificar que se instaló correctamente nos vamos al navegador y ponemos localhost.



Si se instaló correctamente debe aparecer lo siguiente, que indica en que versión de php se instaló.



SINTAXIS BÁSICA

Saliendo de HTML

Hay cuatro formas de salir de HTML y entrar en el "modo de código PHP":

Ejemplo 5-1. Formas de salir de HTML

1. <? echo ("esta es la más simple, una instrucción de procesado SGML\n"); ?>
2. <?php echo ("si quiere servir documentos XML, haga esto\n"); ?>
3. <script language="php"> echo ("a algunos editores (como FrontPage) no les gustan las intrucciones de procesado"); </script>
4. <% echo ("Puedes también usar etiquetas tipo ASP"); %>
<%=$variable; # Esto es una forma abreviada de "<%echo." %>

La primera forma sólo está disponible si se han habilitado las etiquetas cortas. Esto se puede hacer a través de la función short_tags ( ),habilitando la opción de configuración short_open_tagenel archivo de configuración de PHP, o compilando PHPcon la opción–enable-short-tags en configure.

La cuarta manera está disponible sólo si se han habilitado las etiquetas tipo ASP usando la opción de configuración asp_tags.

Nota: El soporte para las etiquetas tipo ASP se añadió en 3.0.4.

La etiqueta de cierre de un bloque incluirá el carácter de nueva línea final si hay uno presente.

Separación de instrucciones

Las instrucciones se separan igual que en Coperl- terminando cada sentencia con un punto y coma.

La etiqueta de cierre (?>) también implica el fin de la sentencia, así lo siguiente es equivalente:

```
<?php
echo "Esto es una prueba";
?>
<?php echo "Esto es una prueba" ?>
```

Comentarios

PHP soporta comentarios tipo “C”, “C++” y shell de Unix. Por ejemplo:

```
<?php
echo "Esto es una prueba"; // Esto es un comentario tipo C++ para una línea
/*Esto es un comentario multilínea
otra línea más de comentario*/
echo "Esto es aún otra prueba";
echo "Una Prueba Final"; # Este es un comentario tipo Shell
?>
```

El tipo de comentario de "una línea" sólo comenta, en realidad, hasta el fin de la línea o el bloque actual de código PHP, lo que venga primero.

```
<h1> Esto es un <?# echo "simple";?> ejemplo.</h1>
<p> La cabecera de arriba dirá “Esto es un ejemplo”.
```

Se debería tener cuidado para no anidar comentarios de tipo “C”, lo cual puede ocurrir cuando se comentan grandes bloques.

```
<?php
/*
echo "Esto es una prueba"; /* Este comentario causará un problema */
*/
?>
```

VARIABLES

Todas la variables deben ser precedidas por un signo dólar ($), y le asignamos contenido con el signo igual (=). Con las variables, PHP distingue entre mayúsculas y minúsculas, por lo que PHP distingue entre mayúsculas y minúsculas, por lo que no es lo mismo $myvar que $Myvar, éstas son dos variables totalmente distintas.

```
<html>
<body>
<?php
$myvar = "SEVILLA \n";
$Myvar = "MADRID \n";
//Esto imprimirá SEVILLA
echo $myvar;
//Esto imprimirá MADRID
ECHO $Myvar;
?>
</body>
</html>
```

El uso de la barra invertida, como en \n, no es obligatorio, pero ayuda a la depuración del código que enviamos al navegador, además del \n existen otros usos:

\" Carácter dobles comillas
\\ Carácter barra invertida
\n Nueva línea
\r Retorno de carro
\t Tabulador horizontal.

Variables variables

A veces es conveniente tener nombres de variables variables. Dicho de otro modo, son nombres de variables que se pueden establecer y usar dinámicamente. Una variable normal se establece con una sentencia como:

$a = "hello";

Una variable variable toma el valor de una variable y lo trata como el nombre de una variable. En el ejemplo anterior, hello, se puede usar como el nombre de una variable utilizando dos signos de dólar. p.ej.

$$a = "world";

En este momento se han definido y almacenado dos variables en el árbol de símbolos de PHP: $a, que contiene "hello", y $hello, que contiene "world". Es más, esta sentencia: echo "$a ${$a}"; produce el mismo resultado que: echo "$a $hello"; p. ej. ambas producen el resultado: hello world.

Para usar variables variables con arrays, hay que resolver un problema de ambigüedad. Si se escribe $$a[1] el intérprete necesita saber si nos referimos a utilizar $a[1] como una variable, o si se pretendía utilizar $$a como variable y el índice [1] como índice de dicha variable. La sintaxis para resolver esta ambigüedad es: ${$a[1]} para el primer caso y ${$a}[1] para el segundo.

CONSTANTES

Las constantes son similares a las variables, con la salvedad de que no llevan el signo dólar delante, y sólo la podemos asignar una vez. Para definir una constante usaremos la función “define” como sigue:

```
<html>
<body>
<?php
define ("CONSTANTE", "Hola Mundo");
printf (CONSTANTE);
?>
</body>
</html>
```

PHP crea diversas constantes al arrancar, como PHP_VERSION que contiene la versión de PHP, TRUE que le asigna 1 o FALSE que le asigna 0.

TIPOS DE OPERADORES

Operadores Aritméticos:

$a + $b Suma
$a - $b Resta
$a * $b Multiplicación
$a / $b &ss=código en linea>$a / $b División
$a % $b Resto de la división de $a por $b
$a++ Incrementa en 1 a $a
$a-- Resta 1 a $a

Operadores de Cadenas

El único operador de cadenas que existen es el de concatenación, el punto. Pero PHP dispone de toda una batería de funciones que permiten trabajar cómodamente con las cadenas.

```
$a = "Hola";
$b = $a . "Mundo"; // Ahora $b contiene "Hola Mundo"
```

En este punto hay que hacer una distinción, la interpretación que hace PHP de las simples y dobles comillas. En el segundo caso PHP interpretará el contenido de la cadena.

```
$a = "Mundo";
echo = 'Hola $a'; //Esto escribirá "Hola $a"
echo = "Hola $a"; //Esto escribirá "Hola Mundo&q; //Esto escribirá "Hola Mundo".
```

Operadores de Comparación

$a < $b $a menor que $b
$a > $b $a mayor que $b
$a <= $b $a menor o igual que $b
$a >= $b $a mayor o igual que $b
$a == $b $a igual que $b
$a != $b $a distinto que $b

Operadores Lógicos

$a AND $b Verdadero si ambos son verdadero
$a && $b Verdadero si ambos son verdadero
$a OR $b Verdadero si alguno de los dos es verdadero
$a !!$b Verdadero si alguno de los dos es verdadero
$a XOR $b Verdadero si sólo uno de los dos es verdadero
!$a Verdadero si $a es falso, y recíprocamente.

Operadores de Asignación

$a = $b Asigna a $a el contenido de $b

$a += $b Le suma a $b a $a

$a -= $b Le resta a $b a $a

$a *= $b Multiplica $a por $b y lo asigna a $a

$a /= $b Divide $a por $b y lo asigna a $a

$a .= $b Añade la cadena $b a la cadena $a.

EJERCICIOS EN PHP

1. Crear una página que imprima “Hola”.
2. Dada la longitud de un lado calcular el área y el perímetro de un cuadrado.
3. Dada la longitud de un lado y de la altura calcular el área y el perímetro de un triángulo.
4. Obtener el valor de e, dados c y m de $e=m*c^2$.
5. Dados el radio y la altura de un cilindro, obtener su volumen.
6. Dado un número obtener x^2, x^3 y x^{-1}.
7. Crear la tabla del 2, en lista.
8. Dados 5 números, obtener la suma de ellos.
9. Obtener la media de 10 números.
10. Dados dos puntos en $\mathbb{R}^2$ calcular su distancia.
11. Dada la longitud de un lado de la base y la altura, calcular el volumen de una pirámide.
12. Dada la longitud del radio y la altura, calcular el volumen de un cono.
13. Convertir de grados Fahrenheit a grados Centígrados.
14. Convertir de grados Centígrados a Fahrenheit.
15. Convertir de metros a pulgadas.
16. Convertir de pulgadas a metros.
17. Convertir de litros a galones.

18. Convertir de galones a litros.
19. Obtener las raíces de una ecuación cuadrática.
20. Calcular la pendiente de una recta dados dos puntos.
21. Dado el número 5, obtener los primeros diez números impares a partir de él.
22. Dado el número 2, obtener los primeros 10 números pares a partir de él.
23. Dado un número sacar su factorial.
24. Dada una cantidad de pesos y un interés dado. Calcular el interés pagado en un año si la reinversión es mensual y diario.

SENTENCIAS DE CONTROL

Las sentencias de control permiten ejecutar bloque de códigos dependiendo de unas condiciones. Para PHP el 0 es equivalente a Falso y cualquier otro número es Verdadero.

IF...ELSE

La sentencia IF...ELSE permite ejecutar un bloque de instrucciones si la condición es Verdadera y otro bloque de instrucciones si ésta es Falsa. Es importante tener en cuenta que instrucciones si ésta es Falsa. Es importante tener en cuenta que la condición que evaluemos ha de estar encerrada entre paréntesis (esto es aplicable a todas la sentencias de control).

```
if (condición) {

Este bloque se ejecuta si la condición es VERDADERA

} else {

Este boque se ejecuta si la condición es FALSA
```

```
}
```

Existe una forma sencilla de usar la sentencia IF cuando no tenemos que usar el ELSE y solo tenenemos que ejecutar una línea de código.

```
if ($a > 4) echo "$a es mayor que 4";
```

IF...ELSEIF...ELSE

La sentencia IF...ELSEIF...ELSE permite ejecuta varias condiciones en cascada. Para este caso veremos un ejemplo, en el que utilizaremos los operadores lógicos.

```
<?php
if ($nombre == ""){
        echo "Tú no tienes nombre";
} elseif (($nombre=="eva") OR ($nombre=="Eva")) {
        echo "
        echo "Tu nombre es EVA";<
} else {
        echo "Tu nombre es " . $nombre;
}
```

SWITCH...CASE...DEFAULT

Una alternativa a IF...ELSEIF...ELSE, es la sentencia SWITCH, la cuál evalúa y compara cada expresión de la sentencia CASE con la expresión que evaluamos, si llegamos al final de la lista de CASE y encuentra una condición Verdadera, ejecuta el código de bloque que haya en DEFAULT. Si encontramos una condición verdadera debemos ejecutar un BREAK para que la sentencia SWITCH no siga buscando en la lista de CASE. Veamos un ejemplo.

```
<?php
switch ($dia) {
case "Lunes":
echo "Hoy es Lunes";
break;
C
case "Martes":
echo "Hoy es Martes";
break;
case "Miercoles":
echo "Hoy es Miercoles";
break;
case "Jueves":
echo "Hoy es Jueves";
break;
case "Viernes":
echo "Hoy es Viernes";
break;
case "Sábado":
echo "Hoy es Sábado";
break;
case "Domingo":
echo "Hoy es Domingo";
break;
default:
default:
echo "Esa cadena no corresponde a ningún día de la semana".
```

WHILE

La sentencia WHILE ejecuta un bloque de código mientras se cumpla una determinada condición.

```
<?php
$num = 1;
while ($num < 5) {
echo $num;
$num++
}
?>
```

Podemos romper un bucle WHILE utilizando la sentencia BREAK.

```
<?php
$num = 1;
while ($num < 5) {
echo $num;
if ($num == 3) {
echo "Aquí nos salimos \n";
break
}
$num++
}
?>
```

DO...WHILE

Esta sentencia es similar a WHILE, salvo que con esta sentencia primero ejecutamos el bloque de código y después se evalúa la condición, por lo que el bloque de código se ejecuta siempre al menos una vez.

```
<?php
$num = 1;
do {
echo $num;
if ($num == 3) {
echo "Aquí nos salimos \ n";
break
}
$num++
} while ($num < 5);
?>
```

FOR

El bucle FOR no es estrictamente necesario, cualquier bucle FOR puede ser sustituido fácilmente por otro WHILE. Sin embargo, el bucle FOR resulta muy útil cuando debemos ejecutar un bloque de código a condición de que una variable se encuentre entre un valor mínimo y otro máximo. El bucle FOR también se puede romper mediante la sentencia BREAK.

```
<?php
for ($num = 1; Snum <=5; $num++) {
echo $num;
if ($num == 3) {
echo "Aquí nos salimos \ n";
break
```

BREAK

Break escapa de la estructuras de control iterante (bucle) actuales for, while, o switch.

Break acepta un parámetro opcional, el cual determina cuantas estructuras de control hay que escapar.

```
$arr = array (’one’, ’two’, ’three’, ’four’, ’stop’, ’five’);
while (list (, $val) = each ($arr)) {
        if ($val == ’stop’) {
                break; /* You could also write ’break 1;’ here. */
        }
        echo "$val<br>\n";
}
/* Using the optional argument. */
$i = 0;
while (++$i) {
        switch ($i) {
        case5:
                echo "At 5<br>\n";
                break1;  /* Exit only the switch. */
        case10:
                echo "At 10; quitting<br>\n";
                break2; /* Exit the switch and the while. */
        default:
                break;
        }
}
```

CONTINUE

CONTINUE se utiliza dentro de las estructuras iterativas para saltar el resto de la iteración actual del bucle y continuar la ejecución en la evaluación de la condición, y luego comenzar la siguiente iteración.

Nota.- Tenga en cuenta que en PHP la sentencia switch se considera una estructura iterativa para los propósitos de *continue.*

Continue acepta un argumento numérico opcional, que indica a cuántos niveles de bucles encerrados se ha de saltar al final. El valor por omisión es 1, por lo que salta al final del bucle actual.

```
<?php
while (list ($clave, $valor) = each($arr)) {
    if (!($clave % 2)) { // saltar los miembros impares
        continue;
    }
    hacer_algo ($valor);
}
$i = 0;
while ($i++ < 5) {
    echo "Exterior<br />\n";
    while (1) {
        echo "Medio<br />\n";
        while (1) {
            echo "Interior<br />\n";
            continue 3;
        }
        echo "Esto nunca se imprimirá.<br />\n";
    }
    echo "Ni esto tampoco.<br />\n";
}
?>
```

Omitir el punto y coma después del *continue* puede llevar a confusión. He aquí un ejemplo de lo que no se debe hacer.

```
<?php
for ($i = 0; $i < 5; ++$i) {
    if ($i == 2)
        continue
    print "$i\n";
}
?>
```

Se esperaría que el resultado fuera:

```
0
1
3
4
```

pero la salida de este script será:

```
2
```

debido a que *continue print "$i\n";* se evalúa completo como una sola expresión, y así print se llama solamente cuando *$i == 2* es verdadero (El valor de retorno de *print* es pasado a *continue* como el argumento numérico).

Registro de cambios para *continue*	
Versión	**Descripción**
5.4.0	*continue 0;* ya no es válido. En versiones anteriores era interpretado de la misma manera que *continue 1;*.

Registro de cambios para *continue*	
Versión	**Descripción**
5.4.0	Se eliminó la posibilidad de pasar variables (por ejemplo, *$num = 2; continue $num;*) como el argumento numérico.

DECLARE

El constructor *declare* es usado para fijar directivas de ejecución para un bloque de código. La sintaxis de *declare* es similar a la sintaxis de otros constructores de control de flujo:

declare (directive)
statement

La sección *directive* permite que el comportamiento de *declare* sea configurado. Actualmente, sólo dos directivas están reconocidas: *ticks* (Ver abajo para más información sobre la directiva ticks) y encoding (Ver abajo para más información sobre la direvtiva encoding).

Nota: La directiva encoding fue agregada en PHP 5.3.0

La parte *statement* del bloque declare será ejecutada como se ejecuta y que efectos secundarios ocurran durante la ejecución puede depender de la directiva fijada en el bloque *directive*.

El constructor *declare* también se puede utilizar en el alcance global, afectando a todo el código que le sigue (sin embargo, si el archivo con el *declare* fue incluido entonces no afectará al archivo padre).

```
<?php
// estos son lo mismo:
// se puede usar ésto:
declare (ticks=1) {
 // script entero aquí
}
// o se puede usar ésto:
declare (ticks=1);
// script entero aquí
?>
```

TICKS

Un tick es un evento que ocurrepara cada sentencia tickable N de bajo nivel ejecutada por el intérprete dentro del bloque *declare*. El valor para N se especifica usando ticks=N dentro del bloque de declare de la sección *directive*.

No todas las sentencias son tickable. Por lo general, expresiones de condición y expresiones de argumento no son tickables.

Los eventos que ocurren en cada tick se especifican mediante el register_tick_function ( ) (Ver el ejemplo abajo para más detalles). Tener en cuenta que más de un envento puede ocurrir por cada tick.

Ejemplo # 1 Ejemplo de uso del tick

```
<?php
declare (ticks=1);
// Una función llamada en cada evento tick
function tick_handler ( )
```

```
{
    echo "tick_handler ( ) llamado\n";
}
register_tick_function ('tick_handler');
$a = 1;
if ($a > 0) {
    $a += 2;
    print($a);
}
?>
```

Ejemplo # 2 Ejemplo de uso de ticks

```
<?php
function tick_handler ( )
{
  echo "tick_handler ( ) llamado\n";
}
$a = 1;
tick_handler ( );
if ($a > 0) {
    $a += 2;
    tick_handler ( );
    print ($a);
    tick_handler ( );
}
tick_handler ( );
?>
```

ENCODING

Una codificación de script puede ser especificada para cada script usando la directive encoding.

Ejemplo # 3 Declarando un encoding para el script

```
<?php
Declare (encoding='ISO-8859-1');
// código aquí
?>
```

Precaución

Cuando se combina con espacios de nombres, la única sintaxis legal para declarer es declare (*encoding*='…'); donde … es el valor del encoding. declare (*encoding*='…') { } resultará en un error de análisis cuando se combina con espacios de nombres.

El valor declarado de encoding es ignorado en PHP 5.3 a menos que php este compilado con –*enable-zend-multibyte*.

Tener en cuenta que PHP no expone si –*enable-zend-multibyte* fue utilizado para compilar PHP que no sea por phpinfo ( ).

RETURN

Si se llama desde una función, la sentencia return inmediatamente termina la ejecución de la función actual, y devuelve su argumento como el valor de la llamada a la función. *return* tambien pondrá fin a la ejecución de una sentencia eval ( ) o a un archivo de script.

Si se llama desde el ámbito global, entonces la ejecución del script actual se termina. Si el archivo script actual fue incluido o requerido con include o require, entonces el control es pasado de regreso al archive que hizo el llamado. Además, si el archivo script actual fue incluido con include, entonces el valor dado a *return* será retornado como el valor de la llamada include. Si *returnes* llamado desde dentro del fichero del script principal, entonces termina la ejecución del script. Si el archivo script actual fue nombrado por las opciones de configuración auto_prepend_file o auto_aooend_file en php.ini, entonces se termina la ejecución de ese archive script.

Nota.- Cabe señalar que dado que *return* es un constructor del lenguaje y no una función, los paréntesis que rodean sus argumentos, no son necesarios. Es común no utilizarlos y en realidad se debería hacer así a fin de que PHP tenga menos trabajo que hacer en este caso.

Nota.- Si no se suministra un parámetro, entonces el paréntesis debe omitirse y NULL será retornado. Llamadas a *return* con paréntesis pero sin argumentos, resultarán en un error del intérprete.

Nota.- *Nunca* se deben usar paréntesis alrededor de la variable de retorno cuando se retorna por referencia, ya que esto no funcionará. Sólo se pueden retornar variables por referencia, no el resultado de una sentencia. Si se utiliza *return* ($a); entonces no se está retornando una variable, sino el resultado de la expression ($a) (el cual es, por supuesto, el valor de $a).

REQUIRE

Require es idéntico a include except que en caso de fallo producirá un error fatal de nivel E_COMPILE_ERROR. En otras palabras, éste detiene el script mientras que include solo emitirá una advertencia (E_WARNING) lo cual permite continuar el script.

INCLUDE

La sentencia *include* incluye y evalua el archivo especificado.

Los archivos son incluidos con base en la ruta de acceso dada o, si ninguna es dada, el include_path especificado. Si el archivo no se encuentra en el include_path, *include* finalmente verificará en elpropio directorio del script que hace el llamado y en el directorio de trabajo actual, antes de fallar. El constructor *include* emitirá una advertencia si no puede encontrar un archivo, éste es un comportamiento diferente al de require, el cual emitirá un error fatal.

Si una ruta es definida –ya sea absoluta (comenzando con una letra de unidad o / en Windows o / en sistemas Unix/Linux) o relativa al directorio actual (comenzando con .o..) – el include_path será ignorado por completo. Por ejemplo, si un nombre de archivo comienza con …/, el intérprete buscará en el directorio padre para encontrar el archivo solicitado.

Cuando se incluye un archivo, el código que contiene hereda el ámbito de las variables de la línea en la cual ocurre la inclusión. Cualquier variable disponible en esa línea de archivo que hace el llamado, estará disponible en el archivo llamado, desde ese punto en adelante. Sin embargo, todas las funciones y clases definidas en el archivo incluido tienen el ámbito global.

Ejemplo # 1 Ejemplo básico de *include*

vars.php

```
<?php
$color = 'verde';
$fruta = 'manzana';
?>
```

test.php

```
<?php
echo "Una $fruta $color"; // Una
include 'vars.php';
echo "Una $fruta $color"; // Una manzana verde
?>
```

Si la inclusión ocurre al interior de una función dentro del archivo que hace el llamado, entonces todo el código contenido en el archivo llamado se comportará como si hubiera sido definida dentro de esa función. Por lo tanto, seguirá el ámbito de las variables de esa función. Una excepción a esta regla son las constantes mágicas las cuales son evaluadas por el intérprete antes que ocurra la inclusión.

Ejemplo # 2 Incluyendo dentro de funciones

```
<?php
function foo ( )
{
    global $color;
    include 'vars.php';
    echo "Una $fruta $color";
}
/* vars.php está en el ámbito de foo ( ) así que *
* $fruta NO está disponible por fuera de éste  *
* ámbito. $color sí está porque fue declarado *
* como global.         */
foo ( );           // Una manzana verde
echo "Una $fruta $color";   // Una verde
?>
```

Cuando un archivo es incluido, el intérprete abandona el modo PHP e ingresa al modo HTML al comienzo del archivo objetivo y se reanuda de nuevo al final. Por esta razón, cualquier código al interior del archivo objetivo que deba ser ejecutado como código PHP, tendrá que ser encerrado dentro de etiquetas válidas de comienzo y terminación de PHP.

Si las “envolturas URL include” están activadas en PHP, se puede especificar el archivo a ser incluido usando una URL en lugar de una ruta de acceso local. Si el servidor objetivo interpreta el archivo objetivo como código PHP, las variables se pueden pasar al archivo incluido usando una string de petición como la usada con HTTP GET. Esto no es, en estricto rigor, lo mismo que haber incluido el archivo y que haya heredado el ámbito de variables del archivo padre; el script realmente está siendo ejecutado en el servidor remoto y el resultado entonces se incluye dentro del script local.

Advertencia

Versiones de PHP para Windows anteriores a 4.3.0, no soportan el acceso remeto a archivos para esta función, no funcionará ni activando siquiera allow url fopen.

Ejemplo # 3 *include* por medio de HTTP

```
<?php
/* Este ejemplo asume que www.example.com está configurado para interpretar archivos
* .php y no archivos .txt. Además, aquí 'Funciona' quiere decir que las variables
* $foo y $bar están disponibles dentro del archivo incluido. */
// No funciona; file.txt no puede ser manejado por www.example.com como PHP
include 'http://www.example.com/file.txt?foo=1&bar=2';
// No funciona; busca por un archivo llamado 'file.php?foo=1&bar=2' en el
// sistema de archivos local.
```

```
include 'file.php?foo=1&bar=2';
// Si funciona.
include 'http://www.example.com/file.php?foo=1&bar=2';
$foo = 1;
$bar = 2;
include 'file.txt';  // Funciona.
include 'file.php';  // Funciona.
?>
```

Advertencia de seguridad

El archivo remoto puede ser procesado en el servidor remoto (dependiendo de la extensión del archivo y del hecho de si el servidor remoto corre PHP o no) pero aun así tiene que producir un script PHP válido, porque será procesado en el servidor local. Si el archivo desde el servidor remoto debe ser procesado allá y entregar la salida solamente, readfile ( ) es la mejor función para usar. De lo contrario, debe tenerse especial cuidado para asegurar que el script remoto produce un código válido y deseado.

Manejando retornos: *include* devuelve *FALSE* en caso de falla y eleva una advertencia. Inclusiones exitosas, a menos que sea reemplazado por el archivo incluido, devolverá *1*. Es posible ejecutar una sentencia return dentro de un archivo incluido con el fin de terminar el procesamiento en ese archivo y volver a script que lo llamó. Además, es posible retornar valores desde los archivos incluidos. Se puede tomar el valor de la llamada "include" de la misma forma como se haría con una función normal. Esto no es, sin embargo, posible si se incluyen archivos remotos, a menos que la salida del archivo remoto tenga unas etiquetas válidas de inicio y terminación de PHP (igual que con cualquier archivo local). Se pueden declarar las variables necesarias dentro de esas etiquetas y serán introducidas en cualquiera, así sea el punto del archivo en el cual fue incluido.

Debido a que *include* es un constructor especial del lenguaje, los paréntesis no son necesarios en torno a su argumento. Se debe tener cuidado cuando se compara el valor de retorno.

Ejemplo # 4 Comparando el valor de retorno de include

```
<?php
// no funciona, evaluado como include (('vars.php') == 'OK'), es decir include ('')
if (include ('vars.php') == 'OK') {
    echo 'OK';
}
// si funciona
if ((include 'vars.php') == 'OK') {
    echo 'OK';
}
?>
```

Ejemplo # 5 ***include*** **y la sentencia return**

return.php

```
<?php
$var = 'PHP';
return $var;
?>
```

noreturn.php

```
<?php
$var = 'PHP';
?>
```

testreturns.php

```
<?php
```

```
$foo = include 'return.php';
echo $foo; // muestra 'PHP'
$bar = include 'noreturn.php';
echo $bar; // muestra 1
?>
```

$bar tiene el valor *1* debido a que el include fue exitoso. Nótese la diferencia entre los ejemplos anteriores. El primero usa return dentro del archivo incluido, mientras que el otro no. Si el archivo no se pueden incluir, se retorna FALSE y se emite un E_WARNING.

Si hay funciones definidas en el archivo incluido, se pueden utilizar en el archivo principal independientemente que hayan return antes o después. Si el archivo se incluye dos veces, PHP 5 arrojará un error fatal ya que las funciones ya han sido declaradas, mientras que PHP 4 no se queja acerca de las funciones definidas después de un return. Se recomienda el uso de include_once en lugar de comprobar si el archivo ya estaba incluido y hacer el retorno de forma condicionada dentro del archivo incluido.

Otra forma de "incluir" un archivo PHP en una variable es capturar la salida mediante el uso de Funciones de control de salida con *include*. Por ejemplo:

Ejemplo # 6 Usando buffering de salida para incluir un archivo PHP dentro de una cadena

```
<?php
$string = get_include_contents ('somefile.php');
function get_include_contents ($filename) {
   if (is_file ($filename)) {
      ob_start ( );
```

```
        include $filename;
        return ob_get_clean ( );
    }
    return false;
}
?>
```

Nota.- Puesto que esto es una construcción del lenguaje y no una función, no puede ser llamada usando funciones variables.

REQUIRE_ONCE

La sentencia *require_once* es idéntica a require excepto que PHP verificará si el archivo ya ha sido incluido y si es así, no se incluye (require) de nuevo.

INCLUDE_ONCE

La sentencia *include_once* incluye y evalúa el fichero especificado durante la ejecución del script. Es un comportamiento similar al de la sentencia include, siendo la única diferencia que si el código del fichero ya ha sido incluido, no se volverá a incluir. Como su nombre lo indica, será incluido sólo una vez.

include_once puede ser usado en casos donde el mismo fichero podría ser incluido y evaluado más de una vez durante una ejecución particular de un script, así que en este caso, puede ayudar a evitar problemas como la redefinición de funciones, reasignación de valores de variables, etc.

Nota

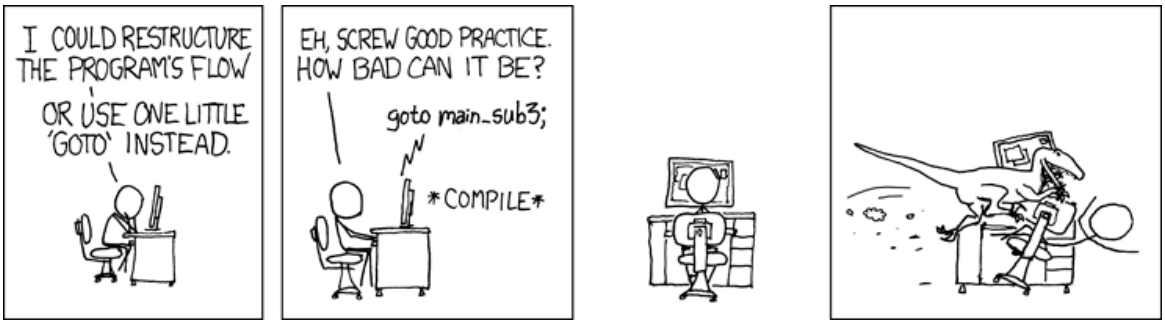
Con PHP 4, la funcionalidad *_once* difiere entre sistemas operativos insensibles a mayúsculas y minúsculas (como Windows) así que por ejemplo:

Ejemplo # 1 *include_once* **con un SO insensible a mayúsculas y minúsculas en PHP 4**

```
<?php
include_once "a.php"; // esto incluirá a.php
include_once "A.php"; // esto incluirá a.php otra vez! (sólo PHP 4)
?>
```

Este comportamiento cambió en PHP 5, así que por ejemplo con Windows primero se normaliza la ruta de acceso para que C: \PROGRA~1\a.php sea identificado igual que C: \Program Files\a.php y el fichero sea incluido sólo una vez.

GOTO



El operador *goto* puede ser usado para saltar a otra sección en el programa. El punto de destino es especificado mediante una etiqueta seguida de dos puntos y la instrucción es dada como *goto* seguida de la etiqueta del destino deseado. Este *goto* no es completamente sin restricciones. La etiqueta de destino debe estar dentro del mismo fichero y contexto, lo que significa que no se puede saltar fuera de una función o método, ni se puede saltar dentro de uno. Tampoco se puede saltar dentro de cualquier clase de estructura de bucle o switch. Se puede saltar fuera de estos y un uso común es utilizar un *goto* en lugar de un *break* multi-nivel.

Ejemplo # 1 Ejemplo de goto

```
<?php
goto a;
echo 'Foo';
a:
echo 'Bar';
?>
```

El resultado del ejemplo sería:

```
Bar
```

Ejemplo # 2 Ejemplo de goto **en un bucle**

```
<?php
for ($i=0,$j=50; $i<100; $i++) {
 while ($j--) {
  if ($j==17) goto end;
  }
}
echo "i = $i";
end:
echo 'j alcanzó 17';
?>
```

El resultado del ejemplo sería:

```
j alcanzó 17
```

Ejemplo # 3 Esto no funcionará

```
<?php
goto loop;
for ($i=0,$j=50; $i<100; $i++) {
while ($j--) {
    loop:
  }
}
echo "$i = $i";
?>
```

El resultado del ejemplo sería:

Fatal error: 'goto' into loop or switch statement is disallowed in script on line 2
(Error fatal: 'goto' hacia el interior de un bucle o sentencia switch no esta permitido en el script en la línea 2).

Nota

El operador goto está disponible a partir de PHP 5.3

EJERCICIOS CON SENTENCIAS DE CONTROL IF, FOR Y WHILE

EJERCICIOS CON CICLO IF

1. Obtener las raíces de una ecuación cuadrática.
2. Dados 5 números, decir cuál es el mayor.
3. Dados 5 números, decir cuál es el menor.

EJERCICIOS CON CICLO FOR

1. Imprimir en pantalla los números del 1 al 10, en lista.
2. Imprimir los números del 1 al 10, en un mismo renglón.
3. Imprimir en pantalla 100 múltiplos de 5.
4. Calcular el factorial de un número.
5. Calcular ∑i, dado el valor de n.
6. Calcular ∑1/i, dado el valor de n.
7. Calcular $\sum (-1)^n$ i, dado el valor de n.
8. Calcular $\sum (-1)^n$ 1/i, dado el valor de n.
9. Dada una cantidad de pesos y un interés dado. Calcular el interés pagado en un año si la reinversión es mensual y diario.
10. Dado el valor de n, calcular 1+3/5+5/7+9/11+…
11. Dado el valor de n, calcular 1-3/5+5/7-9/11+…
12. Dado el valor de n, calcular 3/7+6/14+18/42+72/168+360/840+2160/5040…
13. Calcular $\sum ((2*i)^i)/i$, dado el valor de n.

EJERCICIOS CON CICLO WHILE

Los mismos que el ciclo for.

SOLUCIONES A LOS EJERCICIOS PROPUESTOS

EJERCICIOS SIN CICLOS

1. Crear una página que imprima “Hola”.

```
<?php
        echo "HOLA";
?>
```

2. Dada la longitud de un lado calcular el área y el perímetro de un cuadrado.

```
<?php
        $a=6;
```

```
$c=4;
$area=$a*$a;
$perimetro=$a*$c;
echo"El área es $area, y el perimetro es $perimetro";
?>
```

3. Dada la longitud de un lado y de la altura calcular el área y el perímetro de un triángulo.

```
<?php
$a=10;
$e=2;
$c=$a/$e;
$d=3;
$area=sqrt(($a*$a)-($c*$c));
$perimetro=$a*$d;
echo"El área es $area, y el perimetro es $perimetro";
?>
```

4. Obtener el valor de e, dados c y m de $e=m*c^2$.

```
<?php
$m=9;
$c=3;
$e=($m)*($c*$c);
echo "Si c=$c y m=$m entonces e=$e";
?>
```

5. Dados el radio y la altura de un cilindro, obtener su volumen.

```
<?php
$r=5;
$h=15;
$p=3.1416;
$v=($p*($r*$r))*$h;
echo "Sea el radio $r, la altura $h, entonces elvolumen del cilindro es
$v";
```

```
?>
```

6. Dado un número obtener x^2, x^3 y x^{-1}.

```
<?php
        $x=18;
        $c=$x*$x;
        $r=$x*$x*$x;
        $u=1/$x;
        echo "Dado el numero 18; el cuadrado es $c el cubo es $r y 1/x=$u";
?>
```

7. Crear la tabla del 2, en lista.

```
<?php
        echo "2*1=2";
?>
<br/>
<?php
        echo "2*2=4";
?>
<br/>
<?php
        echo "2*3=6";
?>
<br/>
<?php
        echo "2*4=8";
?>
<br/>
<?php
        echo "2*5=10";
?>
<br/>
<?php
```

```
        echo "2*6=12";
?>
<br/>
<?php
        echo "2*7=14";
?>
<br/>
<?php
        echo "2*8=16";
?>
<br/>
<?php
        echo "2*9=18";
?>
<br/>
<?php
        echo "2*10=20";
?>
```

8. Dados 5 números, obtener la suma de ellos.

```
<?php
        $a=2;
        $b=8;
        $c=1;
        $d=9;
        $e=50;
        $f=$a+$b+$c+$d+$e;
        echo "$a+$b+$c+$d+$e=$f";
?>
```

9. Obtener la media de 10 números.

```
<?php
        $a=12;
```

```
$b=8;
$c=11;
$d=9;
$e=30;
$g=4;
$h=6;
$i=7;
$j=3;
$k=10;
$m=10;
$f=($a+$b+$c+$d+$e+$g+$h+$i+$j+$k)/$m;
echo "La media de $a,$b,$c,$d,$e,$g,$h,$i,$j,$k es $f";
?>
```

10. Dados dos puntos en $\mathbb{R}^2$ calcular su distancia.

```
<?php
$x1=5;
$x2=2;
$y1=7;
$y2=3;
$e=2;
$d1=pow($x1-$x2,$e);
$d2=pow($y1-$y2,$e);
$d=sqrt($d1+$d2);
echo "La distancia entre los puntos (5,7) y (2,3),es $d";
?>
```

11. Dada la longitud de un lado de la base y la altura, calcular el volumen de una pirámide.

```
<?php
$b=5;
$h=15;
$a=$b*$b;
```

```
        $v=($a*$h)/3;
        echo "El volumen de la pirámide de base $b y altura $h es $v";
?>
```

12. Dada la longitud del radio y la altura, calcular el volumen de un cono.

```
<?php
        $r=5;
        $h=12;
        $p=3.1416;
        $v=($p*($r*$r)*$h)/3;
        echo "El volumen de un cono de radio $r y altura $h es $v";
?>
```

13. Convertir de grados Fahrenheit a grados Centígrados.
14. Convertir de grados Centígrados a Fahrenheit.
15. Convertir de metros a pulgadas.
16. Convertir de pulgadas a metros.
17. Convertir de litros a galones.
18. Convertir de galones a litros.
19. Obtener las raíces de una ecuación cuadrática.

```
<?php
        $a=1;
        $b=-1;
        $c=-20;
        $f=sqrt(($b*$b)-(4*$a*$c));
        $x1=(-$b+$f)/(2*$a);
        $x2=(-$b-$f)/(2*$a);
        echo "Dada la ecuación $a x^2 $b x $c=0, los valores de x son x=
        $x1 y x= $x2";
?>
```

20. Calcular la pendiente de una recta dados dos puntos.

```
<?php
        $x1=5;
```

```
        $y1=10;
        $x2=2;
        $y2=7;
        $p=($y1-$y2)/($x1-$x2);
        echo "Dados los puntos ($x1,$y1) y ($x2,$y2); la pendiente es $p";
?>
```

21. Dado el número 5, obtener los primeros diez números impares a partir de él.

```
<?php
        echo "Dado el numero 5, los primeros 10 num impares son:"
?></br>
<?php
        $a=5;
        $c=2;
        $b=$a+($c*1);
        echo "$b";
?></br>
<?php
        $a=5;
        $c=2;
        $b=$a+($c*2);
        echo "$b";
?></br>
<?php
        $a=5;
        $c=2;
        $b=$a+($c*3);
        echo "$b";
?></br>
<?php
        $a=5;
```

```
        $c=2;
        $b=$a+($c*4);
        echo "$b";
?></br>
<?php
        $a=5;
        $c=2;
        $b=$a+($c*5);
        echo "$b";
?></br>
<?php
        $a=5;
        $c=2;
        $b=$a+($c*6);
        echo "$b";
?></br>
<?php
        $a=5;
        $c=2;
        $b=$a+($c*7);
        echo "$b";
?></br>
<?php
        $a=5;
        $c=2;
        $b=$a+($c*8);
        echo "$b";
?></br>
<?php
        $a=5;
        $c=2;
```

```
        $b=$a+($c*9);
        echo "$b";
?></br>
<?php
        $a=5;
        $c=2;
        $b=$a+($c*10);
        echo "$b";
?></br>
```

22. Dado el número 2, obtener los primeros 10 números pares a partir de él.

```
<?php
        echo "Dado el numero 2, los primeros 10 num pares son:"
?></br>
<?php
        $a=2;
        $c=2;
        $b=$a+($c*1);
        echo "$b";
?></br>
<?php
        $a=2;
        $c=2;
        $b=$a+($c*2);
        echo "$b";
?></br>
<?php
        $a=2;
        $c=2;
        $b=$a+($c*3);
        echo "$b";
?></br>
```

```
<?php
        $a=2;
        $c=2;
        $b=$a+($c*4);
        echo "$b";
?></br>
<?php
        $a=2;
        $c=2;
        $b=$a+($c*5);
        echo "$b";
?></br>
<?php
        $a=2;
        $c=2;
        $b=$a+($c*6);
        echo "$b";
?></br>
<?php
        $a=2;
        $c=2;
        $b=$a+($c*7);
        echo "$b";
?></br>
<?php
        $a=2;
        $c=2;
        $b=$a+($c*8);
        echo "$b";
?></br>
<?php
```

```
        $a=2;
        $c=2;
        $b=$a+($c*9);
        echo "$b";
?></br>
<?php
        $a=2;
        $c=2;
        $b=$a+($c*10);
        echo "$b";
?></br>
```

23. Dado un número sacar su factorial.

```
<?php
        $a=5;
        $f=($a)*($a-1)*($a-2)*($a-3)*($a-4);
        echo "Dado el numero 5, su factorial es $f"
?>
```

24. Dada una cantidad de pesos y un interés dado. Calcular el interés pagado en un año si la reinversión es mensual y diario.

```
<?php
        $x=1000;
        $t=12/100;
        $u=1/100;
?>
<br/>
<?php
        $i=($x*$t)+$x;
        echo "anual=$i";
?>
<br/>
<?php
```

```
        $a=($x*$u)+$x;
        $b=($a*$u)+$a;
        $c=($b*$u)+$b;
        $d=($c*$u)+$c;
        $e=($d*$u)+$d;
        $f=($e*$u)+$e;
        $g=($f*$u)+$f;
        $h=($g*$u)+$g;
        $j=($h*$u)+$h;
        $k=($j*$u)+$j;
        $l=($k*$u)+$k;
        $m=($l*$u)+$l;
        echo "total anual con interes mensual=$m";
?>
```

EJERCICIOS CON CICLO IF

1. Obtener las raíces de una ecuación cuadrática.

```
<?php
        $a=5;
        $b=-20;
        $c=29;
        $g=($b*$b)-(4*$a*$c);
if ($g<0)
{
        $f=sqrt(-$g)/(2*$a);
        $x1=(-$b)/(2*$a);
        $x2=(-$b)/(2*$a);
        echo "Las raíces son x1= $x1 + $f i y x2= $x2 - $f i";
}
else
```

```
{
        $f=sqrt($g);
        $x1=(-$b+$f)/(2*$a);
        $x2=(-$b-$f)/(2*$a);
        echo "Dada la ecuación $a x^2 $b x $c=0, los valores de x son x=
$x1 y x= $x2";
}
?>
```

2. Dados 5 números, decir cuál es el mayor.

```
<?php
        $a=9;
        $b=95;
        $c=52;
        $d=800;
        $e=10;
if ($a>$b) {
        if ($a>$c) {
                if ($a>$d) {
                        if ($a>$e)
                        {
                                echo "El mayor es $a";
                        }
                        else
                        {
                        echo "El mayor es $e";
                        }
                }
                else
                {
                        if ($d>$e)
                        {
```

```
                        echo "El mayor es $d";
                    }
                    else
                    {
                        echo "El mayor es $e";
                    }
                }
            }
            else
            {
                if ($c>$d) {
                    if ($c>$e) {
                        echo "El mayor es $c";
                    }
                    else {
                        echo "El mayor es $e";
                    }
                }
                else {
                    if ($d>$e) {
                        echo "El mayor es $d";
                    }
                    else {
                        echo "El mayor es $e";
                    }
                }
            }
        }
        else {
            if ($b>$c) {
                if ($b>$d) {
```

```
                if ($b>$e) {
                        echo "El mayor es $b ";
                }
                else {
                        echo "El mayor es $e ";
                }
        }
        else {
                if ($d>$e) {
                        echo "El mayor es $d";
                }
                else {
                        echo "El mayor es $e";
                }
        }
}
else {
        if ($c>$d) {
                if ($c>$e) {
                        echo "El mayor es $c ";
                }
                else {
                        echo "El mayor es $e ";
                }
        }
        else {
                if ($d>$e) {
                        echo "El mayor es $d";
                }
                else {
                        echo "El mayor es $e";
```

```
                }
            }
        }
    }
    ?>
```

3. Dados 5 números, decir cuál es el menor.

```
<?php
        $a=9;
        $b=95;
        $c=52;
        $d=800;
        $e=10;
    if ($a<$b) {
        if ($a<$c) {
            if ($a<$d) {
                if ($a<$e)
                {
                    echo "El menor es $a";
                }
                else
                {
                echo "El menor es $e";
                }
            }
            else
            {
                if ($d<$e)
                {
                    echo "El menor es $d";
                }
                else
```

```
                {
                        echo "El menor es $e";
                }
        }
        else
        {
                if ($c<$d) {
                        if ($c<$e) {
                                echo "El menor es $c ";
                        }
                        else {
                                echo "El menor es $e";
                        }
                }
                else {
                        if ($d<$e) {
                                echo "El menor es $d";
                        }
                        else {
                                echo "El menor es $e";
                        }
                }
        }
}
else {
        if ($b<$c) {
                if ($b<$d) {
                        if ($b<$e) {
                                echo "El menor es $b ";
                        }
```

```
                else {
                        echo "El menor es $e ";
                }
        }
        else {
                if ($d<$e) {
                        echo "El menor es $d";
                }
                else {
                        echo "El menor es $e";
                }
        }
}
else {
        if ($c<$d) {
                if ($c<$e) {
                        echo "El menor es $c ";
                }
                else {
                        echo "El menor es $e ";
                }
        }
        else {
                if ($d<$e) {
                        echo "El menor es $d";
                }
                else {
                        echo "El menor es $e";
                }
        }
}
```

```
}
?>
```

EJERCICIOS CON CICLO FOR

1. Imprimir en pantalla los números del 1 al 10, en lista.

```
<?php
        for ($x=1;$x<=10;$x=$x+1)
        {echo $x."<br>";};
?>
```

2. Imprimir los números del 1 al 10, en un mismo renglón.

```
<?php
        for ($x=1;$x<=10;$x=$x+1)
        {echo " $x";};
?>
```

3. Imprimir en pantalla 100 múltiplos de 5.

```
<?php
        for ($x=5;$x<=500;$x=$x+5)
        {echo $x."<br>";};
?>
```

4. Calcular el factorial de un número.

```
<?php
        $a=5;
        $c=1;
        for ($x=1;$x<=$a;$x=$x+1)
        {
        $c=$c*$x;};
        echo "El factorial de 5 es $c";
?>
```

5. Calcular ∑i, dado el valor de n.

```
<?php
        $n=100;
        $c=0;
        for ($x=1;$x<=$n;$x=$x+1)
        {
        $c=$c+$x;};
        echo "Si n = $n la suma es $c";
?>
```

6. Calcular ∑1/i, dado el valor de n.

```
<?php
        $n=100;
        $c=0;
        for ($x=1;$x<=$n;$x=$x+1)
        {
        $c=$c+(1/$x);};
        echo "Si n = $n la sumatoria es $c";
?>
```

7. Calcular $\sum (-1)^n$ i, dado el valor de n.

```
<?php
        $n=100;
        $b=-1;
        $c=0;
        for ($x=1;$x<=$n;$x=$x+1)
        {
        $c=$c+((pow($b,$x))*($x));};
        echo "Si n = $n la sumatoria es $c";
?>
```

8. Calcular $\sum (-1)^n$ 1/i, dado el valor de n.

```
<?php
        $n=5;
        $b=-1;
```

```
$c=0;
for ($x=1;$x<=$n;$x=$x+1)
{
$c=$c+((pow($b,$x))*(1/$x));};
echo "Si n = $n la sumatoria es $c";
?>
```

9. Dada una cantidad de pesos y un interés dado. Calcular el interés pagado en un año si la reinversión es mensual y diario.

```
<?php
$inversion_inicial=1000;
$inversion1=1000;
$inversion2=1000;
$interes_mensual=8/100;
$interes_diario=(8/100)/(30.416667);
$meses=12;
$dias=365;
$cantmensual=0;
for ($i=1;$i<=$meses;$i++){
$inversion1=$inversion1+($inversion1*$interes_mensual);
};
echo"Se invirtieron  $inversion_inicial , la cantidad despúes de
$meses meses es $inversion1";
echo "<br>";
for ($k=1;$k<=$dias;$k++) {
$inversion2=$inversion2+($inversion2*$interes_diario);
};
echo"Se invirtieron  $inversion_inicial ,  la cantidad despúes de $dias
dias es $inversion2";
?>
```

10. Dado el valor de n, calcular 1+3/5+5/7+9/11+…

```
<?php
```

```
$n=3;
$suma=1;
for ($i=1;$i<=$n;$i++) {
$variable=pow(2,$i);
$suma=$suma+((1+$variable)/(3+$variable));
}
echo "En la serie tenemos n= $n , entonces la suma es $suma";
?>
```

11. Dado el valor de n, calcular 1-3/5+5/7-9/11+…

```
<?php
$n=3;
$suma=1;
for ($i=1;$i<=$n;$i++){
$potencia=pow (-1,$i);
$variable=pow (2,$i);
$suma=$suma+(($potencia)*((1+$variable)/(3+$variable)));
}
echo "En la serie tenemos n= $n , entonces la suma es $suma";
?>
```

12. Dado el valor de n, calcular
    3/7+6/14+18/42+72/168+360/840+2160/5040…

```
<?php
$n=6;
$arriba=3;
$abajo=7;
$suma=0;
for ($i=1;$i<=$n;$i++)
{
$arriba=$arriba*$i;
$abajo=$abajo*$i;
$suma=$suma+($arriba/$abajo);;
```

```
        };
        echo "En la serie tenemos n= $n , entonces la suma es $suma";
?>
```

13. Calcular $\sum ((2*i)^{i})/i$, dado el valor de n.

```
<?php
        $n=3;
        $suma=0;
        for ($i=1;$i<=$n;$i++)
        {
        $potencia=pow (2*$i,$i);
        $suma=$suma+($potencia/$i);
        };
        echo "En la serie tenemos n= $n , entonces la suma es $suma";
?>
```

EJERCICIOS CON CICLO WHILE

1. Imprimir en pantalla los números del 1 al 10, en lista.

```
<?php
        while ($count<=10) {
        echo $count . "<br/>";
        $count++;
        }
?>
```

2. Imprimir los números del 1 al 10, en un mismo renglón.

```
<?php
        while ($count<=10) {
        echo $count." ";
        $count++;
        }
?>
```

3. Imprimir en pantalla 100 múltiplos de 5.

```
<?php
        while ($count<=100) {
        echo $count."<br/>";
        $count=$count+5;
        }
?>
```

4. Calcular el factorial de un número.

```
<?php
        $a=10;
        $c=1;
        while ($count<=$a) {
        $count++;
        $c=$c*$count;
        }
        echo $c;
?>
```

5. Calcular ∑i, dado el valor de n.

```
<?php
        $a=10;
        $c=0;
        while ($count<=$a) {
        $c=$c+$count;
        $count++;
        }
        echo $c;
?>
```

6. Calcular ∑1/i, dado el valor de n.

```
<?php
        $a=10;
        $c=0;
```

```
while ($count<=$a)
{
$c=$c+(1/$count);
$count++;
}
echo $c;
?>
```

7. Calcular $\sum (-1)^n$ i, dado el valor de n.

```
<?php
$a=10;
$c=0;
$b=-1;
while ($count<=$a) {
$c=$c+((pow ($b,$count))*$count);
$count++;
}
echo $c;
?>
```

8. Calcular $\sum (-1)^n$ 1/i, dado el valor de n.

```
<?php
$a=10;
$c=0;
$b=-1;
while ($count<=$a) {
$c=$c+((pow($b,$count))*(1/$count));
$count++;
}
echo $c;
?>
```

9. Dado el valor de n, calcular 1+3/5+5/7+9/11+…

```
<?php
```

```
        $n=3;
        $c=1;
        while ($count<=$n) {
        $variable=pow (2,$count);
        $c=$c+((1+$variable)/(3+$variable));
        $count++;
        }
        echo $c;
?>
```

10. Dado el valor de n, calcular 1-3/5+5/7-9/11+…

```
<?php
        $n=3;
        $c=1;
        while($count<=$n){
        $potencia=pow (-1,$count);
        $variable=pow (2,$count);
        $c=$c+(($potencia)*((1+$variable)/(3+$variable)));
        $count++;
        }
        echo $c;
?>
```

11. Dado el valor de n, calcular 3/7+6/14+18/42+72/168+360/840+2160/5040…

```
<?php
        $n=3;
        $arriba=3;
        $abajo=7;
        $c=0;
        $count=1;
        while ($count<=$n) {
        $arriba=$arriba*$count;
```

```
        $abajo=$abajo*$count;
        $c=$c+($arriba/$abajo);
        $count++;
        }
        echo $c;
?>
```

12. Calcular $\sum ((2*i)^{i})/i$, dado el valor de n.

```
<?php
        $n=3;
        $c=0;
        $count=1;
        while ($count<=$n) {
        $potencia=pow (2*$count,$count);
        $c=$c+($potencia/$count);
        $count++;
        }
        echo $c;
?>
```

FUNCIONES EN PHP

FUNCIÓN STR_REPLACE

Esta función se utiliza para reemplazar caracteres dentro de una cadena de caracteres. Es decir, nos permite definir una cadena que debe ser reemplazada con otra. Veámoslo con un ejemplo para entenderla mejor.

Escribe este código y guárdalo con un nombre de archivo como ejemplo1.php. A continuación, sube el fichero al servidor y visualiza el resultado.

```
<?php //Ejemplo funciones básicas aprenderaprogramar.com
$texto = "Donde dije digo digo Diego.";
echo str_replace ("Diego", "recortes", $texto);
echo "<br />";
echo $texto;
?>
```

Como podemos observar, en la cadena de caracteres $texto, hemos sustituido la palabra o conjunto de caracteres "Diego" por "Recortes" y hemos devuelto el resultado sin modificar la variable de entrada.

La sintaxis general para esta función es:

str_replace (“cadena a buscar”, “cadena de reemplazo”, $variableString)

Fíjate que en el código que hemos escrito no hemos reemplazado el valor de la variable $texto por un nuevo contenido. Unicamente hemos impreso por pantalla el resultado que nos devuelve la función. Si hubiéramos escrito lo siguiente sí hubiéramos cambiado el contenido de la variable: $texto = str_replace ("Diego", "recortes", $texto).

FUNCIONES TIME Y DATE

Estas dos funciones nos permitirán mostrar o capturar la fecha y hora, lo cual tiene una gran cantidad de aplicaciones. Por ejemplo, supón que tienes una tienda on-line y para cada operación de venta tienes que registrar la fecha y hora de la venta. Estas funciones resultarán útiles para ello.

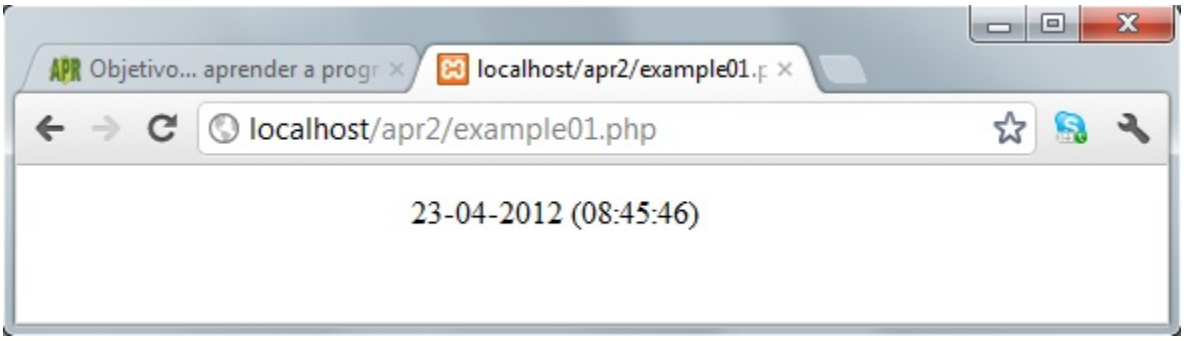
La función time devuelve el momento actual medido como el número de segundos desde el 1 de Enero de 1970 00:00:00 GMT. Cuando hablamos de momento actual nos referimos a la "hora actual del servidor". Hay que tener cuidado con esta circunstancia. Por ejemplo, si trabajas con un servidor localizado en Estados Unidos lo más probable es que el servidor trabaje con la hora de Estados Unidos. Si quieres obtener la hora local de tu país, tendrás que tenerlo en cuenta para restarle o sumarle cierto número de horas a la hora que te devuelva el servidor.

Por otro lado, la función date muestra la fecha en formato d-m-Y (donde d representa día, m representa mes y Y representa año) desde el valor de time u otra fecha dada en segundos desde el 1 de Enero de 1970 00:00:00 GMT.

En general, te recomendamos que hagas pruebas para comprobar que los resultados de fecha y hora obtenidos son los adecuados, ya que la configuración de cada servidor es distinta.

Escribe ahora este código y guárdalo con un nombre de archivo como ejemplo2.php. A continuación, sube el fichero al servidor y visualiza el resultado.

```
<?php //Ejemplo funciones básicas aprenderaprogramar.com
$time = time ( );
echo date ("d-m-Y (H:i:s)", $time);
?>
```



Como podemos observar, la función date da formato a la variable en segundos $time. Hay distintos formatos para la función date pero no entraremos a describirlos todos.

La función time ( ) devuelve un valor numérico entero largo, por ejemplo 1335169779. Este número representa el número de segundos transcurridos desde el 1 de enero de 1970.

Para transformar ese número en una fecha “entendible por las personas” usamos la función date, cuya sintaxis general es: date (“formato de salida”, valorTimeValido).

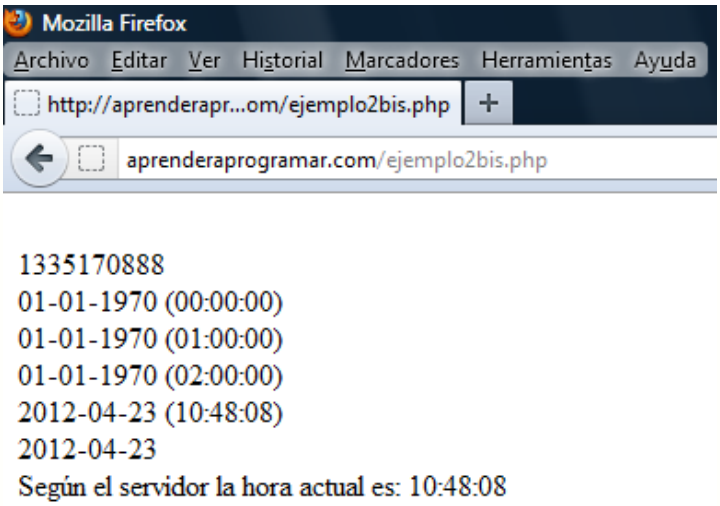
En cuanto a “formato de salida”, disponemos de las siguientes equivalencias:

d: representa el día
m: representa el mes
Y: representa el año
H: representa la hora (dos dígitos)
i: representa los minutos (dos dígitos)
s: representa los segundos (dos dígitos)

En cuanto a valorTimeValido, será un número, generalmente contenido en una variable.

Escribe ahora este código y guárdalo con un nombre de archivo como ejemplo 2bis.php. A continuación, sube el fichero al servidor y visualiza el resultado.

```
<?php //Ejemplo funciones básicas aprenderaprogramar.com
$time = time ( );
echo "<br/>";
echo $time;
echo "<br/>";
echo date ("d-m-Y (H:i:s)", -3600);
echo "<br/>";
echo date ("d-m-Y (H:i:s)", 0);
echo "<br/>";
echo date ("d-m-Y (H:i:s)", 3600);
echo "<br/>";
echo date ("Y-m-d (H:i:s)", $time);
echo "<br/>";
echo date ("Y-m-d ", $time);
echo "<br/>";
echo ("Según el servidor la hora actual es: ". date ("H:i:s", $time));
?>
```



Con este ejemplo podemos comprobar algunas cosas. En primer lugar, que la función time ( ) devuelve un número, mientras que la función date devuelve un String o cadena de texto.

En segundo lugar, que para obtener fechas anteriores al 1 de enero de 1970 podemos usar números negativos.

En tercer lugar, que el momento 0 que es el 01-01-1970 00:00:00 GMT, cuando es mostrado por un servidor concreto, es adaptado a su horario local. En el servidor del ejemplo, el horario local es GMT + 01:00, es decir, su hora es la hora GMT +01:00, por eso nos devuelve como hora cero la una de la mañana del 1 de enero de 1970. No te preocupes si te resulta un poco confuso, simplemente ten en cuenta que debes hacer pruebas para ver cómo responde el servidor con el que estés trabajando.

En cuarto lugar que podemos hacer que la fecha se muestre en el formato u orden que queramos, usando los términos clave “ d, m, Y, H, i, s” en el orden que queramos.

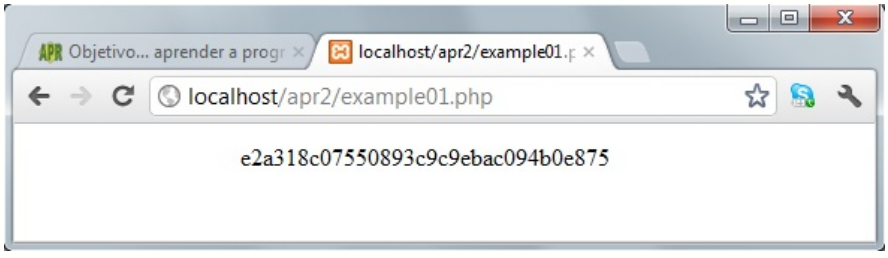
En quinto lugar, que podemos concatenar cómo se muestra una fecha (o almacenarla en otra variable si quisiéramos) concatenando el resultado de la función date con una cadena de texto cualquiera.

FUNCIÓN MD5

La función md5 es utilizada para encriptar contraseñas. Se llama encriptar a convertir una palabra o cadena de caracteres en un conjunto de letras y números aparentemente aleatorios. Para guardar contraseñas es recomendable usar esta función por motivos de seguridad. Veamos un ejemplo.

Escribe este código y guárdalo con un nombre de archivo como ejemplo 3.php. A continuación, sube el fichero al servidor y visualiza el resultado.

```
<?php //Ejemplo funciones básicas aprenderaprogramar.com
$password = "micontraseña";
echo md5 ($password);
?>
```



Observamos que la función devuelve la cadena de caracteres "micontraseña" con el encriptado md5, de forma que una persona que vea este conjunto de letras y números no sabe a qué palabra o conjunto de caracteres equivale.

Imagínate que un hacker consiguiera acceder a un listado de contraseñas de usuarios de tu página web. Si las tienes encriptadas, no podrá hacer uso de ellas. Si no las tienes encriptadas, las podrá robar y usar. La mayoría de aplicaciones web (como Joomla, Drupal, phpBB, etc.) usan sistemas de encriptación de modo que las contraseñas de los usuarios no son visibles para nadie, ni siquiera para los propios administradores de las páginas web.

Cuando una persona introduce una contraseña en una página web que usa encriptado md5, el sistema compara el encriptado de esa contraseña con el md5 almacenado. Si coincide, se permite el acceso. Si no coincide, se deniega.

FUNCIONES STRTOLOWER Y STRTOUPPER

Las funciones strtolower y strtoupper transforman una cadena de caracteres en la misma cadena en minúsculas o mayúsculas respectivamente.

Escribe ahora este código y guárdalo con un nombre de archivo como ejemplo 4.php. A continuación, sube el fichero al servidor y visualiza el resultado.

```
<?php //Ejemplo funciones básicas aprenderaprogramar.com
$cadena = "EstO eS UnA cadeNA de CARActeres";
echo strtolower ($cadena);
echo "<br />";
echo strtoupper ($cadena);
?>
```

Como vemos el comportamiento de las funciones es sencillo. Simplemente devuelven la cadena de caracteres pasada como argumento en minúsculas o mayúsculas respectivamente.

Ten en cuenta que muchas veces para realizar comparaciones o para almacenar datos será interesante uniformizar la información que se almacena. Por ejemplo, si se pide una ciudad, un usuario puede introducir Buenos Aires, otro Buenos aires, otro BUENOS AIRES. Si usamos estas funciones, podemos uniformizar y hacer que siempre se muestren o guarden de una misma manera, lo cual facilitará el trabajo posterior.

ABS

Abs Valor absoluto

Descripción

number abs (mixed $number)
Devuelve el valor absoluto de number.

Parámetros

Number-Valor decimal a convertir.

Valores devueltos

El valor absoluto de number. Si el argumento number es del tipo float, el valor devuelto es también del tipo float, siendo para los otros casos del tipointeger (debido a que float es normalmente de mayor rango de valores queinteger).

Ejemplo de abs ( )

```
<?php
$abs = abs (-4.2); // $abs = 4.2; (double/float)
$abs2 = abs (5);   // $abs2 = 5; (integer)
$abs3 = abs (-5);  // $abs3 = 5; (integer)
?>
```

ACOS

Acos – Arco coseno

Descripción

float acos (float $arg).

Devuelve el arco coseno de arg en radianes. acos ( ) es la función complementaria de cos ( ), lo que quiere decir que a==cos (acos (a)) para cada valor de a en el rango de acos ( )'.

Parámetros

Arg – El argumento a procesar.

EJEMPLO

```
<?php
    $valor=1;
    echo acos ($valor);
?>
```

ACOSH

Acosh – Arco coseno hiperbólico

Descripción

float acosh (float $arg)
Devuelve el arco coseno hiperbólico de arg, es decir, el valor cuyo coseno hiperbólico es arg.

Parámetros

arg – El argumento a procesar

Valores devueltos

El arco coseno hiperbólico de arg

EJEMPLO

```
<?php
    $valor=0;
    echo acosh ($valor);
?>
```

ASIN

asin - Arco seno

Descripción

float asin (float $arg)
Devuelve el arco seno de arg en radiantes. asin ( ) es la función complementaria de sin ( ), lo que quiere decir que a==sin (asin (a)) para cada valor de a en el rango de asin ( ).

Parámetros

arg – El argumento a procesar

EJEMPLO

```
<?php
    $valor=1;
    echo asin ($valor);
?>
```

ASINH

asinh – Arco seno hiperbólico

Descripción

float asinh (float $arg)

Devuelve el arco seno hiperbólico de arg, es decir, el valor cuyo seno hiperbólico es arg.

Parámetros

arg – El argumento a procesar

Valores devueltos

El arco seno hiperbólico de arg

EJEMPLO

```
<?php
    $valor=0;
    echo asinh ($valor);
?>
```

ATAN2

atan2 – Arco tangente de dos variables

Descripción

float atan2 (float $y , float $x)

Esta función calcula el argo tangente de las dos variables x y y. Es similar a calcular el arco tangente de y / x, excepto que los signos de ambos argumentos son usados para determinar el cuadrante del resultado.

La función devuelve el resultado de radiantes, que se encuentra entre –PI y PI (inclusive).

Parámetros

y

Parámetro dividendo

x

Parámetro divisor

Valores devueltos

El arco tangente de y/x en radiantes.

EJEMPLO

```
<?php
    $valor=1;
    $valor2=2
    echo atan2 ($valor,$valor2);
    ?>
```

ATAN

Atan – Arco tangente

Descripción

float atan (float $arg)

Devuelve el arco tangente de arg en radiantes. atan ( ) es la función complementaria de tan ( ), lo que quiere decir que *a==tan (atan (a))* para cada valor de a en el rango de atan ( ).

Parámetros

arg – El argumento a procesar

EJEMPLO

```
<?php
    $valor=1;
    echo atan ($valor);
?>
```

ATANH

atanh – Arco tangente hiperbólica

Descripción

float atanh (float $arg)
Devuelve el arco tangente hiperbólico de arg, es decir, el valor cuya tangente hiperbólica es arg.

Parámetros

arg – El argumento a procesar

Valores devueltos

El arco tangente hiperbólico de arg

EJEMPLO

```
<?php
    $valor=1;
    echo atanh ($valor);
    ?>
```

BASE_CONVERT

base_convert – Convertir un número entre bases arbitrarias.

Descripción

string base_convert (string $number, int $frombase, int $tobase)
Devuelve una cadena que contiene el número dado representado en base a_base. La base en la cual es dado el valor número es especificado en desde_base. Tanto desde_base como a_base tienen que ser valores entre 2 y 36, inclusive. Los dígitos en números con una base mayor que 10 serán representados con las letras a-z, en donde a significa 10, b significa 11 y z significa 35.

Advertencia

base_convert ( ) puede perder precisión en números grandes debido a las propiedades relacionadas al tipo interno "doble" o "flotante" usado. Por favor consulte la sección sobre Números de punto flotante en el manual para más información específica sobre las limitaciones.

Parámetros

number – El número a convertir
frombase – La base en que se encuentra number
tobase – La base a la cual convertir number

Valores devueltos

number convertido a base tobase
Ejemplo de base_convert ( )

```
<?php
$hexadecimal = 'A37334';
echo base_convert ($hexadecimal, 16, 2);
?>
```

El resultado del ejemplo sería:

101000110111001100110100

BINDEC

Bindec – Binario a decimal

Descripción

number bindec (string $binary_string)

Devuelve el equivalente decimal del número binario representado por el argumento binary_string.

bindec ( ) convierte un número binario a un integer o, si es necesario por razones de tamaño, a un float.

bindec ( ) interpreta todos los valores de binary_string como enteros sin signo. Esto es debido a que bindec ( ) considera al bit más significativo como otro orden de magnitud en lugar de como el bit de signo.

Parámetros

binary_string – El string binario a convertir

Advertencia

Este parámetro debe ser un string. Cualquier otro tipo de dato empleado puede producir resultados inesperados.

Valores devueltos

El valor decimal de binary_string
Ejemplo de bindec ( )

```
<?php
echo bindec ('110011') . "\n";
echo bindec ('000110011') . "\n";
echo bindec ('111');
?>
```

El resultado del ejemplo sería:
51
51
7

CEIL

ceil – Redondear fracciones hacia arriba

Descripción

float ceil (float $value)
Devuelve el siguiente valor entero mayor redondeando hacia arriba valuesi fuera necesario.

Parámetros

value – El valor a redondear

Valores devueltos

value redondeado al siguiente entero más alto. El valor de retorno deceil ( ) sigue siendo de tipo float ya que el rango de valores de float es usualmente mayor que el del tipo integer.

Ejemplo de ceil ( )

```
<?php
echo ceil (4.3);    // 5
echo ceil (9.999);  // 10
echo ceil (-3.14);  // -3
?>
```

COS

cos – Coseno

Descripción

float cos (float $arg)

cos ( ) devuelve el coseno del parámetro arg. El parámetro arg está en radianes.

Parámetros

arg – un ángulo en radianes

Valores devueltos

El coseno de arg

Ejemplo de cos ( )

```
<?php
echo cos(M_PI); // -1
?>
```

COSH

cosh – Coseno hiperbólico

Descripción

float cosh (float $arg)

Devuelve el coseno hiperbólico de arg, definido como (exp (arg) + exp (-arg))/2.

Parámetros

arg – El argumento a procesar

Valores devueltos

El coseno hiperbólico de arg

EJEMPLO

```
<?php
    $valor=0;
    echo cosh ($valor);
?>
```

DECBIN

Decbin – Decimal a binario

Descripción

string decbin (int $number)

Devuelve una cadena que contiene una representación binaria del argumento number.

Parámetros

number – Valor decimal a convertir

Rango de entradas en máquinas de 32-bit		
numberpositivo	numbernegativo	valor de retorno
0		0
1		1
2		10
... progresión normal ...		
2147483646		11111111111111111111111111111110
2147483647 (mayor entero con signo)		11111111111111111111111111111111 (31 1's)
2147483648	-2147483648	10000000000000000000000000000000
... progresión normal ...		
4294967294	-2	11111111111111111111111111111110
4294967295 (mayor entero sin signo)	-1	11111111111111111111111111111111 (32 1's)

Rango de entradas en máquinas de 64-bit		
number positivo	number negativo	valor de retorno
0		0
1		1

2		10
... progresión normal ...		
9223372036854775806		1111111111111111111111111111111111111111111111111111111111111110
9223372036854775807 (mayor entero con signo)		1111111111111111111111111111111111111111111111111111111111111111 (63 1's)
	-9223372036854775808	1000000000000000000000000000000000000000000000000000000000000000
... progresión normal ...		
	-2	1111111111111111111111111111111111111111111111111111111111111110
	-1	1111111111111111111111111111111111111111111111111111111111111111 (64 1's)

Valores devueltos

Cadena de la representación binaria de number

Ejemplo de decbin ( )

```
<?php
echo decbin (12) . "\n";
echo decbin (26);
?>
```

El resultado del ejemplo sería:

```
1100
11010
```

DECHEX

dechex – Decimal a hexadecimal

Descripción

string dechex (int $number)

Devuelve una cadena que contiene una representación hexadecimal del argumento number dado.

El mayor número que puede convertirse es PHP_INT_MAX * *2* + *1* (o *-1*): en plataformas de 32 bits, este será *4294967295* en decimal, cuyos resultados utilizando dechex() devolverán *ffffffff.*

Parámetros

Number – Valor decimal a convertir.

Aunque los tipos integer PHP tiene signo, la dechex ( ) los trata como enteros sin signo, de igual forma los enteros negativos serán tratados como si no tuvieran signo.

Valores devueltos

Representación tipo cadena hexadecimal de number.

Ejemplo dechex ( )

```
<?php
echo dechex (10) . "\n";
echo dechex (47);
?>
```

El resultado del ejemplo sería:

```
a
2f
```

DECOCT

dedoct – Decimal a octal

Descripción

string decoct (int $number)
Devuelve una cadena que contiene una representación octal del argumento numero dado. El mayor número que puede convertirse es 4294967295 en decimal, que resulta en "37777777777".

Parámetros

Number – Valor decimal a convertir

Valores devueltos

Representación tipo cadena octal de numero

Ejemplo de decoct ( )

```
<?php
echo decoct (15) . "\n";
echo decoct (264);
?>
```

El resultado del ejemplo sería:

17
410

DEG2RAD

deg2rad – Convierte el número en grados a su equivalente en radianes

Descripción

float deg2rad (float $number)
Esta función convierte a número desde grados a su equivalente en radianes.

Parámetros

number – Valor angular en grados

Valores devueltos

El equivalente en radianes de number

Ejemplo de deg2rad ( )

```
<?php
echo deg2rad (45); // 0.785398163397
var_dump (deg2rad (45) === M_PI_4); // bool (true)
?>
```

EXP

exp – Calcula la exponencial de e

Descripción

float exp (float $arg)
Devuelve e elevado a la potencia de arg.

Nota

'e' es la base del sistema natural de logaritmos, y su valor aproximado es 2.718282.

Parámetros

Arg – El argumento a procesar

Valores devueltos

'e' elevado a la potencia de arg

Ejemplo de exp ( )

```
<?php
echo exp (12) . "\n";
echo exp (5.7);
?>
```

El resultado de ejemplo sería:

```
1.6275E+005
298.87
```

EXPM1

expm1 – Devuelve exp (número) -1, calculado de tal forma que no pierde precisión incluso cuando el valor del número se aproxima a cero.

Descripción

float expm1 (float $arg)
expm1 ( ) devuelve el equivalente a 'exp (arg) - 1' calculado de tal forma que no pierde precisión incluso cuando el valor de arg se aproxima a cero, un caso donde

'exp (arg) - 1' puede ser inexacto debido a la resta de dos numeros que son casi iguales.

Parámetros

arg – El argumento a procesar

Valores devueltos

‘e’ elevado a la potencia de arg menos uno

EJEMPLO

```
<?php
    echo expm1 (25);
?>
```

FLOOR

floor – Redondear fracciones hacia abajo

Descripción

float floor (float $value)
Devuelve el máximo de los enteros menores o iguales, redondeando el valor si es necesario.

Parámetros

value – El valor a redondear

Valores devueltos

value redondeado al anterior entero más bajo. El valor de retorno de floor ( ) sigue siendo de tipo float ya que el rango de valores de float es usualmente mayor que el del tipo integer.

Ejemplo de floor ( )

```
<?php
echo floor (4.3);   // 4
echo floor (9.999); // 9
echo floor (-3.14); // -4
?>
```

FMOD

fmod – Devuelve el residuo de punto flotante (módulo) de la división de los argumentos.

Descripción

float fmod (float $x , float $y)
Devuelve el residuo de punto flotante de dividir el dividendo (x) por el divisor (y). El residuo (r) es definido como: x = i * y + r, para algún entero i. Si y es diferente de cero r tiene el mismo signo que X y una magnitud menor que la magnitud de y.

Parámetros

x

El dividiendo

y

El divisor

Valores devueltos

El residuo en punto flotante de x/y

Ejemplo: Uso de fmod ( )

```
<?php
$x = 5.7;
$y = 1.3;
$r = fmod ($x, $y);
// $r es igual a 0.5, ya que 4 * 1.3 + 0.5 = 5.7
?>
```

GETRANDMAX

getrandmax – Mostrar el mayor valor aleatorio posible

Descripción

int getrandmax (void)

Devuelve el valor máximo que puede ser devuelto por una llamada a rand ( ).

Valores devueltos

El valor aleatorio más grande posible devuelto por rand ( ).

HEXDEC

hexdec – Hexadecimal a decimal

Descripción

number hexdec (string $hex_string)

Devuelve el equivalente decimal del número hexadecimal representado por el argumento cadena_hex. Hexdec ( ) convierte un string hexadecimal a un número decimal.

hexdec ( ) ignoran cualquier caracter no hexadecimal que encuentre.

Parámetros

hex_string- El string hexadecimal a convertir

Valores devueltos

La representación decimal de hex_string

Ejemplo de hexdec ( )

```
<?php
var_dump (hexdec ("See"));
var_dump (hexdec ("ee"));
// ambos imprimen "int (238)"
var_dump (hexdec ("that")); // imprime "int (10)"
var_dump (hexdec ("a0")); // imprime "int (160)"
?>
```

HYPOT

hypot – Calcula la longitud de la hipotenusa de un triángulo de ángulo recto

Descripción

float hypot (float $x, float $y)

hypot ( ) devuelve la longitud de la hipotenusa de un triángulo de ángulo recto con lados de longitud x y y, o la distancia del punto (x, y) a partir del origen. Esto es equivalente a sqrt (x * x + y * y).

Parámetros

x

Longitud del primer lado

y

Longitud del segundo lado

Valores devueltos

La longitud calculada de la hipotenusa

EJEMPLO

```
<?php
    $x=12;
    $y=5;
    echo hypot ($x,$y);
?>
```

IS_FINITE

Is_finite – Encuentra si un valor es un número finito legal

Descripción

bool is_finite (float $val)

Verifica si va es un número finite legal en esta plataforma.

Parámetros

val – El valor a chequear

Valores devueltos

TRUE si val es un número finito legal en el rango permitido para un valor flotante PHP en esta plataforma, FALSE de lo contrario.

IS_INFINITE

is_finite – Encuentra si un valor es infinito

Descripción

bool is_infinite (float $val)

Devuelve TRUE si val es infinito (positivo o negativo), como el resultado de log (0) o cualquier valor demasiado grande para caber en un flotante en esta plataforma.

Parámetros

val - El valor a chequear

Valores devueltos

TRUE si val es infinito, FALSE de lo contrario.

IS_NAN

Is_nan – Encuentra si un valor no es un número

Descripción

bool is_nan (float $val)

Verifica si val no es un número, como por ejemplo el resultado de acos (1.01).

Parámetros

Val – El valor a revisar

Valores devueltos

Devuelve TRUE si val no es un número, en caso contrario devuelve FALSE.

Ejemplo de is_nan ( )

```
<?php
// Invalid calculation, will return a
// NaN value
$nan = acos (8);
var_dump ($nan, is_nan ($nan));
?>
```

El resultado del ejemplo sería:

float (NAN)
bool (true)

ICG_VALUE

icg_value – Generador lineal congruente combinado

Descripción

float lcg_value (void)

icg_value ( ) devuelve un número pseudo-aleatorio en el rango (0, 1). La función combina dos generadores congruentes con periodos de 2^31 - 85 y 2^31 - 249. El periodo de esta función es igual al producto de ambos primos.

Valores devueltos

Un valor flotante pseudo-aleatorio en el rango de (0, 1)

LOG 10

log 10 – Logaritmo en base 10

Descripción

float log10 (float $arg)

Devuelve el logaritmo en base 10 de arg.

Parámetros

arg – El argumento a procesar

Valores devueltos

El logaritmo en base 10 de arg.

EJEMPLO

```
<?php
    echo log 10 100);
?>
```

LOG1P

log1p – Devuelve log (1 + número), calculado de tal forma que no pierde precisión incluso cuando el valor del número se aproxima a cero.

Descripción

float log1p (float $number)

log1p ( ) devuelve el equivalente a log (1 + number) calculado de tal forma que no pierde precision incluso cuando el valor de number se aproxima a cero. log ( ) puede devolver solo log (1) en este caso debido a la falta de precisión.

Parámetros

number – El argumento a procesar

Valores devueltos

log (1 + number)

EJEMPLO

```
    <?php
        echo log1p (100);
?>
```

LOG

log – Logaritmo natural

Descripción

float log (float $arg [, float $base = M_E])

Si se especifica el parámetro opcional base, log ( ) devuelve devuelve $\log_{base}$ arg, y en caso contrario log ( ) devuelve el logaritmo natural de arg.

Parámetros

arg – El valor al que se desea calcular el logaritmo
base – La base opcional del logaritmo a usamr (por omisión es ‘e’ y por lo tanto el logaritmo natural).

Valores devueltos

El logaritmo de arg en la base dada por base, si se indica, o en su lugar el logaritmo natural.

MAX

Max Encontrar el valor más alto

Descripción

mixed max (array $values)
mixed max (mixed $value1, mixed $value2 [, mixed $...])

Si el primer y único parámetro es una matriz, max ( ) devuelve el valor más alto en esa matriz. Si al menos dos parámetros son entregados, max ( ) devuelve el mayor de estos valores.

Nota

PHP evaluará un valor string no-numérico como *0* si se compara con uninteger, pero aun devuelve la cadena si ésta es vista como el valor numérico más alto. Si varios argumentos evalúan a *0*, max ( ) devolverá el valor de cadena alfanumérica más alto si se da alguna cadena, o de lo contrario se devuelve un *0* numérico.

Parámetros

values
: Un array que contiene los valores.

value 1
: Cualquier valor comparable.

value 2
: Cualquier valor comparable.

…..
: Cualquier valor comparable.

Valores devueltos

max ( ) devuelve el mayor valor numérico de los parámetros. Si varios valores pueden considerarse del mismo tamaño, se devolverá el listado en primer lugar.

Cuando max ( ) debe evaluar varios arrays, devuelve el array de mayor tamaño. Si todos los arrays tienen el mismo tamaño, max ( ) empleará la ordenación lexicográfica para encontrar el valor a devolver.

Cuando se debe comparar un string, se le considerará como un integer en la comparación.

Ejemplos de uso de max ( )

```
<?php
echo max (1, 3, 5, 6, 7);  // 7
echo max (array (2, 4, 5)); // 5
// Cuando 'hola' es convertido en integer se convierte en 0. Ambos parámetros son
// del mismo tamaño por lo que el orden en el que son colocados determina el resultado
echo max (0, 'hola');     // 0
echo max ('hola', 0);     // hola
echo max ('42', 3); // '42'
// En este caso 0 > -1, asi que 'hola' es el valor devuelto.
echo max (-1, 'hola');    // hola
// Con varias matrices de diferentes tamaño, max devuelve el de mayor tamaño
$val = max (array (2, 2, 2), array (1, 1, 1, 1)); // array (1, 1, 1, 1)
// Con varias matrices, max las compara de izquierda a derecha
// empleando el orden lexicográfico, así que en nuestro ejemplo: 2 == 2, pero 4 < 5
$val = max (array (2, 4, 8), array (2, 5, 7)); // array (2, 5, 7)
```

```
// Si se pasan una matriz y una no-matriz, la matriz
// es devuelta siempre ya que se considera el valor mayor
$val = max ('cadena', array (2, 5, 7), 42);   // array (2, 5, 7)
?>
```

MIN

Min – Encontrar el valor más bajo

Descripción

mixed min (array $values)
mixed min (mixed $value1 , mixed $value2 [, mixed $...])

Si el primer y único parámetro es un array, min ( ) devuelve el valor más bajo de ese array. Si se proporcionan al menos dos parámetros, min ( ) devuelve el menor de estos valores.

Nota

PHP evaluará un valor string no numérico como *0* si se compara con uninteger, pero aún devuelve la cadena si ésta es vista como el valor numérico más bajo. Si varios argumentos se evalúan a *0*, min ( ) devolverá el valor de cadena alfanumérica más bajo si se da alguna cadena, o de lo contrario se devuelve un *0* numérico.

Parámetros

valores – Un array que contiene los valores.
value 1 –Cualquier valor comparable.
value 2 – Cualquier valor comparable.

……
Cualquier valor comparable.

Valores devueltos

min ( ) devuelve el valor numéricamente más bajo de los parámetros.

Ejemplos de uso de min ( )

```
<?php
echo min (2, 3, 1, 6, 7);  // 1
echo min (array (2, 4, 5)); // 2
echo min (0, 'hola');     // 0
echo min ('hola', 0);     // hola
echo min ('hola', -1);    // -1
// Con varios arrays, min las compara de izquierda a derecha
// así que en nuestro ejemplo: 2 == 2, pero 4 < 5
$val = min (array (2, 4, 8), array (2, 5, 1)); // array (2, 4, 8)
// Si se pasan un array y algo que no sea un array, el array nunca
// es devuelto ya que se considera el valor mayor
$val = min ('cadena', array (2, 5, 7), 42);   // cadena
?>
```

MT_GETRANDMAX

mt_getrandmax – Mostrar el mayor valor aleatorio posible

Descripción

int mt_getrandmax (void)
Devuelve el valor máximo que puede ser devuelto por una llamada amt_rand ( ).

Valores devueltos

Devuelve el valor aleatorio máximo devuelto por mt_rand ( ).

Ejemplo: Calcular un número de coma flotante aleatorio

```
<?php
function randomFloat($min = 0, $max = 1) {
    return $min + mt_rand ( ) / mt_getrandmax( ) * ($max - $min);
}
var_dump (randomFloat ( ));
var_dump (randomFloat (2, 20));
?>
```

El resultado del ejemplo sería algo similar a:

```
float (0.91601131712832)
float (16.511210331931)
```

MT_RAND

mt_rand – Genera un mejor número entero aleatorio

Descripción

```
int mt_rand (void)
int mt_rand (int $min, int $max)
```

Muchos generadores de números aleatorios de antiguas libcs tienen características dudosas o desconocidas y son lentas. De manera predeterminada, PHP usa la libc generadora de números aleatorios con la función rand ( ). La función mt_rand ( ) es un sustituto de dicha función. Utiliza en generador de

números aleatorios con características conocidas usando » Mersenne Twister, que produce números aleatorios cuatro veces más rápido que el promedio proporcionado por la libc rand ( ).

Si emplea sin los argumentos opcionales min y max, mt_rand ( ) devuelve un valor pseudoaleatorio entre 0 y mt_getrandmax ( ). Para obtener un número aleatorio entre 5 y 15 (incluidos), por ejemplo, use *mt_rand (5, 15).*

Parámetros

Min

Opcionalmente, el menor valor a devolver (por defecto: 0).

max

Opcionalmente, el mayor valor a devolver (por defecto: mt_getrandmax ( ))

Valores devueltos

Un valor entero aleatorio entre min (o 0) y max (o mt_getrandmax ( ), incluidos), o FALSE si max es menor que min.

Ejemplo de mt_rand ( )

```
<?php
echo mt_rand ( ) . "\n";
echo mt_rand ( ) . "\n";
echo mt_rand (5, 15);
?>
```

El resultado del ejemplo sería algo similar a:

```
1604716014
1478613278
6
```

Notas

Precaución

Esta función no genera valores criptográficos fiables por lo que no debería usarse para propósitos criptográficos. Si fuera necesario un valor criptográfico seguro, considérese usaropenssl_random_pseudo_bytes ( ) en su lugar.

Precaución

La distribución de mt_rand ( ) devuelve valores que se inclinan hacia los números pares en versiones 64 bits de PHP, cuando max está más allá de *2^32.*

MT_SRAND

mt_srand – Genera el mejor número aleatorio a partir de una semilla

Descripción

void mt_srand ([int $seed])

Incorpora la semilla seed al generador de números aleatorios, o con un valor aleatorio si no se proporciona seed.

Nota: Desde PHP 4.2.0, no es necesario usar una semilla para usar el generador de números aleatorios con srand ( ) o mt_srand ( ) ya que ahora se hace automáticamente.

Parámetros

seed – Valor de la semilla opcional

Valores devueltos

No devuelve ningún valor.

Ejemplo de mt_srand ( )

```
<?php
// semilla de microsegundos
function make_seed ( )
{
  list ($usec, $sec) = explode (' ', microtime ( ));
return (float) $sec + ((float) $usec * 100000);
}
mt_srand (make_seed ( ));
$randval = mt_rand ( );
?>
```

OCTDEC

octdec – Octal a decimal

Descripción

number octdec (string $octal_string)

Devuelve el valor decimal equivalente al número octal representado por el argumento cadena_octal.

Parámetros

Octal_string

El string octal a convertir

Valores devueltos

La representación decimal de octal_string

Ejemplo de octdec ( )

```
<?php
echo octdec ('77') . "\n";
echo octdec (decoct (45));
?>
```

El resultado del ejemplo sería:
63
45

PI

pi – Obtener valor de pi

Descripción

float pi (void)
Devuelve una aproximación de pi. El valor float devuelto tiene una precisión basada en la directiva precisión en php.ini, cuyo valor predeterminado es 14. Asimismo, es posible usar la constante M_PI, la cual produce el mismo resultado que pi ( ).

Valores devueltos

El valor de pi como flotante.

Ejemplo de pi ( )

```
<?php
echo pi ( ); // 3.1415926535898
echo M_PI; // 3.1415926535898
?>
```

POW

pow – Expresión exponencial

Descripción

number pow (number $base, number $exp)
Devuelve el valor base elevado a la potencia exp.

Parámetros

base – La base a ser usada
exp – El exponente

Valores devueltos

Base elevada a la potencia de exp. Si ambos argumentos son números enteros no negativos y el resultado puede ser representado como un entero, el resultado será devuelto con tipo integer, de lo contrario se devuelve como un float.

Ejemplo de pow ( )

```
<?php
var_dump (pow (2, 8)); // int (256)
echo pow (-1, 20); // 1
echo pow (0, 0); // 1
echo pow (-1, 5.5); // PHP >4.0.6  NAN
echo pow (-1, 5.5); // PHP <=4.0.6 1.#IND
?>
```

Nota

Esta función convertirá toda entrada en un número, incluso valores no escolares, lo cual podría conducir a resultados *extraños*.

RAD2DEG

rad2deg - Convierte el número en radianes a su equivalente en grados

Descripción

float rad2deg (float $number)
Esta función convierte a un número desde radianes a grados.

Parámetros

number – Un valor en radianes

Valores devueltos

El equivalente de number en grados

Ejemplo de rad2deg ( )

```
<?php
echo rad2deg (M_PI_4); // 45
?>
```

RAND

rand – Genera un número entero aleatorio

Descripción

int rand (void)
int rand (int $min, int $max)

Si se invoca sin los argumentos opcionales min y max, rand ( ) devuelve un entero pseudoaleatorio entre 0 y getrandmax ( ). Para obtener un número aleatorio entre 5 y 15 (incluidos), por ejemplo, use *rand (5, 15).*

Nota: En algunas plataformas (como en Windows), getrandmax ( ) sólo alcanza hasta 32767. En caso de necesitar un valor mayor de 32767, se deberá especificar min y max para crear un valor mayor que este, o considere emplear mt_rand ( ) en su lugar.

Parámetros

Min – El menor valor a devolver (por defecto: 0)
Max – El mayor valor a devolver (por defecto: getrandmax ( ))

Valores devueltos

Un valor pseudoaleatorio entre min (o 0) y max (o getrandmax ( ), incluidos).

Ejemplo de rand ( )

```
<?php
echo rand ( ) . "\n";
echo rand ( ) . "\n";
echo rand (5, 15);
?>
```

El resultado del ejemplo sería algo similar a:

```
7771
22264
11
```

Precaución

Esta función no genera valores criptográficos fiables por lo que no debería usarse para propósitos criptográficos. Si fuera necesario un valor criptográfico seguro, considérese usaropenssl_random_pseudo_bytes ( ) en su lugar.

ROUND

round – Redondea un float

Descripción

float round (float $val [, int $precision = 0 [, int$mode =
PHP_ROUND_HALF_UP]])

Devuelve el valor redondeado de val con la precisión especificada (número de dígitos después del punto decimal). Precisión puede también ser negativo o cero (valor predeterminado).

Nota: PHP no maneja correctamente cadenas como *"12,300.2"* por defecto. Consulte conversión desde cadenas.

Parámetros

val – El valor a redondear

precisión - Opcionalmente, el número de dígitos decimales a redondear.
mode - Use una de las siguientes constantes para especificar el modo de redondeo.

Constante	Descripción
PHP_ROUND_HALF_UP	Redondea val hacia arriba a precisión lugares decimales alejándose de cero, cuando está a medio camino. Hace que 1.5 sea 2, y -1.5 sea -2.
PHP_ROUND_HALF_DOWN	Redondea val hacia abajo a precisión lugares decimales hacia cero, cuando está a medio camino. Hace que 1.5 sea 1, y -1.5 sea -1.
PHP_ROUND_HALF_EVEN	Redondea val a precisión lugares decimales hacia el siguiente valor par.
PHP_ROUND_HALF_ODD	Redondea val a precisión lugares decimales hacia el siguiente valor

Constante	Descripción
	impar.

Valores devueltos

El valor redondeado

Ejemplo de round ( )

```
<?php
echo round (3.4); // 3
echo round (3.5);  // 4
echo round (3.6);  // 4
echo round (3.6, 0); // 4
echo round (1.95583, 2);  // 1.96
echo round (1241757, -3); // 1242000
echo round (5.045, 2);  // 5.05
echo round (5.055, 2);  // 5.06
?>
```

SIN

sin – Seno

Descripción

float sin (float $arg)

sin ( ) devuelve el seno del parámetro arg. El parámetro arg se encuentra en radianes.

Parámetros

arg – Un valor en radianes

Valores devueltos

El seno de arg

Ejemplo de sin ( )

```
<?php
// La precisión depende de su directiva de precisión
echo sin (deg2rad (60));  //  0.866025403 ...
echo sin (60);          // -0.304810621 ...
?>
```

SINH

sinh – Seno hiperbólico

Descripción

float sinh (float $arg)
Devuelve el seno hiperbólico de arg, definido como (exp (arg) – exp (-arg))/2.

Parámetros

arg – El argumento a procesar

Valores devueltos

El seno hiperbólico de arg

SQRT

sqrt – Raíz cuadrada

Descripción

float sqrt (float $arg)
Devuelve la raíz cuadrada de arg.

Parámetros

arg – El argumento a procesar

Valores devueltos

La raíz cuadrada de arg o el valor especial *NAN* para números negativos.

Ejemplo de sqrt ( )

```
<?php
// La precisión depende de la directiva de precisión elegida
echo sqrt (9); // 3
echo sqrt (10); // 3.16227766 ...
?>
```

SRAND

srand – Genera un número aleatorio a partir de una semilla

Descripción

void srand ([int $seed])

Incorpora la semilla seed al generador de números aleatorios, o con un valor aleatorio si no se proporciona seed.

Nota. Desde PHP 4.2.0, no es necesario usar una semilla para usar el generador de números aleatorios con srand ( ) o mt_srand ( ) ya que ahora se hace automáticamente.

Parámetros

seed – Valor de la semilla opcional

Valores devueltos

No devuelve ningún valor.

Ejemplo de srand ( )

```
<?php
// semilla de microsegundos
function make_seed ( )
{
  list ($usec, $sec) = explode (' ', microtime ( ));
  return (float) $sec + ((float) $usec * 100000);
}
srand (make_seed ( ));
$randval = rand ( );
?>
```

TAN

tan – Tangente

Descripción

float tan (float $arg)
tan ( ) devuelve la tangente del parámetro arg. El parámetro arg se encuentra en radianes.

Parámetros

arg – El argumento a procesar en radianes

Valores devueltos

La tangente de arg

Ejemplo de tan ( )

```
<?php
echo tan (M_PI_4); // 1
?>
```

TANH

tanh – Tangente hiperbólica

Descripción

float tanh (float $arg)

Devuelve la tangente hiperbólica de arg, definida como *sinh (arg)/cosh (arg)*.

Parámetros

arg- El argumento a procesar

Valores devueltos

La tangente hiperbólica de arg.

FORMULARIOS CON PHP

Los Formularios no forman parte de PHP, sino del lenguaje estándar de Internet, HTML. Vamos a dedicar en este capítulo algunas líneas al HTML, para entrar posteriormente a tratarlos con PHP.

Todo formulario comienza con la etiqueta <FORM ACTION="lo_que_sea.php" METHOD="post/get"> . Con. Con ACTION indicamos el script que va procesar la información que recogemos en el formulario, mientras que METHOD nos indica si el usuario del formulario va ha enviar datos (post) o recogerlos (get). La etiqueta <FORM> indica el final del formulario.

A partir de la etiqueta <FORM> vienen los campos de entrada de datos que pueden ser:

Cuadro de texto

<input type="text" name="nombre" size="20" value="jose">

Cuadro de texto con barras de desplazamiento

```
<textarea rows="5" name="descripcion" cols="20">Es de color rojo</textarea>
```

Casilla de verificación

```
<input type="checkbox" name="cambiar" value="ON">
```

Botón de opción

```
<input type="radio" value="azul" checked name="color">
```

Menú desplegable

```
<select size="1" name="dia">
<option selected value="lunes">lunes</option>
<option>martes</option>
<option value="miercoles">miércoles</option>
</select>
```

Boton de comando

```
<input type="submit" value="enviar" name="enviar">
```

Campo oculto

<input type="hidden" name="edad" value="55">

Este último tipo de campo resulta especialmente útil cuando queremos pasar datos ocultos en un formulario.

Como habrás observado todos lo tipos de campo tienen un modificador llamado name, que no es otro que el nombre de la variable con la cual recogeremos los datos en el script indicado por el modificador ACTION de la etiqueta FORM FORM, con value establecemos un valor por defecto.

A continuación veamos un ejemplo, para lo cual crearemos un formulario en HTML como el que sigue y lo llamaremos formulario.htm:

<HTML>

<BODY>

<FORM METHOD="post" ACTION="mis_datos.php">

<input type="hidden" name="edad" value="55">

Tu nombre <input type="text" name="nombre" size="30" value="jose">

Tu sistema favorito

<select size="1" name="sistema">

<option selected value="Linux">Linux</option>

<option value="Unix">Unix</option>

```
<option value="Macintosh">Macintosh</option>

<option value=&qmargin-left: 75"><option value="Windows">Windows</option>

</select></p>

<p>¿Te gusta el futbol ? <input type="checkbox" name="futbol" value="ON"></p>

<p>¿Cual es tu sexo?</p>

<blockquote>

<p>Hombre<input type="radio" value="hombre" checked name="sexo"></p>

<p>Mujer <input type="radio" name="sexo" value="mujer"></p>

</blockquote>

<p>Aficiones</p>

<p><textarea rows="5" name="aficiones" cols="28"></textarea></p>

<p><input type="submit" value="Enviar datos" name="enviar">

<input type="res-left: 50"> <input type="reset" value="Restablecer" name="B2"></p>

</FORM>

</BODY>
```

<HTML>

Y ahora creemos el script PHP llamado desde le formulario mis_datos.php:

Todos los datos se encuentran en la variable $_POST, ya que el formulario está enviado por el método post.

```
<?PHP;

if (isset ($_POST ['enviar']) {

echo "Hola <b>" . $_POST ['nombre'] . "</b> que tal estás<BR>n";

echo "Eres " . $_POST ['sexo'] . "<BR>n";

echo "Tienes " . $_POST ['edad'] . "<BR>n";

echo "Tu sistema favorito es " . $_POST ['sistema'] . "<BR>n";

if (isset ($_POST ['futbol']) {

echo "Te gusta el futbol <BR>n";

} else odigo" style="margin-left: 50">} else {

echo "NO te gusta el futbol <BR>n";

}

if ($_POST ['aficiones'] != "") {
```

```
echo "Tus aficiones son: <BR>n";

echo nl2br ($_POST ['aficiones']);

} else {

echo "NO tienes aficiones <BR>n";

}

}

echo "<a href='formulario.htm'>VOLVER AL FORMULARIO</a>"

?>
```

Una vez rellenados los datos del formulario, pulsamos el botón Enviar datos, con lo que el campo enviar toma lo que su etiqueta value indica, es decir enviar = "Enviar datos". En nuestro script lo primero que evaluamos es que se haya enviado el formulario, y para ello nada mejor que comprobar que la variable $ enviar no está vacía. Le ponemos el signo dólar delante a enviar, ponemos el signo dólar delante a enviar, ya que en PHP todas las variables se les refiere con este signo.

Hay que tener en cuenta que si fusionáramos el código de ambos ficheros, nos ahorraríamos uno, pero no también se puede hacer en dos como lo estamos haciendo. Si la variable $ enviar está vacia, enviamos el formulario.

```
<?PHP;

if ($ enviar) {
```

```
echo "Hola <b>". $ nombre. "</b> que tal estás<BR>n";

echo "Eres ". $ sexo. "<BR>n";

echo "Tienes ". $ edad. "<BR>n";

echo "Tu sistema favorito es ". $ sistema. "<BR>n";

if ($ futbol) {

echo "Te gusta el futbol <BR>n";

} else {

echo "NO te gusta el futbol <BR>n";

}

if ($aficiones != "") {

< stuot;)>
echo "Tus aficiones son: <BR>n";

echo nl2br($aficiones);

} else {

echo "NO tienes aficiones <BR>n";

}
```

```
echo "<a href='$PHP_SELF'>VOLVER AL FORMULARIO</a>"

} else {

<HTML>

<BODY>

<FORM METHOD="post" ACTION="<?PHP echo $PHP_SELF ?>">

<input type="hidden" name="edad" value="55">

<p>Tu nombre <input type="text" name="nombre" size="30" nombre" size="30" value="jose"></p>

<p>Tu sistema favorito

<select size="1" name="sistema">

<option selected value="Linux">Linux</option>

<option value="Unix">Unix</option>

<option value="Macintosh">Macintosh</option>

<option value="Windows">Windows</option>

</select></p>

<p>¿Te gusta el futbol ? <input type="checkbox" name="futbol" value="ON"></p>
```

```
<p>¿Cual es tu sexo?</p>

<blockquote>

<p>Hombre<input type="radio" value="hombre" checked name="sexo"></p>

<p>="codigo" style="margin-left: 100"><p>Mujer <input type="radio" name="sexo"
value="mujer"></p>

</blockquote>

<p>Aficiones</p>

<p><textarea rows="5" name="aficiones" cols="28"></textarea></p>

<p><input type="submit" value="Enviar datos" name="enviar">

<input type="reset" value="Restablecer" name="B2"></p>

</FORM>

</BODY>

</HTML>

<?PHP

} //fin IF

?>
```

La variable de entorno $PHP_SELF, es una variable de entorno que nos devuelve el nombre del script que estamos ejecutando. Y por último, hacer notar el uso de la función nl2br ( ), nl2br ( ), con la cuál sustituimos los retornos de carro del texto, los cuáles no reconocen los navegadores, por la etiqueta
 .

Ejemplo

1. Leer un número, hacer su cuadrado e imprimirlo

Leer un número, hacer su cuadrado e imprimirlo

```
<html>
   <head>
      <title>numero elevado</title>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
   </head>
   <body>
<?php

if ($_SERVER['REQUEST_METHOD']=='POST')
{
   $ numero = floatval ($_POST['numero']);
   $ cuadrado=$ numero * $ numero;
   echo "Valor de $ numero al cuadrado es: $ cuadrado<br>\n";
}
 ?>
      <form method="post">
         <table style="text-align: left; margin-left: auto; margin-right: auto;" border="1" cellpadding="1" cellspacing="1">
            <tbody>
```

```
            <tr><td>Ingrese el valor de numero</td><td><input name="numero"></td></tr>
            <tr align="center"><td colspan="2" rowspan="1"><input value="calcular" type="submit"></td></tr>
          </tbody>
        </table>
      </form>
    </body>
</html>
```

30 EJERCICIOS PROPUESTOS SIN SOLUCIÓN

1. Dado un conjunto de calificaciones encontrar la calificación mayor y la menor; y si se repiten las calificaciones, indicar cuántas veces.
2. Dados el peso y la talla de una persona, calcular su índice de masa corporal.
3. Dados dos puntos en el espacio, calcular la distancia del segmento que los separa y encontrar su punto medio.
4. Dado un vector en $\mathbb{R}^3$, expresarlo como unitario.
5. Dados dos vectores en $\mathbb{R}^3$, decir si son ortogonales, paralelos u oblicuos.
6. Dado un vector en $\mathbb{R}^3$, calcular los cosenos directores.
7. Dados 4 puntos en $\mathbb{R}^3$ [(-1,3,2), (3,-3,4), (5,0,3) y (1,6,1)] calcular el área del paralelepípedo.
8. Encontrar la pendiente de la recta tangente a la parábola $y=x^2+2x$ en P (-3, 3).
9. Dado un triángulo rectángulo con catetos a y b, calcular la longitud de la hipotenusa.
10. Dado un número con 3 dígitos, calcular la media entre ellos.
11. Dado un número con tres dígitos, si es mayor que 500, calcular el factorial de cada dígito; si es menor o igual que 500, realizar el producto de los dígitos.
12. Dado un número entero real, indicar si es negativo o positivo; si es negativo, hacerlo positivo y calcular su factorial; si es positivo, calcular su raíz.

13. Dado un número del 1 al 10, convertirlo a binario.
14. Dado un número del 1 al 10, convertirlo a octal.
15. Dado un número del 1 al 10, convertirlo a hexadecimal.
16. Dada una cantidad en pesos, convertirla a euros y obtener el 30% de dicha cantidad.
17. Dado un punto en coordenadas polares, expresarlo en tres formas diferentes.
18. Dado un punto en coordenadas polares, calcular sus coordenadas cartesianas.
19. Dado dos vectores en $\mathbb{R}^3$ calcular su producto vectorial.
20. Dados tres puntos en $\mathbb{R}^3$, calcular la ecuación cartesiana del plano que pasa por los tres puntos.
21. Dado un punto S y un vector normal n, encontrar la ecuación del plano.
22. Dado el valor de n, calcular la serie $\sum (1/ (2 + 5^n))$.
23. La suma de tres números en progresión aritmética es 27 y la suma de sus cuadrados es 293. Encontrar los tres números.
24. Dados los elementos diagonales de una matriz cuadrada, encontrar su traza.
25. Calcular las raíces de una ecuación cuadrática dada e imprimir sólo el (los) término (s) positivo (s).
26. Dados dos puntos en $\mathbb{R}^3$, si los elementos de los puntos son negativos, restar los puntos; si son positivos, sumar los puntos; si hay de ambos, realizar producto punto.
27. Contando el día de hoy, ¿cuántos días faltan para el 06 de junio? Si el resultado es de dos dígitos, calcula el producto de sus dígitos; si es de un solo dígito calcula su factorial.
28. Dado un número del 1 al 10, enlista sus 10 siguientes múltiplos y al último número de la lista calcularle el factorial.
29. Dado un vector de $\mathbb{R}^3$, encuentra 5 vectores paralelos a él.
30. Dado el radio de una esfera, calcular su volumen.

BIBLIOGRAFÍA

- Manual de PHP.
  Por Stig Sæther Bakken, Alexander Aulbach, Egon Schmid, Jim Winstead, Lars Torben Wilson, Rasmus Lerdorf, Zeev Suraski, Andrei Zmievski, y Jouni Ahto
  Editado por Rafael Martínez
  Publicado25-02-2001
  Copyright©1997, 1998, 1999, 2000, 2001
  Por el Grupo de documentación de PHP
- Tutorial de PHP y MySQL COMPLETO
  © José Antonio Rodríguez 2000.

BIBLIOGRAFÍA

http://www.monografias.com/trabajos82/que-es-red/que-es-red.shtml

http://es.tldp.org/Manuales-LuCAS/manual_PHP/manual_PHP/

http://www.php.net/manual/es/intro-whatis.php

http://www.php.net/manual/es/intro-whatcando.php

http://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=542:que-es-un-servidor-y-cuales-son-los-principales-tipos-de-servidores-proxydns-webftpsmtp&catid=57:herramientas-informaticas&Itemid=179

http://doc.ubuntu-es.org/HTTPD_Servidor_web_Apache2

http://paraquesirven.com/para-que-sirven-las-bases-de-datos/

http://www.carlospes.com/minidiccionario/compilador.php

http://enciclopedia.us.es/index.php/Int%C3%A9rprete_inform%C3%A1tico

http://es.wikipedia.org/wiki/MySQL

http://es.wikipedia.org/wiki/Python

http://es.wikipedia.org/wiki/ActionScript

http://es.wikipedia.org/wiki/C_(lenguaje_de_programaci%C3%B3n)

http://www.htmlpoint.com/perl/perl_02.htm

http://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n)

http://es.wikipedia.org/wiki/Xcode

http://www.php.net/manual/es/control-structures.continue.php

http://www.php.net/manual/es/control-structures.declare.php

http://www.php.net/manual/es/function.require.php

http://www.php.net/manual/es/function.include.php

http://www.php.net/manual/es/function.require-once.php

http://www.php.net/manual/es/function.include-once.php

http://www.php.net/manual/es/control-structures.goto.php

http://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=574:funciones-basicas-en-php-strreplace-time-date-md5-strtolower-strtoupper-ejemplos-cu00828b&catid=70:tutorial-basico-programador-web-php-desde-cero&Itemid=193

CAPÍTULO 2

Yenifer Rivas García
Alumna del Cuarto Semestre de Computación III
de la Facultad de Ciencias Exactas

PRÓLOGO

El trabajo que aquí se presenta es para aquellas personas que no conocen el funcionamiento y estructura de PHP entre otras cosas. Se explica desde que es, como se estila y que programas lo conforma, además de la función que desempeñan esos programas en el PHP, Nos mencionan como se desarrolla un programa básico en PHP hasta ejemplos muy complejos.

Se trata de que lo expuesto en este programa sea beneficio para aquellos que quieran saber más a fondo, lo que es y todo lo que conlleva, hasta aquellos que solo buscan algo de lo que es PHP.

Se empezará desde conceptos básicos muy claros que debe tener la persona que lee esto para poder entender lo que desempeña PHP, después pasara a algunas pastes que conforma el PHP, su instalación en este caso en Windows 8 y dira como crear tu primer programa en PHP y su estructura; finalizara con algunos ejemplos para que el lector pueda observar un tipo de razonamiento para la realización de ellos. Desde programas muy básicos hasta programas que son un poco más difíciles de entender lo que está haciendo el programa. Esperando que lo aquí juntado de diversas paginas les puede ayudar.

CONCEPTOS BÁSICOS

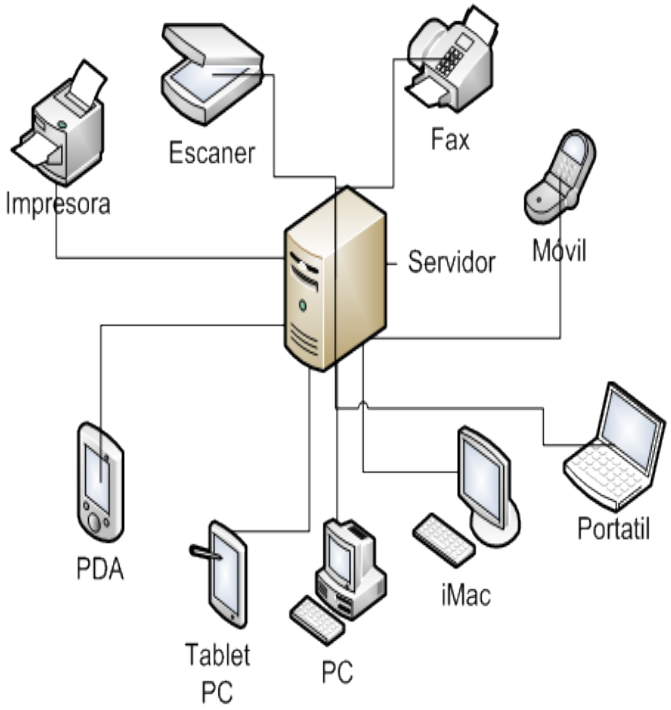
SERVIDOR

Aplicación informática o programa que realiza algunas tareas en beneficio de otras. Algunos servidores habituales son los servidores de archivos, que permiten a los usuarios almacenar y acceder a los archivos de una computadora y los servicios de aplicaciones, que realiza tareas en beneficio directo del usuario final.

Un servidor no necesariamente tiene que ser una máquina de última generación puede ser desde una computadora de bajos recursos, hasta una máquina sumamente potente. Un servidor puede hasta ser un proceso que entregue información o sirva a otro proceso.

Tipos de servidor

Tipos comunes de servidores:



- Servidor de archivos. En el se almacenan varios archivos y los distribuye a otros clientes en la red.
- Servidor de impresoras. Controla una o más impresoras y acepta trabajo de impresión de otros clientes en la red.
- Servidor de correo. Almacena, envía, recibe y realiza otras operaciones realizadas con el correo electrónico para los clientes de la red.
- Servidor proxy. Permite administrar el acceso a internet en una red de computadoras permitiendo o negando el acceso a diferentes sitios web.
- Servidor web. Almacena documentos de HTML, imágenes, archivos de texto y demás material web, y distribuye este contenido a cliente que la pidan en la web.
- Servidor base de datos. Provee servicios de base de datos a otros programas u otras computadoras.

BASE DE DATOS

Conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso.

Tipos de base de datos:

- Estáticos.- datos de solo lectura, utilizados primordialmente para estudiar el comportamiento de un conjunto de datos a través del tiempo, realizar proyecciones.
- Dinámicos.- información almacenada que con el tiempo se modifica, permitiendo operaciones como actualización. Como por ejemplo información de un supermercado, una farmacia, un videoclub o una empresa.
- Bibliográficos.- contiene un resumen o extracto de una publicación, pero nunca el texto completo. Así como el autor, fecha de publicación, editorial, etc.
- Texto completo.- fuentes primarias, como por ejemplo el contenido de ediciones de una colección de revistas.

PÁGINA WEB

Se conoce como página web al documento que forma parte de un sitio web y que suele contar con enlaces (también conocidos como hipervínculos o links**)** para facilitar la navegación entre los contenidos.

Las páginas web están desarrolladas con lenguajes de marcado como el **HTML**, que pueden ser interpretados por **l** navegadores. De esta forma, las páginas pueden presentar información en distintos formatos (texto, imágenes, sonidos, videos, animaciones), estar asociadas a datos de estilo o contar con aplicaciones interactivas.

Entre las múltiples características que tiene una página web y que sirven para identificarla se encuentran las siguientes: cuenta con información textual y también con material de tipo audiovisual, está dotada de un diseño atractivo, está optimizada y ejerce como la tarjeta de presentación de una empresa, una persona o un profesional concreto.

Compilador e intérprete

- **Compilador**, que analiza el programa fuente y lo traduce a otro equivalente escrito en otro lenguaje (por ejemplo, en el lenguaje de la máquina). Su acción equivale a la de un traductor humano, que toma un libro y produce otro equivalente escrito en otra lengua.
- **Intérprete**, que analiza el programa fuente y lo ejecuta directamente, sin generar ningún código equivalente. Su acción equivale a la de un intérprete humano, que traduce las frases que oye sobre la marcha, sin producir ningún escrito permanente.

Intérpretes y compiladores tienen diversas ventajas e inconvenientes que los hacen complementarios:

- Un intérprete facilita la búsqueda de errores, pues la ejecución de un programa puede interrumpirse en cualquier momento para estudiar el entorno

(valores de las variables, etc.). Además, el programa puede modificarse sobre la marcha, sin necesidad de volver a comenzar la ejecución.

- Un compilador suele generar programas más rápidos y eficientes, ya que el análisis del lenguaje fuente se hace una sola vez, durante la generación del programa equivalente. En cambio, un intérprete se ve obligado generalmente a analizar cada instrucción tantas veces como se ejecute (incluso miles o millones de veces).
- Un intérprete permite utilizar funciones y operadores más potentes, como por ejemplo ejecutar código contenido en una variable en forma de cadenas de caracteres. Usualmente, este tipo de instrucciones es imposible de tratar por medio de compiladores. Los lenguajes que incluyen este tipo de operadores y que, por tanto, exigen un intérprete, se llaman interpretativos. Los lenguajes compilativos, que permiten el uso de un compilador, prescinden de este tipo de operadores.

RED

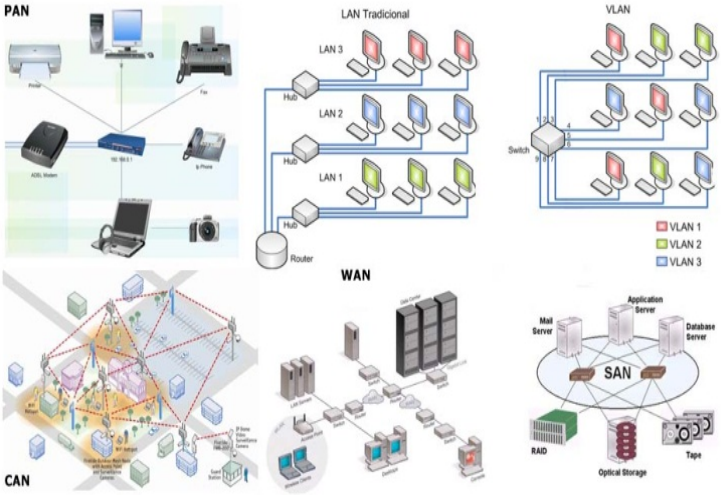
Conjunto de dispositivos interconectados entre sí a través de un medio, que intercambian información y comparten recursos. Básicamente, la comunicación dentro de una red informática es un proceso en el que existen dos roles bien definidos para los dispositivos conectados, emisor y receptor, que se van asumiendo y alternando en distintos instantes de tiempo.

Tipos de redes

Una red informática tiene distintos tipos de clasificación dependiendo de su estructura o forma de transmisión, entre los principales tipos de redes están los siguientes:

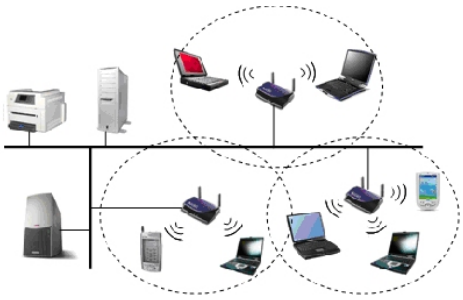
REDES POR ALCANCE

Este tipo de red se nombra con siglas según su área de cobertura: una red de área personal o PAN (Personal Área Network) es usada para la comunicación entre dispositivos cerca de una persona; una LAN (Local Área Network), corresponde a una red de área local que cubre una zona pequeña con varios usuarios, como un edificio u oficina. Para un campus o base militar, se utiliza el término CAN (Campus Área Network). Cuando una red de alta velocidad cubre un área geográfica extensa, hablamos de MAN (Metropolitan Área Network) o WAN (Wide Área Network). En el caso de una red de área local o LAN, donde la distribución de los datos se realiza de forma virtual y no por la simple direccionalidad del cableado, hablamos de una VLAN (Virtual LAN). También cabe mencionar las SAN (Storage Área Network), concebida para conectar servidores y matrices de discos y las Redes Irregulares, donde los cables se conectan a través de un módem para formar una red.



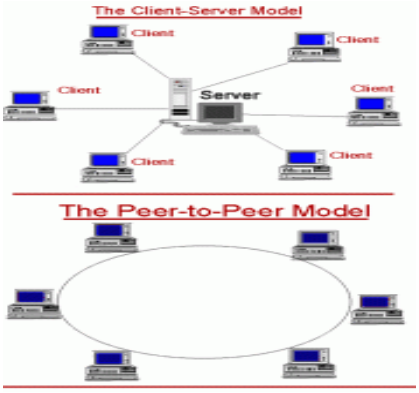
REDES POR TIPO DE CONEXIÓN

Cuando hablamos de redes por tipo de conexión, el tipo de red varía dependiendo si la transmisión de datos es realizada por medios guiados como cable coaxial, par trenzado o fibra óptica, o



medios no guiados, como las ondas de radio, infrarrojos, microondas u otras transmisiones por aire.

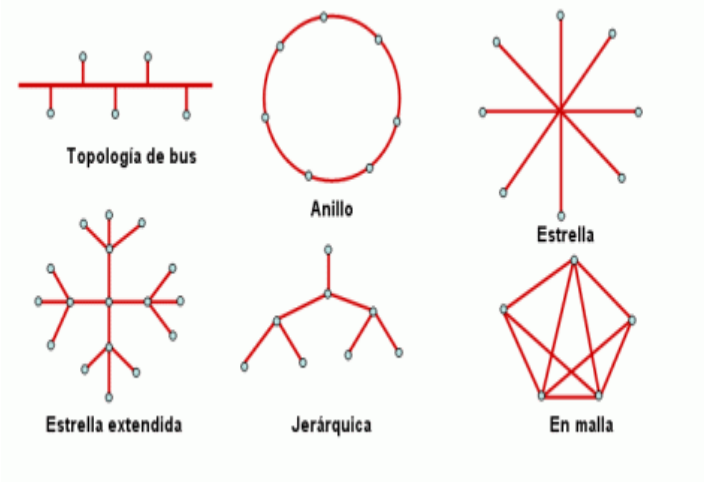
REDES POR RELACION FUNCIONAL



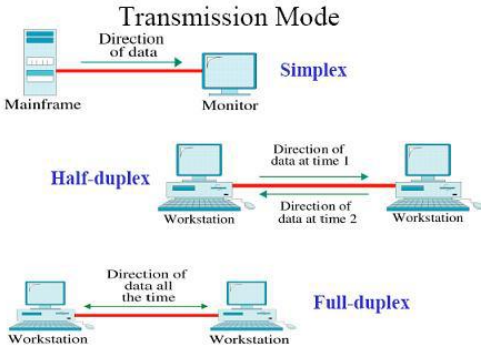
Cuando un cliente o usuario solicita la información a un servidor que le da respuesta es una Relación Cliente/Servidor, en cambio cuando en dicha conexión una serie de nodos operan como iguales entre sí, sin cliente ni servidores, hablamos de Conexiones Peer to Peer o P2P.

REDES POR TOPOLOGIA

La Topología de una red, establece su clasificación en base a la estructura de unión de los distintos nodos o terminales conectados. En esta clasificación encontramos las redes en bus, anillo, estrella, en malla, en árbol y redes mixtas.



REDES POR DIRECCIONALIDAD DE DATOS

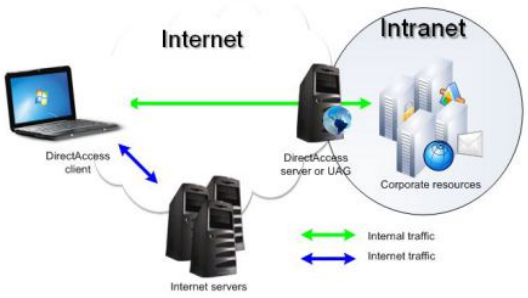


En la direccionalidad de los datos, cuando un equipo actúa como emisor en forma unidireccional se llama Simplex, si la información es bidireccional pero solo un equipo transmite a la vez, es una red Half-Duplex o Semi-Duplex, y si ambos equipos envían y reciben información simultáneamente hablamos de una red Full Duplex.

REDES SEGÚN GRADO DE AUTENTIFICACIÓN

Las Redes Privadas y la Red de Acceso Público, son 2 tipos de redes clasificadas según el grado de autentificación necesario para conectarse a ella. De este modo una red privada requiere el ingreso de claves u otro medio de validación de usuarios, una red de acceso público en cambio, permite que dichos usuarios accedan a ella libremente.

REDES SEGÚN GRADO DE DIFUSIÓN



Otra clasificación similar a la red por grado de autentificación, corresponde a la red por Grado de Difusión, pudiendo ser Intranet o Internet. Una intranet, es un conjunto de equipos que comparte información entre usuarios validados previamente, Internet en cambio, es una red de alcance mundial gracias a que la interconexión de equipos funcionan como una red lógica única, con lenguajes y protocolos de dominio abierto y heterogéneo.

REDES SEGÚN SERVICIO O FUNCIÓN

Por último, según Servicio o Función de las Redes, se pueden clasificar como Redes Comerciales, Educativas o Redes para el Proceso de Datos.

Todas estas clasificaciones, nos permiten identificar la forma en que estamos conectados a una red, qué uso podemos darle y el tipo de información a la cual tendremos acceso. Conocerlas entonces nos servirá para elegir con una base mucho más sólida, qué conexión necesitamos para cubrir las necesidades de nuestro negocio y valorizar los costos que implica cada una de ellas.

PROGRAMAS NECESARIOS PARA EL FUNCIONAMIENTO DE PHP Y OTROS TIPOS DE LENGUAJES DE PROGRAMACIÓN

APACHE

Servidor web de distribución libre y de código abierto, siendo el más popular del mundo desde abril de 1996, con una penetración actual del 50% del total de servidores web del mundo (agosto de 2007).

Algunas características de apache:

- Corre en una multitud de Sistemas Operativos, lo que lo hace prácticamente universal.

- Apache es una tecnología gratuita de código fuente abierto. El hecho de ser gratuita es importante pero no tanto como que se trate de código fuente abierto. Esto le da una transparencia a este software de manera que si

queremos ver que es lo que estamos instalando como servidor, lo podemos saber, sin ningún secreto, sin ninguna puerta trasera.

- Apache es un servidor altamente configurable de diseño modular. Es muy sencillo ampliar las capacidades del servidor Web Apache. Actualmente existen muchos módulos para Apache que son adaptables a este, y están ahí para que los instalemos cuando los necesitemos. Otra cosa importante es que cualquiera que posea una experiencia decente en la programación de C o Perl puede escribir un modulo para realizar una función determinada.

- Apache trabaja con gran cantidad de Perl, PHP y otros lenguajes de script. Perl destaca en el mundo del script y Apache utiliza su parte del pastel de Perl tanto con soporte CGI como con soporte mod perl. También trabaja con Java y páginas jsp. Teniendo todo el soporte que se necesita para tener páginas dinámicas.

- Apache te permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. Es posible configurar Apache para que ejecute un determinado script cuando ocurra un error en concreto.

- Tiene una alta configurabilidad en la creación y gestión de logs. Apache permite la creación de ficheros de log a medida del administrador, de este modo puedes tener un mayor control sobre lo que sucede en tu servidor .

MySQL

MySQL es una base de datos muy rápida en la lectura cuando utiliza el motor no transaccional MyISAM, pero puede provocar problemas de integridad en entornos de alta concurrencia en la modificación. En aplicaciones web hay baja concurrencia en la modificación de datos y en cambio el entorno es intensivo en lectura de datos, lo que hace a MySQL ideal para este tipo de aplicaciones. Sea

cual sea el entorno en el que va a utilizar MySQL, es importante monitorizar de antemano el rendimiento para detectar y corregir errores tanto de SQL como de programación.

Características disponibles en las últimas versiones se puede destacar:

- Amplio subconjunto del lenguaje SQL. Algunas extensiones son incluidas igualmente.
- Disponibilidad en gran cantidad de plataformas y sistemas.
- Posibilidad de selección de mecanismos de almacenamiento que ofrecen diferentes velocidades de operación, soporte físico, capacidad, distribución geográfica, transacciones...
- Transacciones y claves foráneas.
- Conectividad segura.
- Replicación.
- Búsqueda e indexación de campos de texto.

MySQL es un sistema de administración de bases de datos. Una base de datos es una colección estructurada de tablas que contienen datos. Esta puede ser desde una simple lista de compras a una galería de pinturas o el vasto volumen de información en una red corporativa. Para agregar, acceder a y procesar datos guardados en un computador, usted necesita un administrador como MySQL Server. Dado que los computadores son muy buenos manejando grandes cantidades de información, los administradores de bases de datos juegan un papel central en computación, como aplicaciones independientes o como parte de otras aplicaciones.

ACTIONSCRIPT

El ActionScript es el lenguaje de programación que ha utilizado Macromedia Flash desde sus comienzos, y que por supuesto, emplea Flash MX. A grandes rasgos,

podemos decir que el ActionScript nos permitirá realizar con Flash MX todo lo que nos propongamos, ya que nos da el control absoluto de todo lo que rodea a una película Flash. Absolutamente de todo.

Sin embargo, en estos dos temas sólo vamos a ver una pequeña introducción a ActionScript que servirá para sentar las bases que permitirán empezar a trabajar con ActionScript. Enseñar a programar con ActionScript requeriría otro curso completo. Profundizar en el conocimiento de este lenguaje queda por cuenta del lector. Recomendamos seguir la estupenda *Ayuda* incluida en FlashMX.

Características generales del ActionScript

- Como ya hemos comentado, el ActionScript es el lenguaje de programación propio de Flash, tal y como el Lingo lo es de Macromedia Director, por ejemplo. El ActionScript está basado en la especificación ECMA-262.

- El ActionScript es, como su nombre indica, un lenguaje de script, esto quiere decir que no hará falta crear un programa completo para conseguir resultados, normalmente la aplicación de fragmentos de código ActionScript a los objetos existentes en nuestras películas nos permiten alcanzar nuestros objetivos.

- El ActionScript es un lenguaje de programación orientado a objetos, tiene similitudes, por tanto, con lenguajes tales como los usados en el Microsoft Visual Basic, en el Borland Delphi etc... aunque, evidentemente no tiene la potencia de un lenguaje puramente orientado a objetos derivado del C o del Pascal como los anteriores.

- El ActionScript presenta muchísimos parecidos con el Javascript; si conoce Javascript, la sintaxis y el estilo de ActionScript le resultarán muy familiares. Las diferencias entre ambos lenguajes las puede encontrar en la ayuda que acompaña al Flash MX.

- En la mayor parte de las ocasiones, no será necesario "programar" realmente, Flash MX pone a nuestra disposición una impresionante colección de "funciones" (de momento entenderemos "**funciones**" como "*código ActionScript que realiza una función determinada"*) ya implementadas que realizan lo que buscamos, bastará con colocarlas en el lugar adecuado.

JAVA

- Java es una tecnología que se usa para el desarrollo de aplicaciones que convierten a la Web en un elemento más interesante y útil. Java no es lo mismo que javascript, que se trata de una tecnología sencilla que se usa para crear páginas web y solamente se ejecuta en el explorador.

- Java le permite jugar, cargar fotografías, chatear en línea, realizar visitas virtuales y utilizar servicios como, por ejemplo, cursos en línea, servicios bancarios en línea y mapas interactivos. Si no dispone de Java, muchas aplicaciones y sitios web no funcionarán.

- Por defecto, Java le notificará inmediatamente que hay nuevas actualizaciones listas para instalarse. Si desea estar al día y mantener la seguridad de su computadora, es importante que acepte e instale las actualizaciones. Si recibe una notificación de actualización de Java en su computadora Windows y no recuerda haberla descargado o instalado, lo más probable es que Java estuviera ya instalado en la nueva computadora.

PHYTON

Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible.

Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma.

Es administrado por la Python Software Foundation. Posee una licencia de código abierto, denominada Python Software Foundation License,[1] que es compatible con la Licencia pública general de GNU a partir de la versión 2.1.1, e incompatible en ciertas versiones anteriores.

Características y paradigmas

Python es un lenguaje de programación multiparadigma. Esto significa que más que forzar a los programadores a adoptar un estilo particular de programación, permite varios estilos: programación orientada a objetos, programación imperativa y programación funcional. Otros paradigmas están soportados mediante el uso de extensiones.

Python usa tipado dinámico y conteo de referencias para la administración de memoria.

Una característica importante de Python es la resolución dinámica de nombres; es decir, lo que enlaza un método y un nombre de variable durante la ejecución del programa (también llamado enlace dinámico de métodos).

Otro objetivo del diseño del lenguaje es la facilidad de extensión. Se pueden escribir nuevos módulos fácilmente en C o C++. Python puede incluirse en aplicaciones que necesitan una interfaz programable.

Aunque la programación en Python podría considerarse en algunas situaciones hostil a la programación funcional tradicional del Lisp, existen bastantes analogías entre Python y los lenguajes minimalistas de la familia Lisp como puede ser Scheme.

Modo interactivo

El intérprete de Python estándar incluye un *modo interactivo* en el cual se escriben las instrucciones en una especie de intérprete de comandos: las expresiones pueden ser introducidas una a una, pudiendo verse el resultado de su evaluación inmediatamente, lo que da la posibilidad de probar porciones de código en el modo interactivo antes de integrarlo como parte de un programa. Esto resulta útil tanto para las personas que se están familiarizando con el lenguaje como para los programadores más avanzados.

Existen otros programas, tales como IDLE, bpython o IPython, que añaden funcionalidades extra al modo interactivo, como el autocompletado de código y el coloreado de la sintaxis del lenguaje.

HTML

El HTML, Hyper Text Markup Language (Lenguaje de marcación de Hipertexto) es el lenguaje de marcas de texto utilizado normalmente en la www (World Wide Web). Fue creado en 1986 por el físico nuclear Tim Berners-Lee; el cual tomo dos herramientas preexistentes: El concepto de Hipertexto (Conocido también como link o ancla) el cual permite conectar dos elementos entre si y el SGML (Lenguaje Estándar de Marcación General) el cual sirve para colocar etiquetas o marcas en un texto que indique como debe verse. HTML no es propiamente un lenguaje de programación como C++, Visual Basic, etc., sino un sistema de etiquetas. HTML no presenta ningún compilador, por lo tanto algún error de sintaxis que se

presente éste no lo detectará y se visualizara en la forma como éste lo entienda. El entorno para trabajar HTML es simplemente un procesador de texto, como el que ofrecen los sistemas operativos Windows (Bloc de notas), UNIX (el editor vi o ed) o el que ofrece MS Office (Word). El conjunto de etiquetas que se creen, se deben guardar con la extensión .htm o .html

Estos documentos pueden ser mostrados por los visores o "browsers" de páginas Web en Internet, como Netscape Navigator, Mosaic, Opera y Microsoft Internet Explorer.

También existe el HTML Dinámico (DHTML), que es una mejora de Microsoft de la versión 4.0 de HTML que le permite crear efectos especiales como, por ejemplo, texto que vuela desde la página palabra por palabra o efectos de transición al estilo de anuncio publicitario giratorio entre página y página.

A continuación vamos a hablar un poco de historia:

Cuando se inserta un trozo de código Php dentro de una página con código Html es necesario indicar al servidor que esas líneas tan raras son código Php. De lo contrario pensaría que sigue siendo Html o que estamos un poco borrachos, je je.
Al igual que cuando se escribe un párrafo en Html se encerraba este entre las etiquetas y , cuando nos interese insertar código Php entre el código Html de una de nuestras páginas usaremos las etiquetas <?php al principio y ?> al final.

```
<html>
  <head>
    <title>Mi quinta página con php</title>
  </head>
  <body>
    <h1>Mi quinto ejemplo Php</h1>
    <p>Este es mi quinto ejemplo con partes de Php</p>
```

```
  </body>
</html>
```

El código Html de arriba representa una página puramente hecha con Html, más bien simplificada pero que nos vale para ver cómo funciona el Php. Sí, he puesto "mi quinto". Realmente es "mi primer..." pero de este modo si alguien nos ve ahora mismo pensará que debemos ser unos genios ya en Php, je je je je. Será otro secreto entre nosotros, vale? He visto tantos ejemplos ya con el típico mensaje de "Hola Mundo!" que me da un poco de angustia usarlo yo también así que.... seamos originales.

Xcode

Xcode es un IDE (Entorno de desarrollo integrado) de Apple que se ofrece de manera gratuita en sistemas Mac OS X. Puede servir para desarrollar aplicaciones en varios lenguajes, que funcionen sobre varias plataformas de la Empresa de la Manzana. En concreto y entre otras, permite la creación de aplicaciones para dispositivos móviles iOS, como iPad, iPhone o iPod, algo a lo que te pretendemos introducir en DesarrolloWeb.com.

Este es el primero de una serie de webcast en los que te presentaremos Xcode y donde podrás introducirte en el desarrollo de aplicaciones sobre iOS. En este programa empezaremos desde cero, para que veas qué es Xcode y las posibilidades que ofrece para los desarrolladores Mac. Veremos el simulador de iPhone o iPad que contiene y echaremos un vistazo a un primer proyecto iOS.

C

C es un lenguaje de programación de propósito general que ofrece economía sintáctica, control de flujo y estructuras sencillas y un buen conjunto de operadores. No es un lenguaje de muy alto nivel y más bien un lenguaje pequeño,

sencillo y no está especializado en ningún tipo de aplicación. Esto lo hace un lenguaje potente, con un campo de aplicación ilimitado y sobre todo, se aprende rápidamente. En poco tiempo, un programador puede utilizar la totalidad del lenguaje.

Este lenguaje ha sido estrechamente ligado al sistema operativo UNIX, puesto que fueron desarrollados conjuntamente. Sin embargo, este lenguaje no está ligado a ningún sistema operativo ni a ninguna máquina concreta. Se le suele llamar lenguaje de programación de sistemas debido a su utilidad para escribir compiladores y sistemas operativos, aunque de igual forma se puede desarrollar cualquier tipo de aplicación.

C++

C++ es un lenguaje de programación diseñado a mediados de los años 1980 por Bjarne Stroustrup. La intención de su creación fue el extender al exitoso lenguaje de programación C con mecanismos que permitan la manipulación de objetos. En ese sentido, desde el punto de vista de los lenguajes orientados a objetos, el C++ es un lenguaje híbrido.

Posteriormente se añadieron facilidades de programación genérica, que se sumó a los otros dos paradigmas que ya estaban admitidos (programación estructurada y la programación orientada a objetos). Por esto se suele decir que el C++ es un lenguaje de programación multiparadigma.

Actualmente existe un estándar, denominado ISO C++, al que se han adherido la mayoría de los fabricantes de compiladores más modernos. Existen también algunos intérpretes, tales como ROOT.

Una particularidad del C++ es la posibilidad de redefinir los operadores, y de poder crear nuevos tipos que se comporten como tipos fundamentales.

El nombre C++ fue propuesto por Rick Mascitti en el año 1983, cuando el lenguaje fue utilizado por primera vez fuera de un laboratorio científico. Antes se había usado el nombre "C con clases". En C++, la expresión "C++" significa "incremento de C" y se refiere a que C++ es una extensión de C.

MAM PRO

MAMP no es más que una recopilación de software (en realidad son **servicios**), que instalará de forma rápida. Si nos dedicamos (o queremos hacerlo) o simplemente por hobbie, al **desarrollo web**, tener un servidor web montado en casa o en la oficina va a acelerar todo el proceso de manera enorme. Así que, manos a la obra.

MAMP está formado por:

- **Apache**. Es el **servidor web** más usado del mundo, por su velocidad, por que es seguro, y por que es open source (o sea, no hay que pagar ninguna licencia).
- **MySQL**. Es un servidor de **base de datos**, también open source. La gran mayoría de las páginas web que visitáis a diario, almacenan sus datos en mysql. Digamos que es la base de datos más extendida en los **servidores web**.
- **PHP**. Es un lenguaje de script interpretado con el cual programaremos páginas web. Tienes más información en su página oficial, y si te das una vuelta por mi blog, verás que hay bastantes posts al respecto.

Las dos primeras aplicaciones, corren como **servicios del sistema** es decir, una vez arrancados, no veremos nada, simplemente están a la escucha, esperando que les llegue una petición. El tercero, será el encargado de realizar tareas de lado del servidor, generar una respuesta, y mandársela al cliente que ha realizado la

petición. Imaginaros que entráis en una página donde tenéis que introducir vuestro usuario y contraseña. **Apache** sería el encargado de mandaros la página que solicita el usuario y contraseña. Una vez los rellenemos, son devueltos de nuevo al **servidor web**, que mediante lo que hayamos programado en **PHP**, consultaremos la **base de datos** (**mysql**) y permitiremos o denegaremos la entrada.

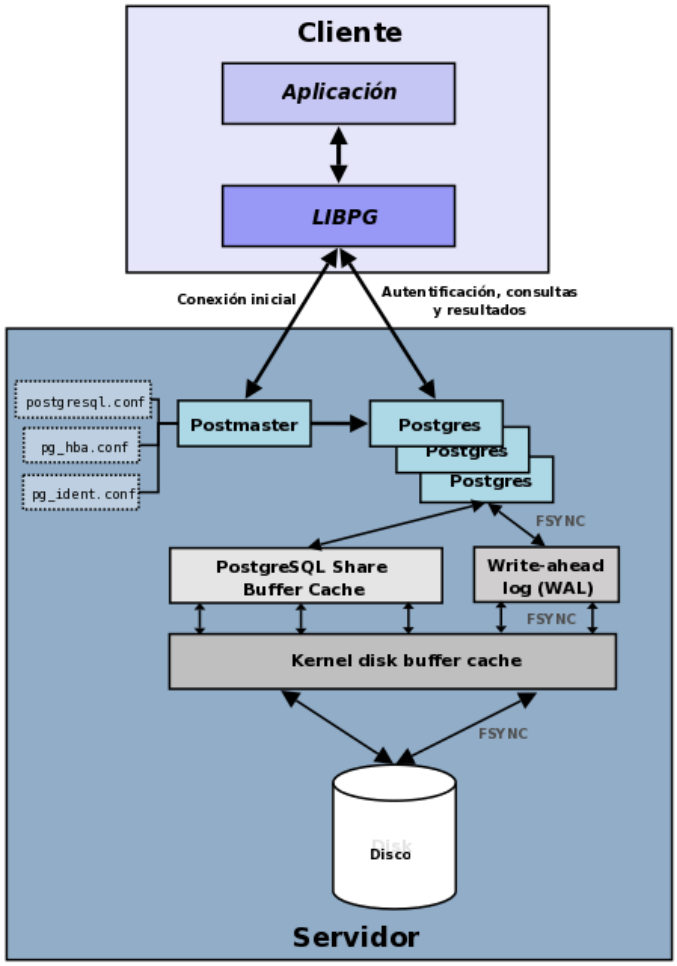
Por ejemplo, el sistema de blogs mas usado del mundo (wordpress) usa estos tres sistemas. Al igual que el CMS más extendido (joomla), o que uno de los paneles de administración de servidores (Plesk).

POSTGRE

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado y en sus últimas versiones no tiene nada que envidiarle a otras bases de datos comerciales.

PostgreSQL utiliza un modelo cliente/servidor y usa *multiprocesos* en vez de *multihilos* para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando.

A continuación teneis un gráfico que ilustra de manera general los componentes más importantes en un sistema PostgreSQL.



- **Aplicación cliente**. Esta es la aplicación cliente que utiliza PostgreSQL como administrador de bases de datos. La conexión puede ocurrir via TCP/IP ó sockets locales.
- **Demonio postmaster**. Este es el proceso principal de PostgreSQL. Es el encargado de escuchar por un puerto/socket por conexiones entrantes de clientes. También es el encargado de crear los procesos hijos que se encargaran de autentificar estas peticiones, gestionar las consultas y mandar los resultados a las aplicaciones clientes.
- **Ficheros de configuración**. Los tres ficheros principales de configuración utilizados por PostgreSQL, postgresql.conf, pg_hba.conf y pg_ident.conf.

- **Procesos hijos postgres**. Procesos hijos que se encargan de autentificar a los clientes, de gestionar las consultas y mandar los resultados a las aplicaciones clientes.
- **PostgreSQL share buffer cache**. Memoria compartida usada por POstgreSQL para almacenar datos en caché.
- **Write-Ahead Log (WAL)**. Componente del sistema encargado de asegurar la integridad de los datos (recuperación de tipo REDO).
- **Kernel disk buffer cache.** Caché de disco del sistema operativo.
- **Disco.** Disco físico donde se almacenan los datos y toda la información necesaria para que PostgreSQL funcione.

A continuación teneis algunas de las características más importantes y soportadas por PostgreSQL:

Generales

- Es una base de datos 100% ACID
- Integridad referencial
- Tablespaces
- Nested transactions (savepoints)
- Replicación asincrónica/sincrónica / Streaming replication - Hot Standby
- Two-phase commit
- PITR - point in time recovery
- Copias de seguridad en caliente (Online/hot backups)
- Unicode
- Juegos de caracteres internacionales
- Regionalización por columna
- Multi-Version Concurrency Control (MVCC)
- Múltiples métodos de autentificación
- Acceso encriptado via SSL
- Actualización in-situ integrada (pg_upgrade)

- SE-postgres
- Completa documentación
- Licencia BSD
- Disponible para Linux y UNIX en todas sus variantes (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64) y Windows 32/64bit.

Diferencia del MySQL CON EL PostgreSQL

- **MySQL**

  - Su principal objetivo de diseño fue la VELOCIDAD. Se sacrificaron algunas características esenciales en sistemas más "serios" con este fin.
  - Otra característica importante es que consume MUY POCOS RECURSOS, tanto de CPU como de memoria.
  - Licencia GPL a partir de la versión 3.23.19.
  - **Ventajas:**
    - Mayor rendimiento. Mayor velocidad tanto al conectar con el servidor como al servir selects y demás.
    - Mejores utilidades de administración (backup, recuperación de errores, etc).
    - Aunque se cuelgue, no suele perder información ni corromper los datos.
    - Mejor integración con PHP.
    - No hay límites en el tamaño de los registros.
    - Mejor control de acceso, en el sentido de qué usuarios tienen acceso a qué tablas y con qué permisos.
    - MySQL se comporta mejor que Postgres a la hora de modificar o añadir campos a una tabla "en caliente".
  - **Inconvenientes:**
    - No soporta transacciones, "roll-backs" ni subselects.
    - No considera las claves ajenas. Ignora la integridad referencial, dejándola en manos del programador de la aplicación.

- **PostgreSQL**

  - Postgres intenta ser un sistema de bases de datos de mayor nivel que MySQL, a la altura de Oracle, Sybase o Interbase.
  - Licencia BSD.
  - **Ventajas:**
    - Por su arquitectura de diseño, escala muy bien al aumentar el número de CPUs y la cantidad de RAM.
    - Soporta transacciones y desde la versión 7.0, claves ajenas (con comprobaciones de integridad referencial).
    - Tiene mejor soporte para triggers y procedimientos en el servidor.
    - Soporta un subconjunto de SQL92 MAYOR que el que soporta MySQL. Además, tiene ciertas características orientadas a objetos.
  - **Inconvenientes:**
    - Consume BASTANTES más recursos y carga más el sistema.
    - Límite del tamaño de cada fila de las tablas a 8k!!! (se puede ampliar a 32k recompilando, pero con un coste añadido en el rendimiento).
    - Es de 2 a 3 veces más lenta que MySQL.
    - Menos funciones en PHP.

En cuanto a consideraciones de estabilidad del servidor, cada comparativa da datos contradictorios. En general parece que MySQL es más estable (aunque también hay gente que opina lo contrario), y que Postgres tiende a desperdiciar memoria y sobrecargar bastante el sistema (aunque de nuevo, hay opiniones distintas).

COMPARATIVA DEL PHYTON vs PHP vs JAVA

Pues ahora les traigo una comparativa entre estos lenguajes aplicados a la Web (bueno en el caso de PHP ya es de naturaleza Web pero no en el caso de Python o de Java).

Php es un lenguaje muy maduro que ha estado en el mercado hace ya muchos años (1995) y como principales ventajas tiene:

1. Una gran comunidad de programadores que te pueden ayudar cuando te atores en algo.
2. Su curva de aprendizaje es muy suave ya que es muy sencillo de aprender (ademas de su gran parecido con C).
3. La cantidad de bibliotecas que te simplifican el desarrollo de cosas como creación de imágenes y PDF manejo de sockets y bases de datos de una forma muy simple y sencilla, por lo tanto,
4. Desde el punto de vista de un negocio el tiempo de desarrollo en PHP es mínima y muy rápida lo que se traduce en alta productividad.
5. La implementación es muy simple por lo tanto la oferta de hosting que soportan esta tecnología es gigantesca.

Como desventajas tiene:

1. Es muy lento lo que en sitios Web que tendrán muchas peticiones por segundo o cargas muy pesadas de acceso a BD, no sera la mejor opción,
2. Si bien la persistencia de datos existe al serializar manualmente o por medio de sesiones, no existe la persistencia de objetos lo cual puede llegar a ser una gran desventaja al programar OOP.
3. Es un lenguaje que al principio no soportaba OOP por lo que hacer código spaguetti es muy facil y casi involuntario, además de que en algunos benchmark demuestra ser lento al ejecutar OOP que un simple código estructurado.

Java es un lenguaje también muy maduro y con mucha experiencia y tiene como ventajas:

1. Al igual que en PHP una gran cantidad de programadores te pueden ayudar.
2. Gran potencia y velocidad, ya que se usan servlets y existe persistencia de objetos.
3. Es un lenguaje totalmente OOP por lo que es imposible programar feo (o al menos no como en PHP XD).

Como desventajas:

1. Su curva de aprendizaje es muy pesada, ya que este lenguaje es muy complejo, lo cual no lo hace justificable si tu desarrollo es medianamente simple.
2. El tiempo de desarrollo y por lo tanto su productividad no es tan bueno que como en PHP al menos en desarrollos relativamente simples como sitios de e-comerce, uso de base de datos sencillo. Aunque en sitios mas complejos donde la carga de consultas a bases de datos sea muy alta es mejor Java.
3. La implementación de esta tecnología es mas costosa por lo tanto, es más escasa la oferta de hosting para la misma.

Python es un lenguaje mas parecido a Java que a PHP (en su forma de operar) y aunque mas antiguo que PHP (1991), su incursión en la web ha sido muy escasa, si bien es un lenguaje interpretado al igual que PHP, se parece mas a Java y si lo piensan bien Java también es interpretado ya que la compilación auténtica no se da en estos lenguajes no como en C o C++, ambos tanto Java como Python traducen el script a un bytecode (muy parecido al lenguaje máquina) a diferencia de PHP que sus opcodes son mas parecidos al lenguaje

ensamblador, y los bytecode son más óptimos, por lo tanto Python es más eficiente en su ejecución que PHP aunque no se le acerca a Java, PERO hay una configuración para Python, que me gusta y por la que apuesto todo, se trata de un framework que soporta sesiones, mvc y mas importante SERVLETS así es Python puede manejar servlets como Java con este framework, por lo que su desempeño mejoraría muchisimo que me atrevería a decir que le anda pisando los talones a Java, pero de eso hablaré mas adelante en un benchmark que pretendo hacer en estas 3 tecnologías.

Como principales ventajas tiene:

1. Curva de aprendizaje muy suave, es un lenguaje que se puede jactar de ser de muy alto nivel casi aproximándose al lenguaje humano, dejando atrás a Java y PHP en este sentido.
2. Se ejecuta como lo hace Java con bytecode lo que lo hace muy rápido resultando estar en medio entre PHP y Java en rapidez.
3. También implementa una gran cantidad de bibliotecas para hacer maravillas como sucede en PHP.

Como desventajas:

1. No hay muchos programadores en esta tecnología y la documentación es muy escasa en español y en inglés también aunque hay mas que en español.
2. Implementar esta tecnología en web es muy compleja por lo tanto los hosting que soportan Python son todavía mas escasos que en Java.

Este es un análisis simple de estas tecnologías, iré haciendo más análisis de estas tecnologías más adelante y más completas, esto es una introducción.

Solo como reflexión final y casi a modo de estocada final y letal, los que estamos a favor de Python como es mi caso, podemos presumir de tener como respaldo a

empresas tan grandes como ni mas ni menos Google, Yahoo y Nasa, estas tres empresas tienen sus sitios Web en Python.

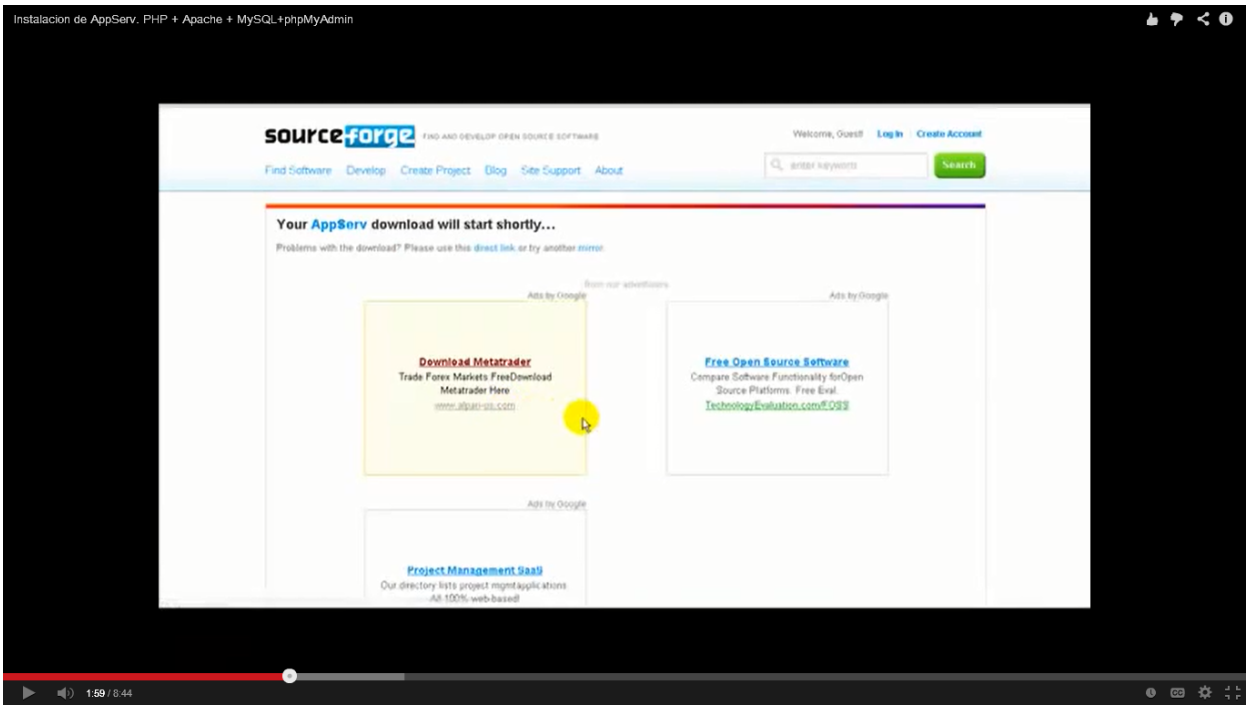
Cabe reflexionar que estas empresas tienen equipos de ingenieros muy capacitados (de lo mejor del mundo) y que si voltearon sus ojos a esta tecnología es por algo ¿no creen?

INSTALACIÓN DE PHP EN WINDOWS 8 O 7

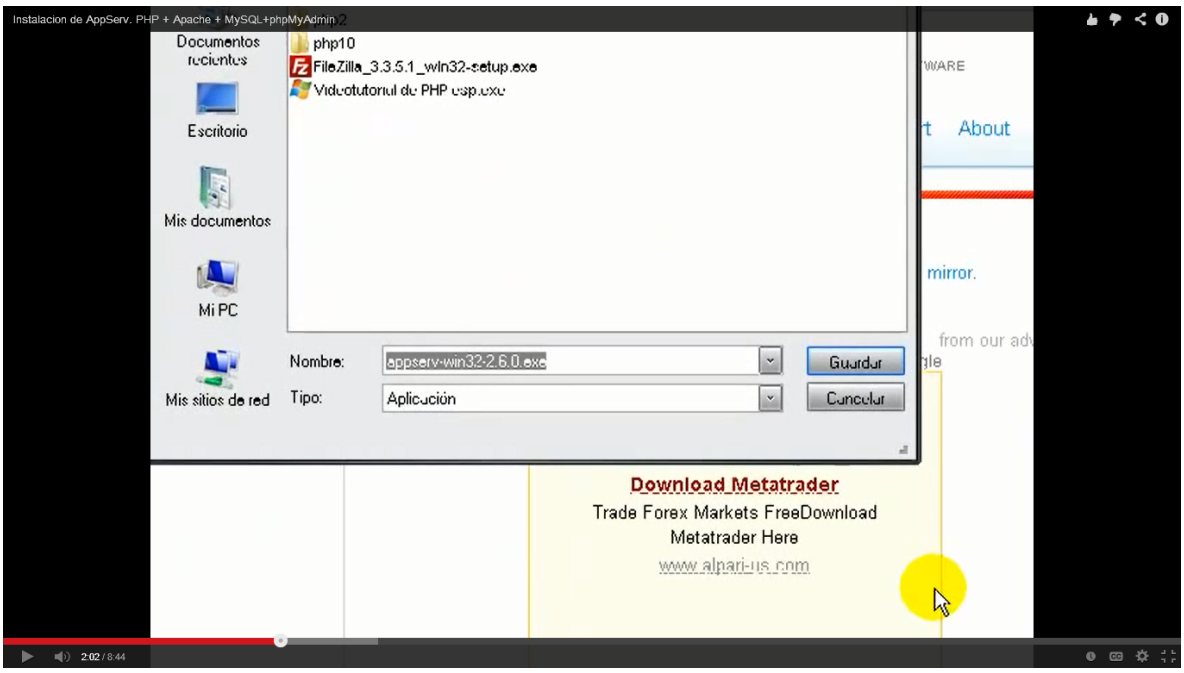
Instalaremos un programa llamado AppServ el cual contiene:
Apache, Php, MySQL, PHPMyAdmin

Nos iremos a la página de Appserv y escogeremos según si es nuestra máquina de 32 bits o 64 bits y también dependiendo de que gama queremos los programas en este caso descargaremos el Appserve de 2.6.0 de 64 bits.

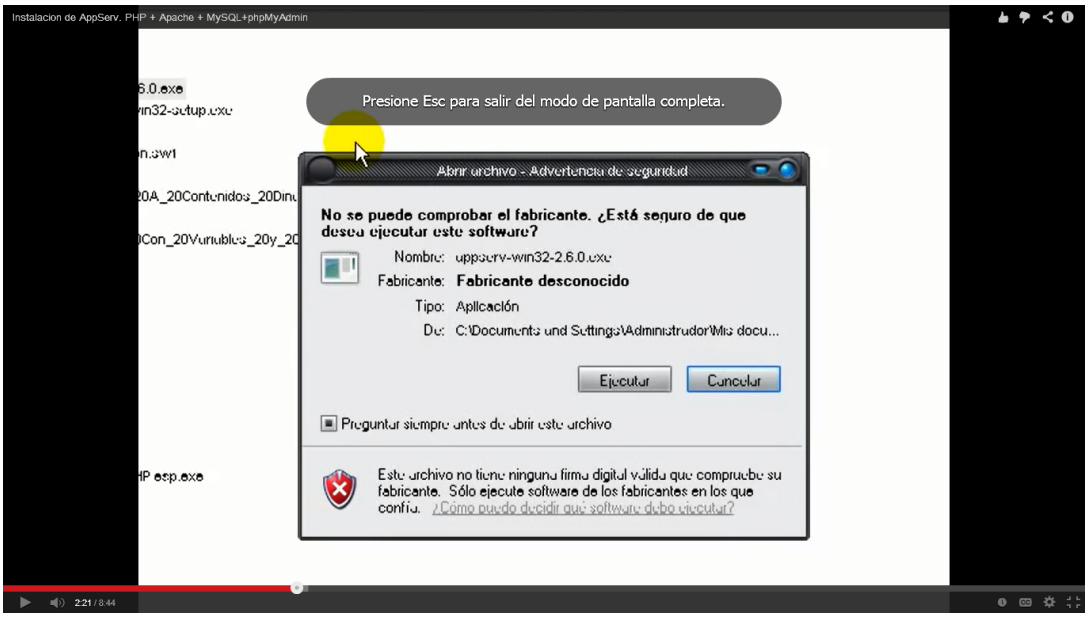
Proseguiremos a descargarlo y aparecerá lo siguiente. Y le daremos clic en el cuadro amarillo



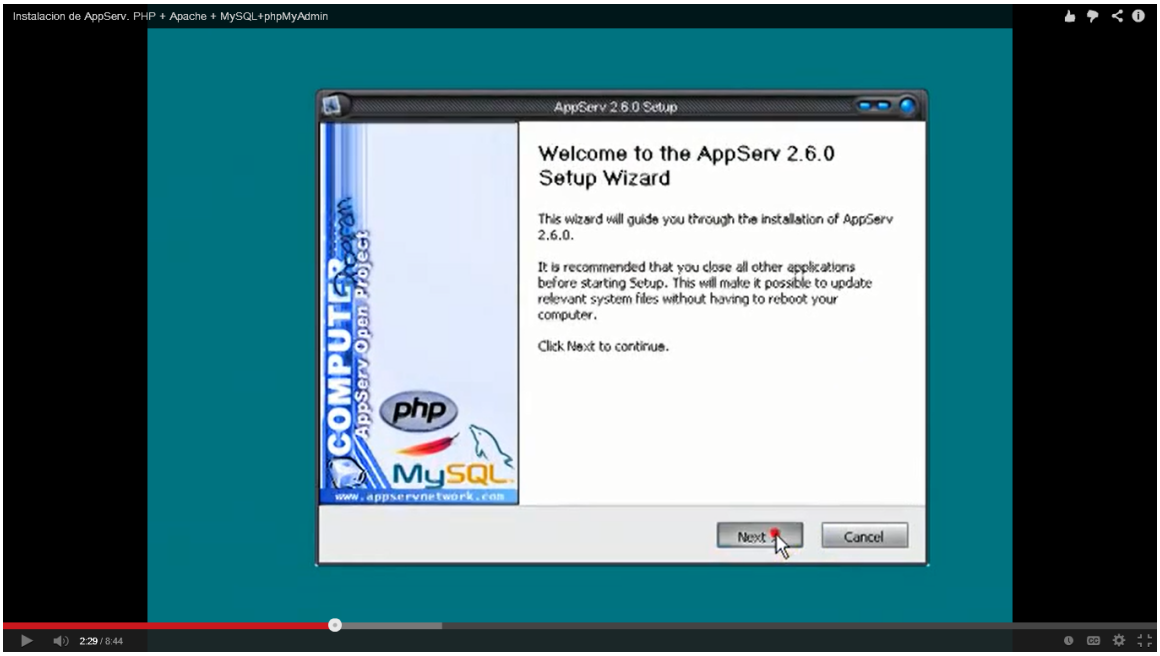
Después aparecerá un cuadro y le pondremos donde lo queremos guardar



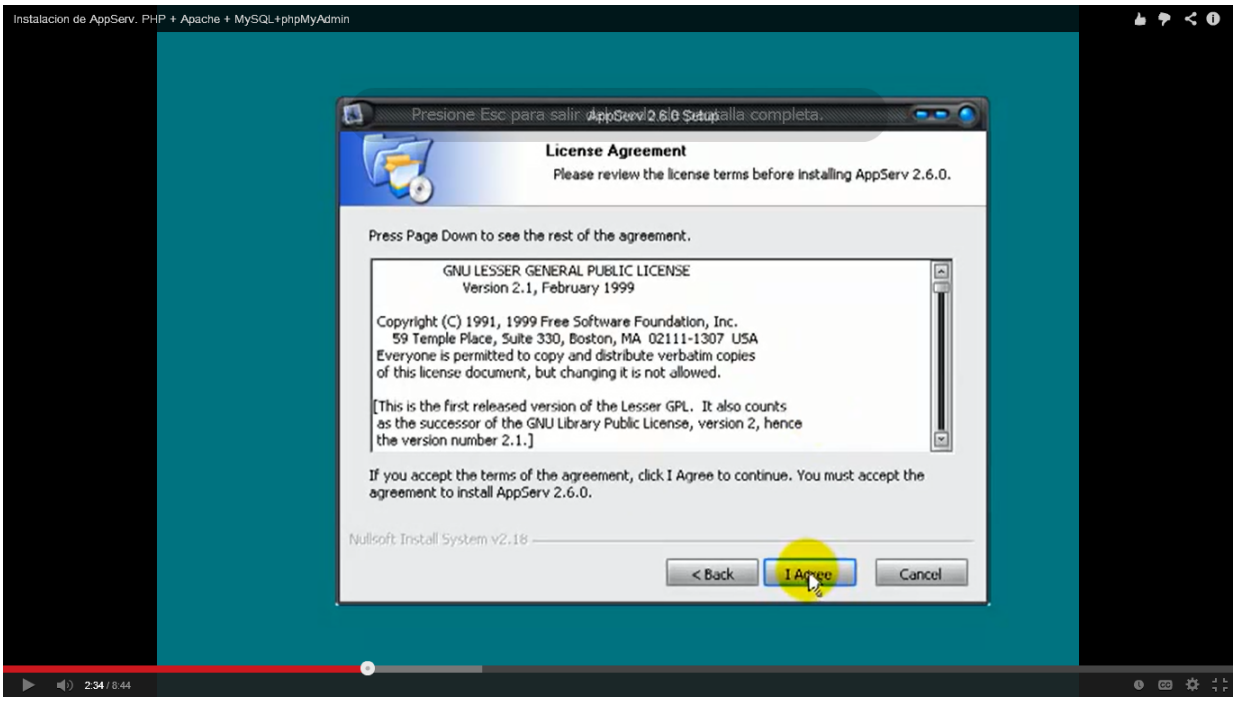
Después de descargarse completo iremos a la carpeta donde lo guardamos y le damos doble clic al lado derecho apareciendo un cuadro después y le damos clic en ejecutar



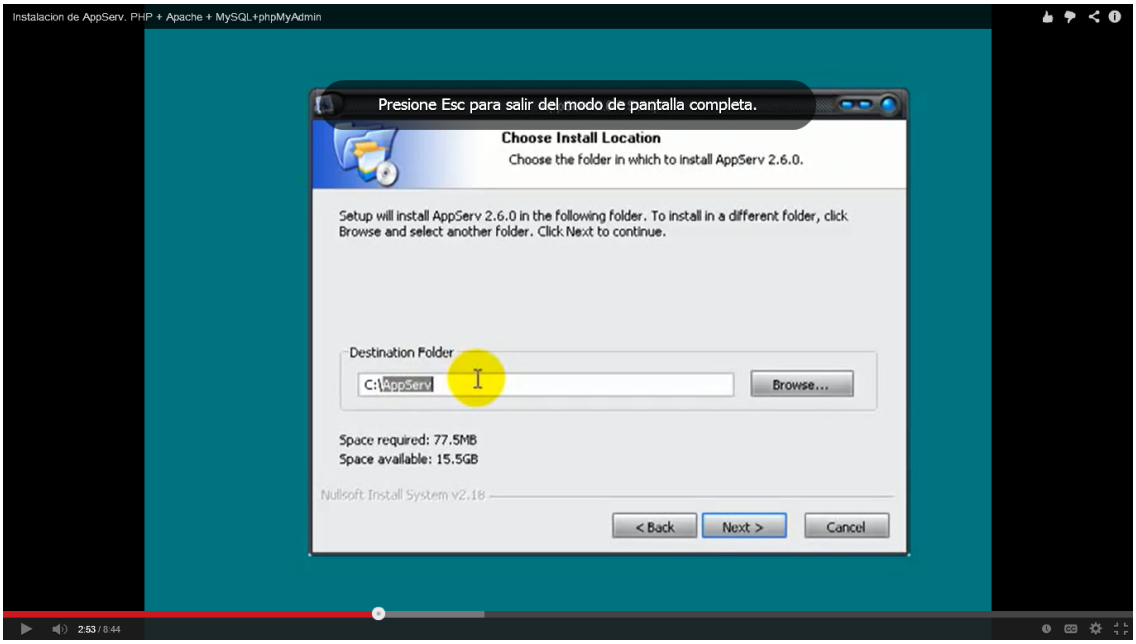
Aparecerá una ventana como la siguiente y le damos en next



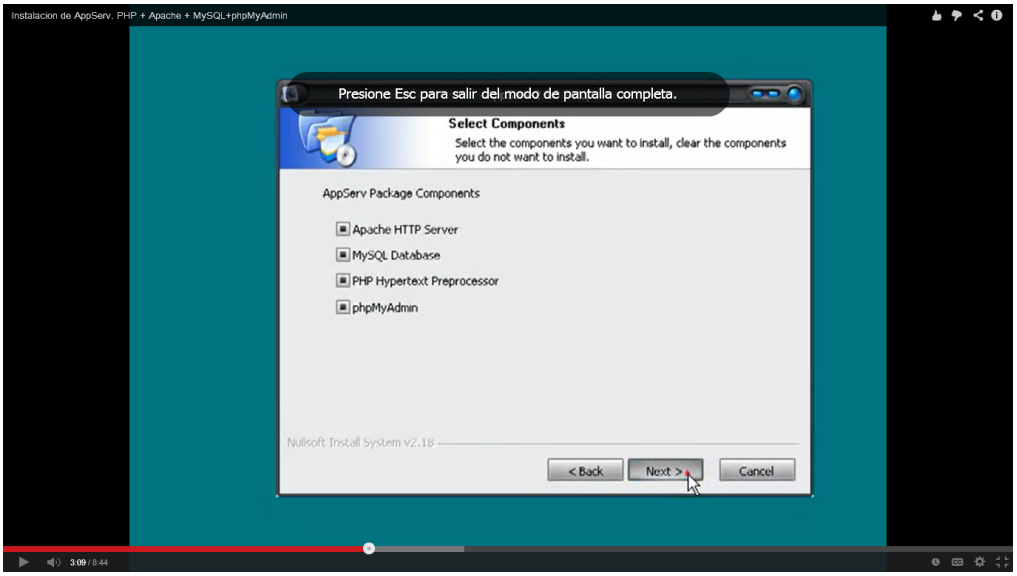
Después aceptamos los términos y condiciones dándole clip en I Agree



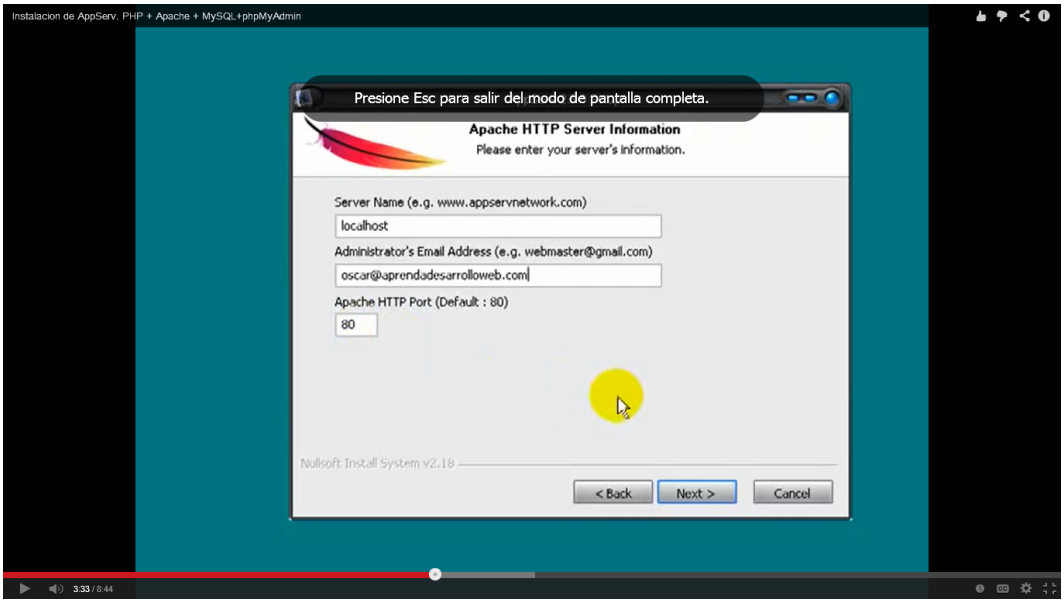
Después aparecerá un cuadro donde nos dirá en que parte guardará lo instalado que será en el disco C en una carpeta que creo llamada Appcer y le damos clip en next



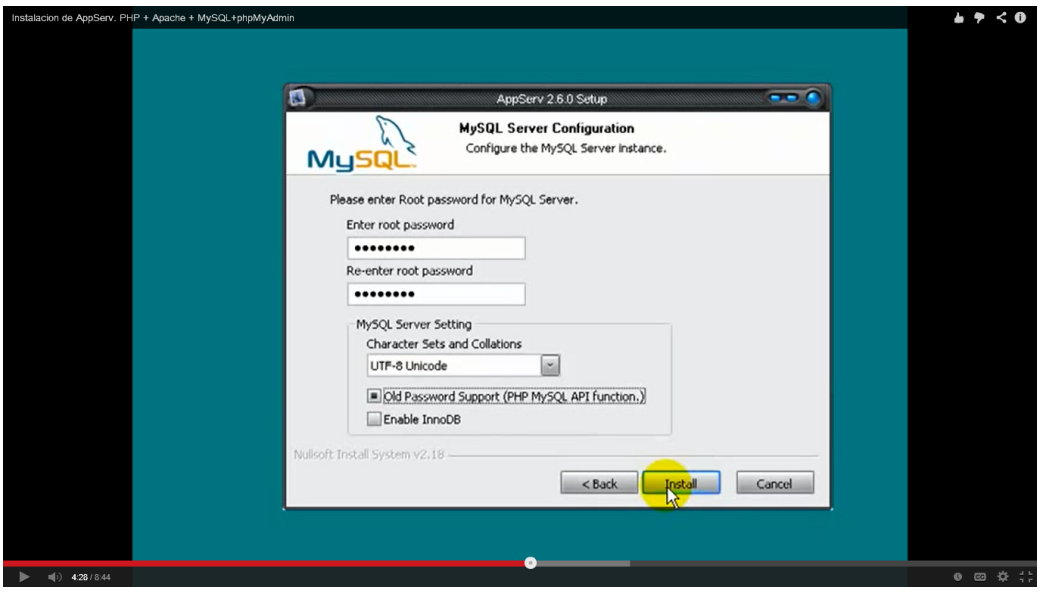
Luego aparecerá un cuadro donde saldrán los distintos programas de paquete que vamos utilizar en este caso los dejamos todos y le damos clip en next



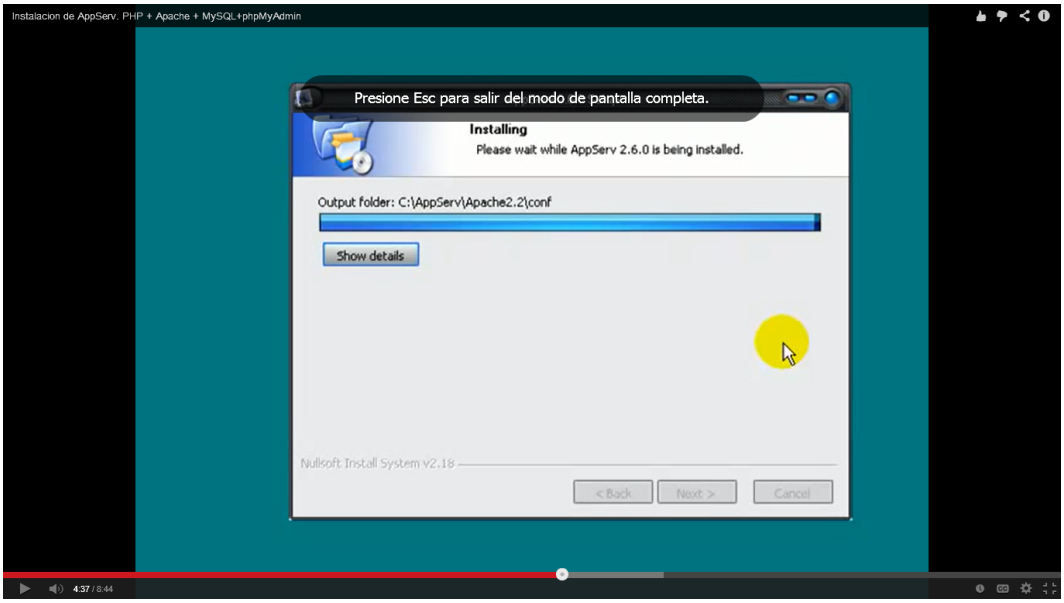
Después nos pedirá el nombre del servidor que le pondremos localhost y un correo de servidor que ese lo pueden escoger ustedes y le damos clip en next



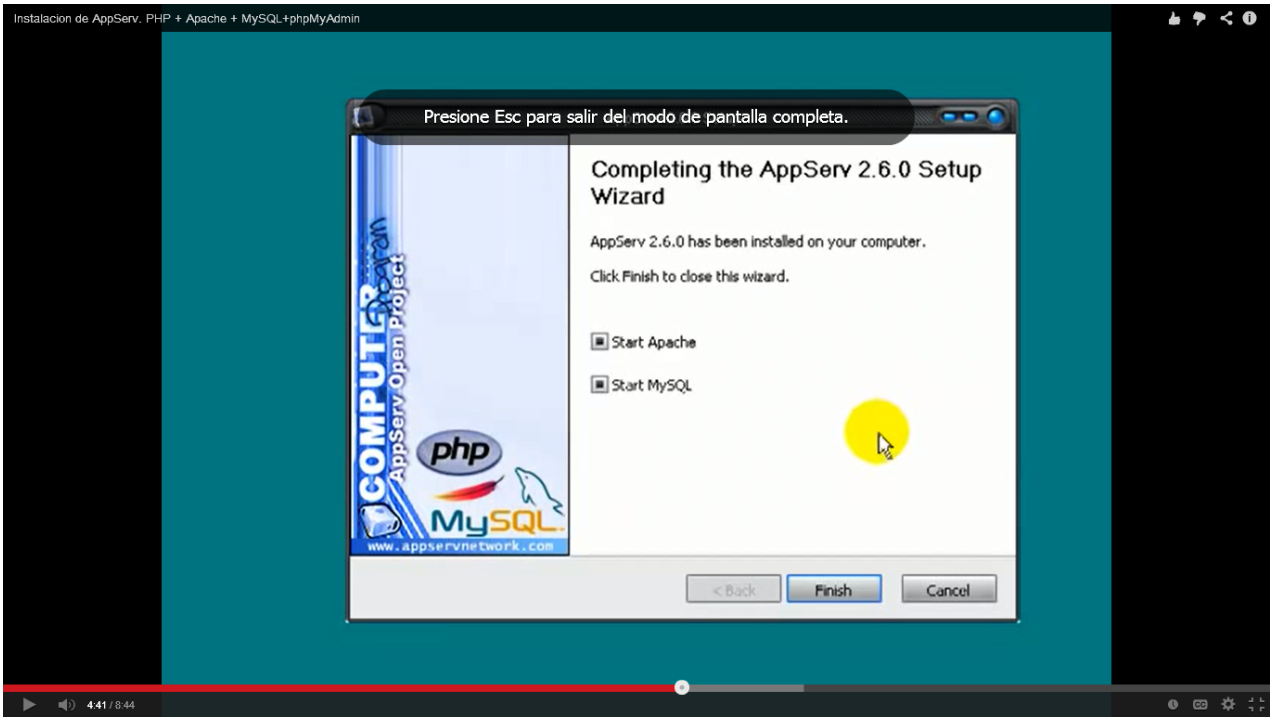
Después nos pedirá el password, debemos tener mucho cuidado con este ya que para entrar a alguna página ya no lo pedirá, lo ponemos 2 veces y le damos palomita en la opción a la primera de las 2 opciones que se nos da. Le damos clip en instalar.



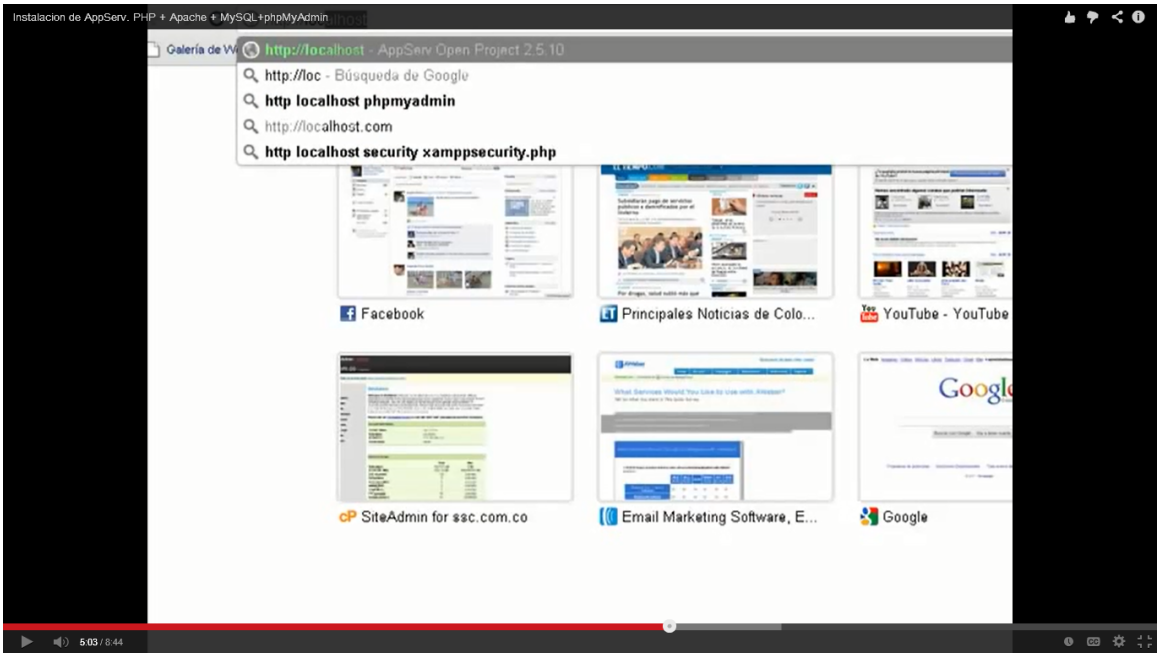
Prosiguiendo a la instalación del paquete



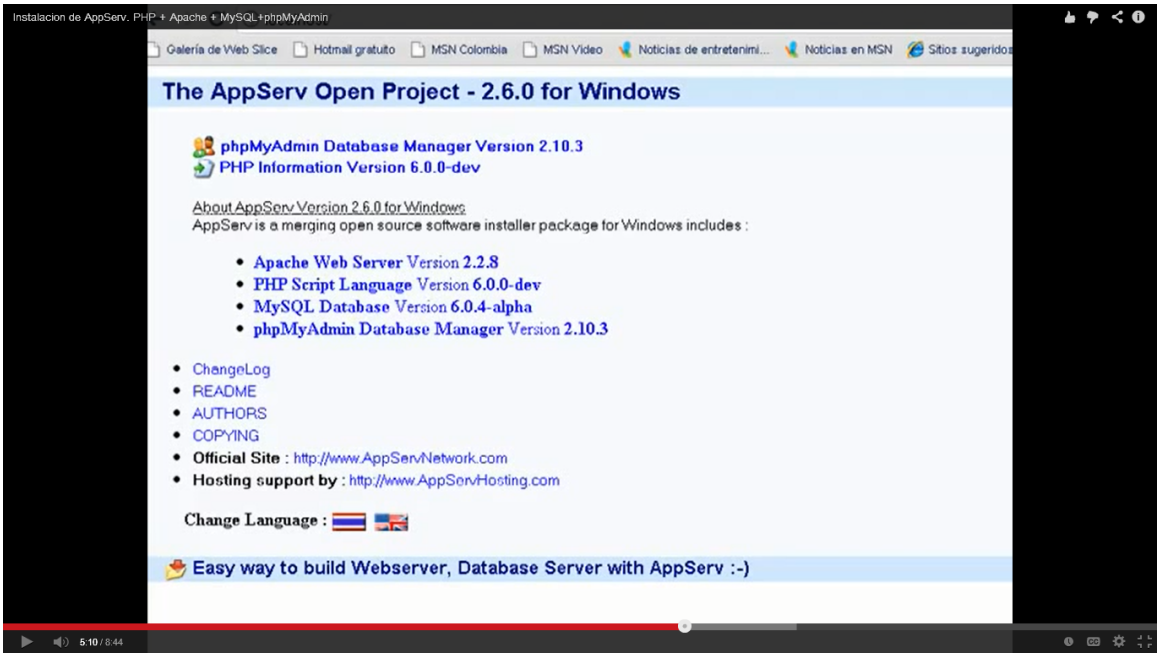
Nos mostrará en la pantalla que ya se instalo el paquete y le damos en finish



Y listo, esta instalado PHP. Para verificar que se instalo correctamente nos vamos al navegador y ponemos localhost.



Y si se instalo correctamente debe aparecer lo siguiente que indica que versión de PHP se instalo.



PHP

HISTORIA DEL PHP

PHP tal y como se conoce hoy en día es en realidad el sucesor de un producto llamado PHP/FI. Creado en 1994 por Rasmus Lerdorf, la primera encarnación de PHP era un conjunto simple de ficheros binarios Common Gateway Interface (CGI) escritos en el lenguaje de programación C. Originalmente utilizado para rastrear visitas de su currículum online, llamó al conjunto de scripts "Personal Home Page Tools", más frecuentemente referenciado como "PHP Tools". Con el paso del tiempo se quiso más funcionalidad, y Rasmus reescribió PHP Tools, produciendo una implementación más grande y rica. Este nuevo modelo fue capaz de interaccionar con bases de datos, y mucho más, proporcionando un entorno de trabajo sobre cuyos usuarios podían desarrollar aplicaciones web dinámicas sencillas tales como libros de visitas. En junio de 1995, Rasmus publicó el codigo fuente de PHP Tools, lo que permitió a los desarrolladores usarlo como considerasen apropiado. Esto también permitió -y animó- a los usuarios a proporcionar soluciones a los errores del código, y generalmente a mejorarlo.

En septiembre de ese mismo año, Rasmus amplió PHP y -por un corto periodo de tiempo- abandonó el nombre de PHP. Ahora, refiriéndose a las herramientas como FI (abreviatura de "Forms Interpreter"), la nueva implementación incluía algunas de las funciones básicas de PHP tal y como la conocemos hoy. Tenía variables como las de Perl, interpretación automática de variables de formulario y sintaxis incrustada HTML. La sintaxis por sí misma era similar a la de Perl, aunque mucho más limitada, simple y algo inconsistente. De hecho, para embeber el código en un fichero HTML, los desarrolladores tenían que usar comentarios de HTML. Aunque este método no era completamente bien recibido, FI continuó gozando de expansión y aceptación como una herramienta CGI --- pero todavía no

completamente como lenguaje. Sin embargo, esto comenzó a cambiar al mes siguiente, en octubre de 1995 Rasmus publicó una versión nueva del código. Recordando el nombre PHP, ahora era llamado (resumidamente) "Personal Home Page Construction Kit," y fue la primera versión que presumía de ser, en aquel momento, considerada como una interfaz de scripts avanzada. El lenguaje fue deliberadamente diseñado para asemejarse a C en estructura, haciéndolo una adopción sencilla para desarrolladores familiarizados con C, Perl, y lenguajes similares. Habiendo sido así bastante limitado a sistemas UNIX y compatibles con POSIX, el potencial para una implementación de Windows NT estaba siendo explorada.

El código fue completamente rehecho de nuevo, y en abril de 1996, combinando los nombres de versiones anteriores, Rasmus introdujo PHP/FI. Esta implementación de segunda generación comenzó realmente a desarrollar PHP desde un conjunto de herramientas dentro de un lenguaje de programación de derecho propio. Incluía soporte interno para DBM, mSQL, y bases de datos Postgres95, cookies, soporte para funciones definidas por el usuario, y mucho más. Ese mes de junio, PHP/FI brindó una versión 2.0. Sin embargo, un interesante hecho sobre esto, es que sólo había una única versión completa de PHP 2.0. Cuando finalmente pasó de la versión beta en noviembre de 1997, el motor de análisis subyacente ya estaba siendo reescrito por completo.

Aunque vivió una corta vida de desarrollo, continuó gozando de un crecimiento de popularidad en el aún joven mundo del desarrollo. En 1997 y 1998, PHP/FI tenía un culto de varios miles de usuarios en todo el mundo. Una encuesta de Netcraft en mayo de 1998 indicó que cerca de 60,000 dominios reportaron que tenían cabeceras que contenían "PHP", indicando en efecto que el servidor host lo tenía instalado. Este número se correspondía con aproximadamente el 1% de todos los dominios de Internet del momento. A pesar de estas impresionantes cifras, la maduración de PHP/FI estaba condenada por limitaciones; mientras habían varios contribuidores menores, aún era desarrollado principalmente por un individuo.

Ejemplo #1 Ejemplo de Código PHP/FI

```
<!--include /text/header.html-->
<!--getenv HTTP_USER_AGENT-->
<!--ifsubstr $exec_result Mozilla-->
  Hey, ¡está usando Netscape!<p>
<!--endif-->
<!--sql database select * from table where user='$username'-->
<!--ifless $numentries 1-->
  Lo siento, esta entrada no existe<p>
<!--endif exit-->
  Bienvenido <!--$user-->!<p>
  Le quedan <!--$index:0--> créditos en su cuenta.<p>
<!--include /text/footer.html-->
```

PHP 3

PHP 3.0 fue la primera versión que más se parecía al PHP que existe hoy. Encontrando todavía PHP/FI 2.0 ineficiente y falto de las características que necesitaban para impulsar una aplicación de comercio electrónico que estaban desarrollando para un proyecto de universidad, Andi Gutmans y Zeev Suraski, de Tel Aviv, Israel, comenzaron otra nueva versión del analizador subyacente en 1997. Proponiendo Rasmus online, discutieron varios aspectos de la implementación actual y su red desarrollo de PHP. En un esfuerzon para mejorar el motor y comenzar a construir sobre la base de usuario de PHP/FI existente, Andi, Rasmus y Zeev decidieron colaborar en el desarrollo de un nuevo e independiente lenguaje de programación. Este lenguaje completamente nuevo fue publicado bajo un nuevo nombre, que eliminó la implicación del uso limitado personal que el nombre PHP/FI tenía. Fue renombrado simplemente como 'PHP', con el significado de un acrónimo recursivo - PHP: Hypertext Preprocessor.

Una de las mejores características de PHP 3.0 era su gran extensibilidad. Además de proveer a los usuarios finales de una interfaz madura para múltiplies bases de datos, protocolos, y APIs, la sencillez de ampliar el lenguaje mismo atrajo a docenas de desarrolladores que presentaron variedad de módulos. Podría decirse que esta fue la clave para el tremendo éxito de PHP 3.0. Otras características clave introducidas en PHP 3.0 incluían el soporte para programación orientada a objetos y una sintaxis de lenguaje mucho más potente y consistente.

En junio de 1998, con muchos nuevos desarrolladores de todo el mundo unidos al esfuerzon, PHP 3.0 fue anunciado por el nuevo Equipo de Desarrollo de PHP como el sucesor oficial de PHP/FI 2.0. El desarrollo activo de PHP/FI 2.0, que estaba casi parado desde noviembre del año anterior, fue oficialmente finalizado. Después de aproximadamente nueve meses de pruebas públicas, cuando el anuncio de la versión oficial de PHP 3.0 vino, ya estaba instalado en más de 70,000 dominios de todo el mundo, y ya no estaba limitado a sistemas operativos compatibles con POSIX. Una relativamente pequeña parte de de los dominios que tenían instalado PHP estaban albergados en servidores que ejecutaban Windows 95, 98, y NT, y Macintosh. En este punto, PHP 3.0 estaba instalado en aproximadamente el 10% de los servidores web de Internet.

PHP 4

En el invierno de 1998, poco después del lanzamiento oficial de PHP 3.0, Andi Gutmans y Zeev Suraski comenzaron a trabajar en una nueva versión del núcleo de PHP. Los objetivos de diseño fueron mejorar la ejecución de aplicaciones complejas y mejorar la modularidad del código base de PHP. Estas aplicaciones se hicieron posibles por las nuevas características de PHP 3.0 y el apoyo de una gran variedad de bases de datos y APIs de terceros, pero PHP 3.0 no estaba diseñado para un mantenimiento tan complejo de aplicaciones eficientemente.

El nuevo motor, apodado 'Motor Zend' (proviene de sus nombres de pila, Zeev y Andi), alcanzó estos objetivos de diseño satisfactoriamente, y se introdujo por primera vez a mediados de 1999. PHP 4.0, basado en este motor, y asociado con un gran rango de nuevas características adicionales, fue oficialmente publicado en Mayo del 2000, casi dos años después que su predecesor. Además de la mejora de rendimiento de esta versión, PHP 4.0 incluía otras características clave como el soporte para la mayoría de los servidores Web, sesiones HTTP, buffers de salida, formas más seguras de controlar las entradas de usuario y muchas nuevas construcciones de lenguaje.

PHP 5

PHP 5 fué lanzado en Julio del 2004 después de un largo desarrollo y varios pre-releases. Está básicamente impulsado por su núcleo, *Zend Engine 2.0* que contiene un nuevo modelo de objetos y docenas de nuevas opciones.

El equipo de desarrollo de PHP incuye docenas de desarrolladores, así como docenas de otras personas trabajando en proyectos relacionados y de soporte para PHP, como PEAR, PECL, y documentación, y una infraestructura en red subyacente de más de cien servidores web individuales en seis de los siete continentes del mundo. Aunque es solo una estimación basada en estadísticas de años anteriores, es seguro suponer que PHP ahora está instalado en diez o quizá cien millones de dominios en todo el mundo.

¿QUE ES PHP?

PHP es un lenguaje potente de alto nivel cuyo código podemos introducir en páginas web HTML. PHP se ejecuta en el servidor, dicho servidor realiza tareas en beneficio de otros programas permitiendo al usuario almacenar y acceder a los archivos de la computadora (no podemos ejecutarlo en nuestro ordenador a no ser que lo hagamos funcionar como servidor). PHP, una vez que es interpretado por el

servidor, genera una salida HTML que permite visualizar los resultados en los navegadores. Este curso permite aprender los fundamentos para la creación de páginas web usando PHP.

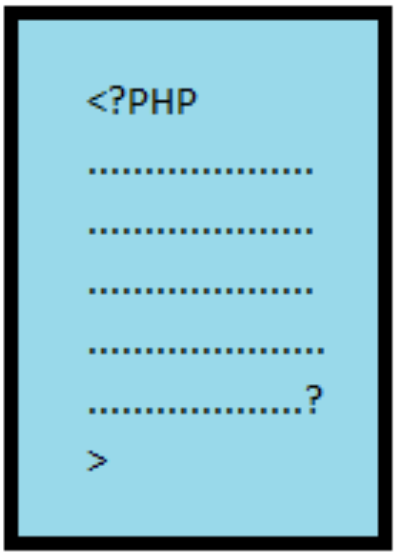
Ejemplo de la estructura básica de PHP.

```
<HTML>
    <head>
        <title>Example</title>
    </head>
    <body>
        <?PHP
            echo "¡Hola!";
        ?>
    </body>
</HTML>
```

Podemos ver que no es lo mismo que un script escrito en otro lenguaje de programación como Perl o C -- En vez de escribir un programa con muchos comandos para crear una salida en HTML, escribimos el código HTML con cierto código PHP embebido (introducido) en el mismo, que producirá cierta salida (en nuestro ejemplo, producir un texto). El código PHP se incluye entre etiquetas especiales de comienzo y final que nos permitirán entrar y salir del modo PHP.

Lo que distingue a PHP de la tecnología Javascript, la cual se ejecuta en la máquina cliente, es que el código PHP es ejecutado en el servidor. Si tuviésemos un script similar al de nuestro ejemplo en nuestro servidor, el cliente solamente recibiría el resultado de su ejecución en el servidor, sin ninguna posibilidad de determinar que código ha producido el resultado recibido. El servidor web puede ser incluso configurado para que procese todos los ficheros HTML con PHP.

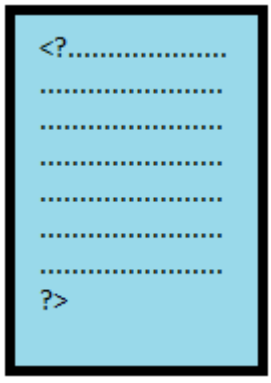
COMO INSCRUSTAR CODIGO EN PHP



Forma más recomendable

- Siempre disponible (no depende de la configuración del Servidor.
- Específica de PHP.
- Única que permite incrustar código PHP en XML y XHTML

Otras formas



Lo mejor de usar PHP es que es extremadamente simple para el principiante, pero a su vez, ofrece muchas características avanzadas para los programadores profesionales. Aunque el desarrollo de PHP está concentrado en la programación de scripts en la parte del servidor, se puede utilizar para muchas otras cosas.

PHP es un lenguaje, formado por un conjunto de librerías orientadas al desarrollo de aplicaciones web. En este entorno, es un lenguaje interpretado. El servidor web, cuando detecta una extensión asociada a php (.php, .php3, .phtml,...), envía esta página al intérprete y éste se la devuelve traducida a html. Se puede ejecutar php sin necesidad de servidor desde la línea de comandos. También existen compiladores, como el desarrollado por Zend, para ejecutar los scripts php a modo de aplicaciones "stand alone", sin necesidad de intérprete.

¿QUÉ HACER CON PHP?

PHP puede hacer cualquier cosa que se pueda hacer con un script CGI, como procesar la información de formularios, generar páginas con contenidos dinámicos, o mandar y recibir cookies. Y esto no es todo, se puede hacer mucho más.

Existen tres campos en los que scripts escritos en PHP son usados.

- Scripts en la parte del servidor. Este es el campo más tradicional y el principal campo de trabajo. Se necesitan tres cosas para que esto funcione. El paseador PHP (CGI o módulo), un servidor web y un navegador. Se necesita correr el servidor web con PHP instalado. El resultado del programa PHP se puede obtener a través del navegador, conectando con el servidor web. Consultar la sección Instrucciones de instalación para más información.

- Scripts en línea de comandos. Podéis crear un script PHP y correrlo sin ningún servidor web o navegador. Solamente necesitáis el parseador PHP para usarlo de esta manera. Este tipo de uso es ideal para scripts ejecutados regularmente desde cron (en *nix ó Linux) o el planificador de tareas (en Windows). Estos scripts también pueden ser usados para tareas simples de

procesado de texto. Consultar la sección usos de PHP en la línea de comandos para más información.

- Escribir aplicaciones grecas de clientes. PHP no es probablemente el mejor lenguaje para escribir aplicaciones grecas, pero si sabéis bien PHP, y os gustaría utilizar algunas características avanzadas en programas clientes, podéis utilizar PHP-GTK para escribir dichos programas. Es también posible escribir aplicaciones independientes de una plataforma. PHP-GTK es una extensión de PHP, no disponible en la distribución principal. Si te interesa PHP-GTK, puedes visitar las páginas web del proyecto (http://gtk.php.net/).

PHP puede ser utilizado en cualquiera de los principales sistemas operativos del mercado, incluyendo Linux, muchas variantes Unix (incluido HP-UX, Solaris y OpenBSD), Microsoft Windows, Mac OS X, RISC OS y probablemente alguno más. PHP soporta la mayoría de servidores web de hoy en día, incluyendo Apache, Microsoft Internet Información Server, Personal Web Server, Netscape y iPlanet, Oreilly Website Pro server, Caudium, Xitami, OmniHTTPd y muchos otros. PHP tiene módulos disponibles para la mayoría de los servidores, para aquellos otros que soporten el estándar CGI, PHP puede usarse como procesador CGI.

PROGRAMAS SELECTOS

- Suma de 5 números

```
<?PHP
    $a=2;
    $b=3;
    $c=1;
    $d=7;
    $e=5;
    $s=$a+$b+$c+$d+$e;
    echo"suma=$s"
```

```
?>
```

- Sacar x^2,x^3 y x^(-1)

```
<?PHP
      $x=18;
      $a=$x*$x;
      $b=$x*$x*$x;
      $c=1/$x;
      echo"x^2=$a, x^3=$b y 1/x=$c"
?>
```

- Promedio entre 10 números

```
<?PHP
    $a=2;
    $b=3;
    $c=1;
    $d=7;
    $e=5;
    $f=6;
    $g=4;
    $h=26;
    $j=9;
    $n=16;
   $s=$a+$b+$c+$d+$e+$f+$g+$h+$j+$n;
    $m=$s/10;
    echo"media=$m"
?>
```

- Volumen de un cilindro

```
<?PHP
   $r=2;
   $p=3.1416;
   $h=5;
```

```
    $a=$r*$r*$p;
    echo"Area de la base $a y su altura $h"
?>
 <br/>
<?PHP
    $e=$a*$h;
    echo"volumen del cilindro v=$e"
?>
```

- Volumen de un cono

```
<?PHP
    $p=3.1416;
    $r=3;
    $h=12;
    $b=($p*$r*$r*$h)/3;
    echo"volumen de un cono con radio=$r y altura=$h v=$b"
?>
```

- Volumen de una pirámide

```
<?PHP
    $x=2;
    $a=$x*$x;
    $h=4;
    $b=($a*$h)/3;
    echo"volumen de una pirámide cuadrada v=$b"
?>
```

- De centígrados a Fahrenheit

```
<?PHP
    $p=3.1416;
    $r=3;
    $h=12;
    $b=($p*$r*$r*$h)/3;
```

```
    echo"volumen de un cono con radio=$r y altura=$h v=$b"
?>
```

- De Fahrenheit a centígrados

```
<?PHP
    $f=23;
    $c=($f-32)*(5/9);
    echo “convertir de fahrenheit a grados=$c"
?>
```

- De galones a litros

```
<?PHP
    $g=23;
    $l=$g*3.785;
    echo"23 litros son en galones=$l"
?>
```

- De litros a galones

```
<?PHP
    $l=23;
    $g=$l/3.785;
    echo"23 litros son en galones=$g"
?>
```

- De metros a pulgadas

```
<?PHP
    $x=2;
    $a=$x*$x;
    $h=4;
    $b=($a*$h)/3;
    echo"volumen de una pirámide cuadrada v=$b"
?>
```

- De pulgadas a metros

```
<?PHP
```

```
    $p=23;
    $m=100/($p*2.54);
    echo"23 pulgadas en metros son   metros=$m"
?>
```

- Factorial

```
<?PHP
    echo"factorial del 3"
?>
<br/>
<?PHP
    $a=3;
    $f=$a*($a-1)*($a-2);
    echo"factorial=$f"
?>
```

- Decir si el número es par o impar

```
<?PHP
    $numero=5;
    if ($numero%2==0) {
       echo"el número es par";
     }
     else {
       echo"el número es impar";
     }
?>
```

- Primeros 10 números impares que partan del 3

```
<?PHP
    $numero=5;
    if ($numero%2==0) {
        echo"el número es par";
    }
```

```
    else {
        echo"el número es impar";
    }
?>
```

- Raíz positivas

```
<?PHP
    $a=1;
    $b=3;
    $c=1;
    $x1=(-$b+(sqrt($b*$b-4*$a*$c)))/(2*$a);
    $x2=(-$b-(sqrt($b*$b-4*$a*$c)))/(2*$a);
    echo"raices de x^2+3x+1 x1=$x1 y x2=$x2"
?>
```

- Pendiente de 2 puntos

```
<?PHP
    echo"pendiente de los puntos a=(5,3) y b=(1,2)"
?>
<br/>
<?PHP
    $a=5;
    $b=3;
    $c=1;
    $d=2;
    $p=($d-$b)/($c-$a);
    echo"pendiente=$p"
?>
```

- Raíces negativas

```
<?PHP
    $a=1;
    $b=3;
```

```
    $c=1;
    $x1=(-$b+(sqrt($b*$b-4*$a*$c)))/(2*$a);
    $x2=(-$b-(sqrt($b*$b-4*$a*$c)))/(2*$a);
    echo"raices de x^2+3x+1 x1=$x1 y x2=$x2"
?>
```

- Distancia entre 2 números

```
<?PHP
    $a=2;
    $b=3;
    $c=1;
    $e=7;
    $x=9;
    $y=3;
    $z=($x-$y)*($x-$y);
    $g=($a-$c)*($a-$c);
    $h=($b-$e)*($b-$e);
    $i=$g+$h+$z;
    $d=sqrt($i);
    echo"distancia=$d"
?>
```

- Interés

```
<?PHP
echo"si el interes es anual";
 $deposito=1000;
 $interes=12;
 $tasa=1/12;
 $mes=12/100;
$dinero=$deposito+$deposito*($interes/100);
  echo" ganancia en 1 año es $dinero";
```

```
$dinero1=$mes*$tasa*$deposito+$deposito;
$dinero2=$dinero1*$tasa*$mes+$dinero1;
$dinero3=$dinero2*$tasa*$mes+$dinero2;
$dinero4=$dinero3*$tasa*$mes+$dinero3;
$dinero5=$dinero4*$tasa*$mes+$dinero4;
$dinero6=$dinero5*$tasa*$mes+$dinero5;
$dinero7=$dinero6*$tasa*$mes+$dinero6;
$dinero8=$dinero7*$tasa*$mes+$dinero7;
$dinero9=$dinero8*$tasa*$mes+$dinero8;
$dinero10=$dinero9*$tasa*$mes+$dinero9;
$dinero11=$dinero10*$tasa*$mes+$dinero10;
$dinero12=$dinero11*$tasa*$mes+$dinero11;
?>
<br/>
  <?echo"por mes tendriamos los siguientes resultados";?><br/>
  <?echo"1 mes $dinero1";?><br/>
  <?echo"2 mes $dinero2";?><br/>
  <?echo"3 mes $dinero3";?><br/>
  <?echo"4 mes $dinero4";?><br/>
  <?echo"5 mes $dinero5";?><br/>
  <?echo"6 mes $dinero6";?><br/>
  <?echo"7 mes $dinero7";?><br/>
  <?echo"8 mes $dinero8";?><br/>
  <?echo"9 mes $dinero9";?><br/>
  <?echo"10 mes $dinero10";?><br/>
  <?echo"11 mes $dinero11";?><br/>
  <?echo"12 mes $dinero12";?><br/>
?>
```

- Tabla del 2

```
<?php
  $x=2*1;
```

```
  echo"2x1=$x"
?>
<br/>
<?php
  $x=2*2;
  echo"2x2=$x"
?>
<br/>
<?php
 $x=2*3;
 echo"2x3=$x"
?>
<br/>
<?php
 $x=2*4;
 echo"2x4=$x"
?>
<br/>
<?php
 $x=2*5;
 echo"2x5=$x"
?>
<br/>
<?php
 $x=2*6;
 echo"2x6=$x"
?>
<br/>
<?php
 $x=2*7;
 echo"2x7=$x"
```

```
    ?>
    <br/>
    <?php
      $x=2*8;
      echo"2x8=$x"
    ?>
    <br/>
    <?php
      $x=2*9;
      echo"2x9=$x"
    ?>
    <br/>
    <?php
      $x=2*10;
      echo"2x10=$x"
?>
```

VARIABLES Y CONSTANTES

DEFINICIÓN DE VARIABLE

En PHP las variables se representan como un signo de dólar seguido por el nombre de la variable. El nombre de la variable es sensible a minúsculas y mayúsculas.

```
$var = "Bob";
$Var = "Joe";
echo "$var, $Var";
// produce la salida "Bob, Joe"
```

En PHP3, las variables siempre se asignan por valor. Esto significa que cuando se asigna una expresión a una variable, el valor íntegro de la expresión original se copia en la variable de destino. Esto quiere decir que, por ejemplo, después de asignar el valor de una variable a otra, los cambios que se efectúen a una de esas variables no afectará a la otra.

Variables predefinidas

PHP proporciona una gran cantidad de variables predefinidas a cualquier script que se ejecute. De todas formas, muchas de esas variables no pueden estar completamente documentadas ya que dependen de sobre qué servidor se esté ejecutando, la versión y configuración de dicho servidor, y otros factores.

- Variables de Apache
- Variables de entorno
- Variables de PHP

DEFINICIÓN DE CONSTANTE

Una constante es un identificador para expresar un valor simple. Como el nombre sugiere, este valor no puede variar durante la ejecución del script (las constantes especiales __FILE__ y __LINE__ son una excepción a esto, ya que actualmente no lo son). Una constante es sensible a mayúsculas por defecto. Por convención, los identificadores de constantes suelen declararse en mayúsculas.

Estas son las diferencias entre constantes y variables:

- Las constantes no son precedidas por un símbolo de dólar ($)

- Las constantes solo pueden ser definidas usando la función ( ) define, nunca por simple asignación.
- Las constantes pueden ser definidas y accedidas sin tener en cuenta las reglas de alcance del ámbito.
- Las constantes no pueden ser redefinidas o eliminadas después de establecerse; y
- Las constantes solo puede albergar valores escalares.

TIPOS DE VARIABLES Y OPERADORES

El tipo de una variable normalmente no lo indica el programador; en su lugar, lo decide PHP en tiempo de ejecución dependiendo del contexto en el que se utilice esa variable.

Si se quisiese obligar a que una variable se convierta a un tipo concreto, se podría forzar la variable o usar la función settype ( ) para ello.

Nótese que una variable se puede comportar de formas diferentes en ciertas situaciones, dependiendo de qué tipo sea en ese momento.

Enteros

Los enteros se puede especificar usando una de las siguientes sintaxis:

```
$a = 1234; # número decimal
$a = -123; # un número negativo
$a = 0123; # número octal (equivalente al 83 decimal)
$a = 0x12; # número hexadecimal (equivalente al 18 decimal).
```

Números en punto flotante

Los números en punto flotante ("double") se pueden especificar utilizando cualquiera de las siguientes sintaxis:

$a = 1.234; $a = 1.2e3;

Cadenas

Las cadenas de caracteres se pueden especificar usando uno de dos tipos de delimitadores.

Si la cadena está encerrada entre dobles comillas ("), las variables que estén dentro de la cadena serán expandidas (sujetas a ciertas limitaciones de interpretación). Como en C y en Perl, el carácter de barra invertida ("\") se puede usar para especificar caracteres especiales:

Caracteres protegidos

\n	Nueva línea
\r	Retorno de carro
\t	Tabulación horizontal
\\	Barra invertida
\$	Signo del dólar
\"	Comillas dobles
\[0-7]{1,3}	la secuencia de caracteres que coincida con la expresión regular es un carácter en notación octal
\x[0-9A-Fa-f]{1,2}	la secuencia de caracteres que coincida con la expresión regular es un carácter en notación hexadecimal

La segunda forma de delimitar una cadena de caracteres usa el carácter de comilla simple ("'"). Cuando una cadena va encerrada entre comillas simples, los únicos caracteres de escape que serán comprendidos son "\\" y "\'". Esto es por convenio, así que se pueden tener comillas simples y barras invertidas en una cadena entre comillas simples. Las variables no se expandirán dentro de una cadena entre comillas simples.

Otra forma de delimitar cadenas es usando la sintaxis de documento incrustado ("<<<"). Se debe proporcionar un identificador después de <<<, después la cadena, y después el mismo identificador para cerrar el entrecomillado.

Conversión de cadenas

Cuando una cadena se evalúa como un valor numérico, el valor resultante y el tipo se determinan como sigue.

La cadena se evaluará como un doble si contiene cualquiera de los caracteres '.', 'e', o 'E'. En caso contrario, se evaluará como un entero.

El valor viene dado por la porción inicial de la cadena. Si la cadena comienza con datos de valor numérico, este será el valor usado. En caso contrario, el valor será 0 (cero). Los datos numéricos válidos son un signo opcional, seguido por uno o más dígitos (que opcionalmente contengan un punto decimal), seguidos por un exponente opcional. El exponente es una 'e' o una 'E' seguidos por uno o más dígitos.

Cuando la primera expresión es una cadena, el tipo de la variable dependerá de la segunda expresión.

$foo = 1 + "10.5"; // $foo es doble (11.5) $foo = 1 + "-1.3e3"; // $foo es doble (-1299) $foo = 1 + "bob-1.3e3"; // $foo es entero (1) $foo = 1 + "bob3"; // $foo es

entero (1) $foo = 1 + "10 Cerditos"; // $foo es entero (11) $foo = 1 + "10 Cerditos"; // $foo es entero (11) $foo = "10.0 cerdos " + 1; // $foo es entero (11) $foo = "10.0 cerdos " + 1.0; // $foo es double (11)

Para más información sobre esta conversión, mire en la página del manual de Unix strtod (3).

Si quisiera probar cualquiera de los ejemplos de esta sección, puede cortar y pegar los ejemplos e insertar la siguiente línea para ver por sí mismo lo que va ocurriendo:

echo "\$foo==$foo; el tipo es " . gettype($foo) . "
\n";

Arrays

Los arrays actualmente actúan tanto como tablas hash (arrays asociativos) como arrays indexados (vectores).

Arrays unidimensionales

PHP soporta tanto arrays escalares como asociativos. De hecho, no hay diferencias entre los dos. Se puede crear una array usando las funciones list ( ) o array ( ), o se puede asignar el valor de cada elemento del array de manera explícita.

$a [0] = "abc"; $a [1] = "def"; $b ["foo"] = 13;

También se puede crear un array simplemente añadiendo valores al array. Cuando se asigna un valor a una variable array usando corchetes vacíos, el valor se añadirá al final del array.

$a [ ] = "hola"; // $a [2] == "hola" $a [ ] = "mundo"; // $a [3] == "mundo"

Los arrays se pueden ordenar usando las funciones asort ( ), arsort ( ), ksort ( ), rsort ( ), sort ( ), uasort ( ), usort ( ), y uksort ( ) dependiendo del tipo de ordenación que se desee.

Se puede contar el número de elementos de un array usando la función count ( ).

Se puede recorrer un array usando las funciones next ( ) y prev ( ). Otra forma habitual de recorrer un array es usando la función each ( ).

Arrays Multidimensionales

Los arrays multidimensionales son bastante simples actualmente. Para cada dimensión del array, se puede añadir otro valor [clave] al final:

```
$a [1] = $f; # ejemplos de una sola dimensión $a ["foo"] = $f;
$a [1] [0] = $f; # bidimensional
$a ["foo"] [2] = $f; # (se pueden mezclar índices numéricos y asociativos)
$a [3] ["bar"] = $f; # (se pueden mezclar índices numéricos y asociativos)
$a ["foo"] [4] ["bar"] [0] = $f; # tetradimensional!
```

Inicialización de Objetos

Para inicializar un objeto, se usa la sentencia new para instanciar el objeto a una variable.

```
class foo {
    function do_foo ( ) {
    echo "Doing foo.";
```

```
    }
}
$bar = new foo;
$bar->do_foo ( );
```

Booleanas

Nos referimos a variables booleanas a aquellas que utilizamos para indicar que algo es verdadero o falso.

Lo importante de estas variables es que sólo pueden tomar dos valores: verdadero o falso, no hay más posibilidades y esta propiedad la utilizaremos para tomar decisiones, por ejemplo: tengo una variable que se llama bTarifa Menor y que es verdadera si tengo una variable edad de tipo entero <= 12 y en caso contrario será falsa. Podríamos decir que si la variable bTarifa Menor es verdadera aplicaremos un descuento del 20% en el precio de un producto.

La definición de estas variables dependerá del lenguaje de programación que estemos utilizando. No podemos encontrar con los tipos: booblean, bool, Boolean, Bool. En cuanto tengamos claro cual de los tipos es el que utiliza el lenguaje con el que estamos programando definiremos la variable constantes.

Constantes

Las constantes son similares a las variables, con la salvedad de que no llevan el signo dólar delante, y sólo la podemos asignar una vez. Para definir una constantes usaremos la función que se define como sigue:

```
<html>
   <body>
      <?PHP
```

```
        define ("CONSTANTE", "Hola Mundo");
        printf (CONSTANTE);
    ?>
  </body>
</html>
```

PHP crea diversas constantes al arrancar, como PHP_VERSION que contiene la versión de PHP, TRUE que le asigna 1 o FALSE que le asigna 0.

Operadores Aritméticos

$a + $b Suma
$a - $b Resta
$a * $b Multiplicación
$a / $b &ss=codigoenlinea>
$a / $b División
$a % $b Resto de la división de $a por $b
$a++ Incrementa en 1 a $a
$a-- Resta 1 a $a

Operadores de Cadenas

El único operador de cadenas que existen es el de concatenación, el punto. Pero no os asustéis, PHP dispone de toda una batería de funciones que os permitirán trabajar cómodamente con las cadenas.

```
$a = "Hola"; $b = $a . "Mundo";   // Ahora $b contiene "Hola Mundo"
```

En este punto hay que hacer una distinción, la interpretación que hace PHP de las simples y dobles comillas. En el segundo caso PHP interpretará el contenido de la cadena.

```
$a = "Mundo";
echo = 'Hola $a';   //Esto escribirá "Hola $a"
echo = "Hola $a";  //Esto escribirá "Hola Mundo&q; //Esto escribirá "Hola Mundo"
```

Operadores de Comparación

$a < $b $a menor que $b
$a > $b $a mayor que $b
$a <= $b $a menor o igual que $b
$a >= $b $a mayor o igual que $b
$a == $b $a igual que $b
$a != $b $a distinto que $b

Operadores Lógicos

$a AND $b Verdadero si ambos son verdadero $a && $b Verdadero si ambos son verdadero $a OR $b Verdadero si alguno de los dos es verdadero $a !! $b Verdadero si alguno de los dos es verdadero $a XOR $b Verdadero si sólo uno de los dos es verdadero !$a Verdadero si $a es falso, y recíprocamente

Operadores de Asignación

$a = $b Asigna a $a el contenido de $b
$a += $b Le suma a $b a $a
$a -= $b Le resta a $b a $a
$a *= $b Multiplica $a por $b y lo asigna a $a
$a /= $b Divide $a por $b y lo asigna a $a
$a .= $b Añade la cadena $b a la cadena $a

CLASIFICACIÓN DE SENTENCIAS DE CONTROL

SENTENCIAS DE CONTROL ITERATIVAS

- while (expresión) {
  sentencia(s) a ejecutar si se cumple la condición
}
- while (expresión):
sentencia(s) a ejecutar si se cumple la condición
endwhile;
- for (inicialización; condición; modificación) {
  sentencia(s) a ejecutar si se cumple la condición
}
- for (inicialización; condición; modificación) :
  sentencia(s) a ejecutar si se cumple la condición
endfor;
- do {
  sentencia(s) a ejecutar si se cumple la condición
}
while (expresión);

SENTENCIAS DE CONTROL SENTENCIAS DE CONTROL SALTO

Sentencia BREAK y CONTINUE

• Permiten alterar la ejecución prevista de una estructura iterativa
• BREAK: - Cancela completamente la ejecución del bucle. - Es utilizada, además, en la sentencia SWITCH.
• CONTINUE: - Cancela una de las iteraciones pasando directamente a la siguiente.

Dentro de un bucle, ambas sentencias admite el uso de un parámetro opcional que indica: Número de estructuras de control de las que hay que salir o Número de niveles a saltar para continuar la ejecución (break n; continue n;).

SENTENCIAS DE CONTROL OTRAS

de expresión

cualquier expresión válida C seguida de punto y coma Ejemplo: func ( ); a=b+c; b+f ( ); ;

de bloque

grupo de sentencias relacionadas que se tratan como una unidad
Los bloques comienzan con "{" y concluyen con "}"
Se pueden colocar en cualquier punto de programa.

ESTRUCTURA DE ALGUNAS SENTENCIAS DE CONTROL

Las sentencias de control permiten ejecutar bloque de códigos dependiendo de unas condiciones. Para PHP el 0 es equivalente a Falso y cualquier otro número es Verdadero.

IF...ELSE

La sentencia IF...ELSE permite ejecutar un bloque de instrucciones si la condición es Verdadera y otro bloque de instrucciones si ésta es Falsa. Es importante tener en cuenta que instrucciones si ésta es Falsa. Es importante tener en cuenta que la

condición que evaluemos ha de estar encerrada entre paréntesis (esto es aplicable a todas las sentencias de control).

```
if (condición) {

 Este bloque se ejecuta si la condición es VERDADERA
 }
else {

      Este boque se ejecuta si la condición es FALSA
}
```

Existe una forma sencilla de usar la sentencia IF cuando no tenemos que usar el ELSE y solo tenemos que ejecutar una línea de código.

```
if ($a > 4) echo "$a es mayor que 4";
```

IF...ELSEIF...ELSE

La sentencia IF...ELSEIF...ELSE permite ejecuta varias condiciones en cascada. Para este caso veremos un ejemplo, en el que utilizaremos los operadores lógicos.

```
<?PHP
   if ($nombre == ""){
        echo "Tú no tienes nombre";
    }
    Else
        if (($nombre=="eva") OR ($nombre=="Eva")) {
```

```
                echo "  echo "Tu nombre es EVA";<
        }
        else {
                echo "Tu nombre es " . $nombre;
        }
    }
‘?>
```

SWITCH...CASE...DEFAULT

Una alternativa a IF...ELSEIF...ELSE, es la sentencia SWITCH, la cual evalúa y compara cada expresión de la sentencia CASE con la expresión que evaluamos, si llegamos al final de la lista de CASE y encuentra una condición Verdadera, ejecuta el código de bloque que haya en DEFAULT. Si encontramos una condición verdadera debemos ejecutar un BREAK para que la sentencia SWITCH no siga buscando en la lista de CASE. Veamos un ejemplo.

```
<?PHP
   switch ($dia) {
        case "Lunes":
            echo "Hoy es Lunes";
            break;
        case "Martes":
            echo "Hoy es Martes";
            break;
        case "Miercoles":
            echo "Hoy es Miercoles";
            break;
       case "Jueves":
             echo "Hoy es Jueves";
              break;
```

```
        case "Viernes":
                echo "Hoy es Viernes";
                 break;
       case "Sábado":
                 echo "Hoy es Sábado";
                   break;
         case "Domingo":
                   echo "Hoy es Domingo";
                    break;
         default:
default:
         echo "Esa cadena no corresponde a ningún día de la semana";
}
?>
```

WHILE

La sentencia WHILE ejecuta un bloque de código mientras se cumpla una determinada condición.

```
<?PHP $num = 1; while ($num < 5) { echo $num; $num++ } ?>
```

Podemos romper un bucle WHILE utilizando la sentencia BREAK.

```
<?PHP
     $num = 1;
      while ($num < 5) {
              echo $num; if ($num == 3) {
                    echo "Aquí nos salimos \n"; break
              }
             $num++
       }
?>
```

DO...WHILE

Esta sentencia es similar a WHILE, salvo que con esta sentencia primero ejecutamos el bloque de código y después se evalúa la condición, por lo que el bloque de código se ejecuta siempre al menos una vez.

```
<?PHP
      $num = 1;
          do {
             echo $num; if ($num == 3) {
                  echo "Aquí nos salimos \n"; break
             }
             $num++
            }
              while ($num < 5);
?>
```

Ciclo while

Programas anteriores pero con el ciclo while

- Número del 1 al 10

```
<?PHP
   $count=1;
   while ($count<=10) {
       echo "$count";
       $count++;
   }
```

```
?>
```

- ```
  <?PHP
     $count=1;
     while ($count<=10) {
         echo $count."<br/>";
         $count++;
      }
  ?>
  ```
- De 5 en 5 asta el 100

```
<?php
   while ($count<=100) {
      echo $count."<br/>";
      $count=$count+5;
   }
?>
```

- factorial

```
<?php
    $a=10;
    $c=1;
      while ($count<=$a) {
        $count++;
        $c=$c*$count;
    }
    echo $c;
?>
```

- Suma del 1 hasta el 10

```
<?php
     $a=10;
    $c=1;
      while ($count<=$a) {
```
```

```
        $count++;
        $c=$c*$count;
      }
     echo $c;
?>
```

•

```
<?php
   $a=10;
   $c=1;
     while ($count<=$a) {
       $count++;
       $c=$c*$count;
     }
     echo $c;
?>
```

•

```
<?php
   $a=10;
   $c=0;
   $b=-1;
    while ($count<=$a) {
     $c=$c+((pow($b,$count))*$count);
     $count++;
    }
    echo $c;
?>
```

•

```
<?php
  $a=10;
  $c=0;
```

```
  $b=-1;
    while ($count<=$a) {
$c=$c+((pow($b,$count))*(1/$count));
    $count++;
    }
   echo $c;
?>
•
<?php
   $n=3;
   $c=1;
    while ($count<=$n) {
      $variable=pow(2,$count);
 $c=$c+((1+$variable)/(3+$variable));
       $count++;
   }
   echo $c;
?>
•
<?php
    $n=3;
    $c=1;
   while ($count<=$n) {
      $potencia=pow (-1,$count);
      $variable=pow (2,$count);
$c=$c+(($potencia)*((1+$variable)/(3+$variable)));
      $count++;
   }
   echo $c;
?>
```

•

```
<?php
  $n=3;
  $arriba=3;
  $abajo=7;
  $c=0;
  $count=1;
    while ($count<=$n) {
      $arriba=$arriba*$count;
      $abajo=$abajo*$count;
      $c=$c+($arriba/$abajo);
      $count++;
    }
  echo $c;
?>
```

•

```
<?php
  $n=3;
  $c=0;
  $count=1;
    while ($count<=$n) {
     $potencia=pow (2*$count,$count);
     $c=$c+($potencia/$count);
     $count++;
    }
  echo $c;
?>
```

FOR

El bucle FOR no es estrictamente necesario, cualquier bucle FOR puede ser sustituido fácilmente por otro WHILE. Sin embargo, el bucle FOR resulta muy útil cuando debemos ejecutar un bloque de código a condición de que una variable se encuentre entre un valor mínimo y otro máximo. El bucle FOR también se puede romper mediante la sentencia BREAK.

```
<?PHP
        for ($num = 1; Snum <=5; $num++) {
                echo $num; if ($num == 3) {
                        echo "Aquí nos salimos \ n"; break
                        }
                }
?>
```

Programas con el ciclo for

Números del 1 al 10

```
<?PHP
  for ($x=1;$x<=10;$x++) {
     echo $x."<br/>";
  };
?>
<?PHP
  for ($x=1;$x<=10;$x++) {
    echo "$x,";
  };
?>
```

- Múltiplos del 5 que esta desde el 5 al 100

```
<?PHP
```

```
  for ($x=5;$x<=500;$x=$x+5) {
    echo "$x,";
  };
?>
```

- Factorial

```
<?PHP
  $x=5;
  $c=1;
  for ($n=1;$n<=$x;$n++) {
 $c=$c*$n;
  };
  echo"$c";
?>
```

- Interés

```
<?PHP
 echo"Interes Mensual con un deposito de 1000$ y un interes anual del 12%";
  $deposito=1000;
  $interes=12;
  $tasa=1/12;
  $tasa2=1/360;
  $g=12/100;
  $total=0;
  $ganancia1=1000;
  $ganancia=1000;
    for ($x=1;$x<=12;$x++) {
      $total=$total+$ganancia1+$tasa*$g*$ganancia;
      $pudin=$total;
      $ganancia=$pudin;
      $ganancia1=0;
}
```

```
    for ($y=1;$y<=360;$y++) {
      $total2=$total2+$deposito+$tasa2*$g*$ganancia;
      $pudin2=$total2;
      $ganancia=$pudin2;
      $deposito=0;
}
?>
<br/>
<?
    echo"La ganancia obtenida mensualmente por un año es $total"
?>
<br/>
<?
    echo"La ganancia obtenida por dia durante un año es $total2"
?>
```

- Suma de la sucesión 1+(3/5)+(5/7)+(9/11)

```
<?PHP
  $suma=0;
  for ($x=1;$x<=3;$x++) {
    $f=(pow (2,$x));
    $suma=$suma+(($f+1)/($f+3));
  }
  $suma1=$suma+1;
  echo"suma de la sucesion      suma=$suma1"
?>
```

- Suma de la sucesión 1-(3/5)+(5/7)-(9/11)

```
<?PHP
   $suma=0;
   for ($x=1;$x<=3;$x++) {
    $f=(pow (2,$x));
```

```
        $suma=$suma+(($f+1)/($f+3))*(pow(-1,$x));
      }
      $suma1=$suma+1;
      echo"suma de la sucesion      suma=$suma1"
?>
```

- Suma de la sucesión (3/7)+(6/14)+(18/42)+(72/168)

```
<?PHP
    $suma=0;
    $z=3;
    $w=7;
    for ($x=1;$x<=6;$x++) {
        $z=$z*$x;
        $w=$w*$x;
     $suma=$suma+($z/$w);
  };
  echo"suma de la sucesion    suma=$suma";
?>
```

- Suma de la serie∑ ((2*n)^n)/n

```
<?PHP
    $n=6;
    $suma=0;
    for ($i=1;$i<=$n;$i++) {
        $f=pow (2*$i,$i);
        $suma=$suma+($f/$i);
     };
     echo"suma de la serie suma=$suma";
 ?>
```

- Suma de la serie∑ (1/n)

```
<?PHP
    $n=3;
```

```
    $s=0;
    for ($i=1;$i<=$n;$i++) {
       $s=$s+(1/$i);
    };
    echo"$s";
?>
```

- Suma de la serie ∑n

```
<?PHP
    $n=3;
    $s=0;
    for ($i=1;$i<=$n;$i++) {
        $s=$s+$i;
     };
     echo"$s";
?>
```

- Suma de la serie ∑((-1^n)*n)

```
<?PHP
   $n=5;
   $s=0;
   for ($i=1;$i<=$n;$i++) {
      $p=(pow (-1,$i))*$i;
      $s=$s+$p;
    };
    echo"$s";
?>
```

- Suma de la serie ∑((-1^n)*(1/n))

```
<?PHP
   $n=5;
   $s=0;
   for ($i=1;$i<=$n;$i++) {
```

```
        $p=(pow (-1,$i))*(1/$i);
        $s=$s+$p;
      };
      echo"$s";
  ?>
```

FOREACH

PHP4 (PHP3 no) incluye una construcción foreach, tal como perl y algunos otros lenguajes. Esto simplemente da un modo fácil de iterar sobre arrays. Hay dos sintaxis; la segunda es una extensión menor, pero útil de la primera:

foreach (expresion_array as $value)
 sentencia foreach (expresion_array as $key => $value) sentencia

La primera forma recorre el array dado por expresion_array. En cada iteración, el valor del elemento actual se asigna a $value y el puntero interno del array se avanza en una unidad (así en el siguiente paso, se estará mirando el elemento siguiente).

La segunda manera hace lo mismo, salvo que la clave del elemento actual será asignada a la variable $key en cada iteración.

Nota: Cuando foreach comienza su primera ejecución, el puntero interno a la lista (array) se reinicia automáticamente al primer elemento del array. Esto significa que no se necesita llamar a reset ( ) antes de un bucle foreach.

Nota: Hay que tener en cuenta que foreach con una copia de la lista (array) especificada y no la lista en si, por ello el puntero de la lista no es modificado como en la construcción each.

Puede haber observado que las siguientes son funcionalidades idénticas:

```
reset ( $arr ); while ( list (, $value ) = each ( $arr ) ) {echo "Valor: $value<br>\n";}
foreach ($arr as $value) {echo "Valor: $value<br>\n";}
```

Las siguientes también son funcionalidades idénticas:

```
reset ($ar);
while (list ( $key, $value) = each ($arr) ) {
    echo "Key: $key; Valor: $value<br>\n";
}
foreach ($arr as $key => $value) {
    echo "Key: $key; Valor: $value<br>\n";
 }
```

Algunos ejemplos más para demostrar su uso:

```
/* foreach ejemplo 1: sólo valor*/ $a = array (1, 2, 3, 17);
foreach ($a as $v) {print "Valor actual de \$a: $v.\n";}
/* foreach ejemplo 2: valor (con clave impresa para ilustrar) */ $a = array (1, 2, 3, 17);
$i = 0; /* sólo para propósitos demostrativos */
foreach ($a as $v) {print "\$a [$i] => $k.\n";}
/* foreach ejemplo 3: clave y valor */ $a = array ("uno" => 1, "dos" => 2, "tres" => 3, "diecisiete" => 17);
foreach ($a as $k => $v) {print "\$a[$k] => $v.\n";}
```

BREAK

break escapa de la estructuras de control iterante (bucle) actuales for, while,oswitch.

break accepta un parámetro opcional, el cual determina cuantas estructuras de control hay que escapar.

```
$arr = array (’one’, ’two’, ’three’, ’four’, ’stop’, ’five’);
  while (list (, $val) = each ($arr)) {
     if ($val == ’stop’) {
       break; /* You could also write ’break 1;’ here. */
     }
    echo "$val<br>\n";
  }
/* Using the optional argument. */
    $i = 0;
    while (++$i) {
        switch ($i) {
          case 5: echo "At 5<br>\n";
          break 1; /* Exit only the switch. */
          case 10: echo "At 10; quitting<br>\n";
          break 2; /* Exit the switch and the while. */
         default: break;
       }
   }
```

CONTINUE

continue se usa dentro de la estructura del bucle para saltar el resto de la iteración actual del bucle y continuar la ejecución al comienzo de la siguiente iteración.

continue accepta un parámetro opcional, el cual determina cuantos niveles (bluces) hay que saltar antes de continuar con la ejecución.

```
while (list($key,$value) = each ($arr)) {
    if ($key % 2) {
       // salta los miembros impares continue;
    } do_something_odd ($value);
} $i = 0;
while ($i++ < 5) {
    echo "Outer<br>\n";
     while (1) {
       echo " Middle<br>\n";
       while (1) {
            echo " Inner<br>\n";
             continue 3;
         }
         echo "This never gets output.<br>\n";
      }
      echo "Neither does this.<br>\n";
 }
```

SWITCH

La sentencia switch es similar a una serie de sentencias IF en la misma expresión. En muchas ocasiones, se quiere comparar la misma variable (o expresión) con muchos valores diferentes, y ejecutar una parte de código distinta dependiendo de a qué valor es igual. Para ello sirve la sentencia switch.

Los siguientes dos ejemplos son dos modos distintos de escribir la misma cosa, uno usa una serie de sentencias if, y el otro usa la sentencia switch:

```
if ($i == 0) {
   print "i es igual a 0";
 }
```

```
if ($i == 1) {
    print "i es igual a 1";
}
 if ($i == 2) {
     print "i es igual a 2";
}
switch ($i) {
   case 0: print "i es igual a 0";
   break;
    case 1: print "i es igual a 1";
   break;
    case 2: print "i es igual a 2";
     break;
 }
```

Es importante entender cómo se ejecuta la sentencia switch para evitar errores. La sentencia switch ejecuta línea por línea (realmente, sentencia a sentencia). Al comienzo, no se ejecuta código. Sólo cuando se encuentra una sentencia case con un valor que coincide con el valor de la expresión switch PHP comienza a ejecutar las sentencias. PHP continúa ejecutando las sentencias hasta el final del bloque switch, o la primera vez que vea una sentencia break. Si no se escribe una sentencia break al final de una lista de sentencias case, PHP seguirá ejecutando las sentencias del siguiente case. Por ejemplo:

```
switch ($i) {
   case 0: print "i es igual a 0";
   case 1: print "i es igual a 1";
   case 2: print "i es igual a 2";
}
```

Aquí, si $i es igual a 0, ¡PHP ejecutaría todas las sentencias print! Si $i es igual a 1, PHP ejecutaría las últimas dos sentencias print y sólo si $i es igual a 2, se obtendría la conducta “esperada” y solamente se mostraría “i es igual a 2”. Así, es importante no olvidar las sentencias break (incluso aunque pueda querer evitar escribirlas intencionadamente en ciertas circunstancias).

En una sentencia switch, la condición se evalúa sólo una vez y el resultado se compara a cada sentencia case. En una sentencia elseif, la condición se evalúa otra vez. Si tu condición es más complicada que una comparación simple y/o está en un bucle estrecho, un switch puede ser más rápido.

La lista de sentencias de un case puede también estar vacía, lo cual simplemente pasa el control a la lista de sentencias del siguiente case.

switch ($i) {case 0: case 1: case 2: print "i es menor que 3, pero no negativo"; break; case 3: print "i es 3";}

Un case especial es el default case. Este case coincide con todo lo que no coincidan los otros case. Por ejemplo:

switch ($i) {case 0: print "i es igual a 0"; break; case 1: print "i es igual a 1"; break; case 2: print "i es igual a 2"; break; default: print "i no es igual a 0, 1 o 2";}

La expresión case puede ser cualquier expresión que se evalúe a un tipo simple, es decir, números enteros o de punto flotante y cadenas de texto. No se pueden usar aquí ni arrays ni objetos a menos que se conviertan a un tipo simple.

La sintaxis alternativa para las estructuras de control está también soportada con switch. Para más información, ver sintaxis alternativa para estructuras de control.

switch ($i):

```
case 0: print "i es igual 0";
break;
case 1: print "i es igual a 1";
break; case 2: print "i es igual a 2";
 break;
default: print "i no es igual a 0, 1 o 2"; endswitch;
require ( )
```

La sentencia require ( ) se sustituye a sí misma con el archivo especificado, tal y como funciona la directiva #include de C.

Un punto importante sobre su funcionamiento es que cuando un archivo se incluye con include ( ) o se requiere con require ( )), el intérprete sale del modo PHP y entra en modo HTML al principio del archivo referenciado, y vuelve de nuevo al modo PHP al final. Por esta razón, cualquier código dentro del archivo referenciado que debiera ser ejecutado como código PHP debe ser encerrado dentro de etiquetas válidas de comienzo y fin de PHP.

Require ( ) no es en realidad una función de PHP; es más una construcción del lenguaje. Está sujeta a algunas reglas distintas de las de funciones. Por ejemplo, require ( ) no esta sujeto a ninguna estructura de control contenedora. Por otro lado, no devuelve ningún valor; intentar leer un valor de retorno de una llamada a un require ( ) resulta en un error del intérprete.

A diferencia de include ( ), require ( ) siempre leerá el archivo referenciado, incluso si la línea en que está no se ejecuta nunca. Si se quiere incluir condicionalmente un archivo, se usa include ( ). La sentencia conditional no afecta a require ( ). No obstante, si la línea en la cual aparece el require ( ) no se ejecuta, tampoco se ejecutará el código del archivo referenciado.

De forma similar, las estructuras de bucle no afectan la conducta de require ( ). Aunque el código contenido en el archivo referenciado está todavía sujeto al bucle, el propio require ( ) sólo ocurre una vez.

Esto significa que no se puede poner una sentencia require ( ) dentro de una estructura de bucle y esperar que incluya el contenido de un archivo distinto en cada iteración. Para hacer esto, usa una sentencia include ( ).

En PHP3, es posible ejecutar una sentencia return dentro de un archivo referenciado con require ( ), en tanto en cuanto esa sentencia aparezca en el ámbito global del archivo requerido (require ( )). No puede aparecer dentro de ningún bloque (lo que siginifica dentro de llaves ({ })). En PHP4, no obstante, esta capacidad ha sido desestimada. Si se necesita esta funcionalidad, véase include().

Ver tambien include ( ), require_once ( ), include_once ( ), readfile ( ), y virtual ( ).

INCLUDE ( )

La sentencia include ( ) incluye y evalúa el archivo especificado.

Si "URL fopen wrappers" esta activada en PHP (como está en la configuración inicial), se puede especificar el fichero que se va a incluir usando una URL en vez de un fichero local (con su Path). Ver Ficheros remotos y fopen ( ) para más información.

Un punto importante sobre su funcionamiento es que cuando un archivo se incluye con include ( ) o se requiere con require ( ), el intérprete sale del modo PHP y entra en modo HTML al principio del archivo referenciado, y vuelve de nuevo al modo PHP al final. Por esta razón, cualquier código dentro del archivo referenciado que debiera ser ejecutado como código PHP debe ser encerrado dentro de etiquetas válidas de comienzo y fin de PHP.

Esto sucede cada vez que se encuentra la sentencia include ( ), así que se puede usar una sentencia include ( ) dentro de una estructura de bucle para incluir un número de archivos diferentes.

$archivos = array (’primero.inc’, ’segundo.inc’, ’tercero.inc’); for ($i = 0; $i < count ($archivos); $i++) {include $archivos [$i];}

Include ( ) difiere de require ( ) en que la sentencia include se re-evalúa cada vez que se encuentra (y sólo cuando está siendo ejecutada), mientras que la sentencia require ( ) se reemplaza por el archivo referenciado cuando se encuentra por primera vez, se vaya a evaluar el contenido del archivo o no (por ejemplo, si está dentro de una sentencia if cuya condición evaluada es falsa).

Debido a que include ( ) es una construcción especial del lenguaje, se debe encerrar dentro de un bloque de sentencias si está dentro de un bloque condicional.

```
/* Esto es ERRÓNEO y no funcionará como se desea. */
if ($condicion) include ($archivo); else include ($otro);
/* Esto es CORRECTO. */
if ($condicion) {include ($archivo);} else {include ($otro);}
```

En ambos, PHP3 y PHP4, es posible ejecutar una sentencia return dentro de un archivo incluido con include ( ), para terminar el procesado de ese archivo y volver al archivo de comandos que lo llamó. Existen algunas diferencias en el modo en que esto funciona, no obstante. La primera es que en PHP3, return no puede aparecer dentro de un bloque a menos que sea un bloque de función, en el cual return se aplica a esa función y no al archivo completo. En PHP4, no obstante, esta restricción no existe. También, PHP4 permite devolver valores desde archivos incluidos con include ( ). Se puede capturar el valor de la llamada a include ( )

como se haría con una función normal. Esto genera un error de intérprete en PHP3.

Ejemplo 11-1. Include ( ) en PHP3 y PHP4

Asumamos la existencia del siguiente archivo (llamado test.inc) en el mismo directorio que el archivo principal:

```
<?php echo "Antes del return <br>\n";
  if (1) {return 27;}
     echo "Después del return <br>\n";
?>
```

Asumamos que el archivo principal (main.html) contiene lo siguiente:

```
<?php
   $retval = include (’test.inc’);
   echo "El archivo devolvió: ’
   $retval’<br>\n"; ?>
```

Cuando se llama a main.html en PHP3, generará un error del intérprete en la línea 2; no se puede capturar el valor de un include ( ) en PHP3. En PHP4, no obstante, el resultado será:

Antes del return el archivo devolvió: ’27’

Ahora, asumamos que se ha modificado main.html para que contenga lo siguiente:

```
<?php
    include (’test.inc’);
    echo "De vuelta en main.html<br>\n";
```

?>

En PHP4, la salida será:

Antes del return de vuelta en main.html

No obstante, PHP3 dará la siguiente salida: Antes del return 27 de vuelta en main.html

Parse error: parse error in /home/torben/public_html/phptest/main.html on line 5

El error del intérprete es resultado del hecho de que la sentencia return está encerrada en un bloque de no-función dentro de test.inc. Cuando el return se mueve fuera del bloque, la salida es:

Antes del return 27 de vuelta en main.html

El ’27’ espúreo se debe al hecho de que PHP3 no soporta devolver valores con return desde archivos como ese.

TABLAS

Las tablas (o array en inglés), son muy importantes en PHP, ya que generalmente, las funciones que devuelven varios valores, como las funciones ligadas a las bases de datos, lo hacen en forma de tabla.

En PHP disponemos de dos tipos de tablas. El primero sería el clásico, utilizando índices:

```
<?PHP
    $ciuda d [ ] = "París";
```

```
    $ciudad [ ] = "París";
    $ciudad [ ] = "Roma";
    $ciudad [ ] = "Sevilla";
    $ciudad [ ] = "Londres";
    print ("yo vivo en " . $ciudad [2] . "<BR>\n");
?>
```

Esta es una forma de asignar elementos a una tabla, pero una forma más formal es utilizando la función array

```
<?PHP
    $ciudad = array ("París", "Roma", "Sevilla", "Londres");
    //contamos el número de elementos de la tabla
    $numelentos = count ($ciudad);
    //imprimimos todos los elementos de la tabla
     for ($i=0; $i < $numelentos; $i++) {
            print ("La ciudad $i es $ciudad[$i] <BR>\n");
     }
?>
```

Si no se especifica, el primer índice es el cero, pero podemos utilizar el operador => para especificar el índice inicial.

```
$ciudad = array (1=>"París", "Roma", "Sevilla", "Londres");
```

Un segundo tipo, son las tablas asociativas, en las cuáles a cada elemento se le asigna un valor (key) para acceder a él.

Para entenderlo, que mejor que un ejemplo, supongamos que tenemos una tabla en la que cada elemento almacena el número de visitas a nuestra web por cada día de la semana.

Utilizando el método clásico de índices, cada día de la semana se representaría por un entero, 0 para lunes, 1 para martes, etc.

```
$visitas [0] = 200;
$visitas [1] = 186;
```

si usamos las tablas asociativas sería

```
$visitas ["lunes"] = 200;
$visitas ["martes"] = 186;
```

o bien,

```
$visitas = array ("luodigo">
$visitas = array ("lunes"=>200; "martes"=>186);
```

Ahora bien, recorrer una tabla y mostrar su contenido es sencillo utilizando los índices, pero ¿cómo hacerlo en las tablas asociativas?. La manipulación de las tablas asociativas se hace a través de funciones que actúan sobre un puntero interno que indica la posición. Por defecto, el puntero se sitúa en el primer elemento añadido en la tabla, hasta que es movido por una función:

current - devuelve el valor del elemento que indica el puntero
pos - realiza la misma función que current
reset - mueve el puntero al primer elemento de la tabla
end - mueve el puntero al último elemento de la tabla
next - mueve el puntero al elemento siguiente
prev - mueve el puntero al elemento anterior
count - devuelve el número de elementos de una tabla.

Veamos un ejemplo de las funciones anteriores:

```
<?PHP
   $semana = array ("lunes", "martes", "miércoles", "jueves", "viernes", "sábado",
"domindo");
   echo count ($semana); //7
    //situamos el puntero en el primer elemento reset ($semana);
    echo current ($semana);
    //lunes next ($semana); echo pos ($semana);
     //martes end ($semana) echo pos ($semana);
     //domingo prev ($semana);
      echo current ($semana);
     //sábado
?>
```

Recorrer una tabla con las funciones anteriores se hace un poco lioso, para ello se recomienda utilizar la función each ( ).

```
<?.
<?PHP
    $visitas = array ("lunes"=>200, "martes"=>186, "miércoles"=>190,
"jueves"=>175);
    reset ($visitas);
    while (list ($clave, $valor) = each ($visitas)) {
            echo "el día $clave ha tenido $valor visitas<BR>";
    }
?>
```

La función each ( ) devuelve el valor del elemento actual, en este caso, el valor del elemento actual y su clave, y desplaza el puntero al siguiente, cuando llega al final devuelve FALSO, y termina el bucle while ( ).

Foreach forma general

```
foreach (nombre_array as nombre_variable) {
        sentencia(s) a ejecutar para cada elemento del array
 }
```

Sentencia incorporada en la versión php 4.0.

(*) Permite recorrer todos los elementos de un array de una forma muy simple.

En la primera iteración la variable contendrá el valor del primer elemento del array nombre_array en las siguientes iteraciones los sucesivos valores hasta recorrer todos los elementos del array.

FUNCIONES

Una función podría ser definida como un conjunto de instrucciones que explotan ciertas variables para realizar una tarea más o menos elemental.

PHP basa su eficacia principalmente en este tipo de elemento. Una gran librería que crece constantemente, a medida que nuevas versiones van surgiendo, es complementada con las funciones de propia cosecha dando como resultado un sinfín de recursos que son aplicados por una simple llamada.

Las funciones integradas en PHP son muy fáciles de utilizar. Tan sólo hemos de realizar la llamada de la forma apropiada y especificar los parámetros y/o variables necesarios para que la función realice su tarea.

Lo que puede parecer ligeramente más complicado, pero que resulta sin lugar a dudas muy práctico, es crear nuestras propias funciones. De una forma general,

podríamos crear nuestras propias funciones para conectarnos a una base de datos o crear los encabezados o etiquetas meta de un documento HTML. Para una aplicación de comercio electrónico podríamos crear por ejemplo funciones de cambio de una moneda a otra o de calculo de los impuestos a añadir al precio de articulo. En definitiva, es interesante crear funciones para la mayoría de acciones más o menos sistemáticas que realizamos en nuestros programas.

Aquí daremos el ejemplo de creación de una función que, llamada al comienzo de nuestro script, nos crea el encabezado de nuestro documento HTML y coloca el titulo que queremos a la página:

```
<?
  function hacer_encabezado ($titulo) {
     $encabezado="<html><head>t<title>$titulo</title></head>";
     echo $encabezado;
  }
?>
```

Esta función podría ser llamada al principio de todas nuestras páginas de la siguiente forma:

```
$titulo="Mi web";
hacer_encabezado ($titulo);
```

De esta forma automatizamos el proceso de creación de nuestro documento. Podríamos por ejemplo incluir en la función otras variables que nos ayudasen a construir las etiquetas meta y de esta forma, con un esfuerzo mínimo, crearíamos los encabezados personalizados para cada una de nuestras páginas. De este mismo modo nos es posible crear cierres de documentos o formatos diversos para nuestros textos como si se tratase de hojas de estilo que tendrían la ventaja de ser reconocidas por todos los navegadores.

Por supuesto, la función ha de ser definida dentro del script ya que no se encuentra integrada en PHP sino que la hemos creado nosotros. Esto en realidad no pone ninguna pega ya que puede ser incluida desde un archivo en el que iremos almacenando las definiciones de las funciones que vayamos creando o recopilando.

Estos archivos en los que se guardan las funciones se llaman librerías. La forma de incluirlos en nuestro script es a partir de la instrucción *require* o *include*:

require ("libreria.php") o include ("libreria.php")

En resumen, la cosa quedaría así:

Tendríamos un archivo libreria.php como sigue

```
<?
   //función de encabezado y colocación del titulo
   function hacer_encabezado ($titulo) {
      $encabezado="<html>n<head>nt<title>$titulo</title>n</head>n";
      echo $encabezado;
    }
?>
```

Por otra parte tendríamos nuestro script principal página.php (por ejemplo):

```
<?
 include ("libreria.php");
 $titulo="Mi Web";
 hacer_encabezado ($titulo);
?>
```

```
<body>
  El cuerpo de la página
</body>
</html>
```

Podemos meter todas las funciones que vayamos encontrando dentro de un mismo archivo pero resulta muchísimo más ventajoso ir clasificándolas en distintos archivos por temática: Funciones de conexión a bases de datos, funciones comerciales, funciones generales, etc. Esto nos ayudara a poder localizarlas antes para corregirlas o modificarlas, nos permite también cargar únicamente el tipo de función que necesitamos para el script sin recargar éste en exceso además de permitirnos utilizar un determinado tipo de librería para varios sitios webs distintos.

También puede resultar muy práctico el utilizar una nomenclatura sistemática a la hora de nombrarlas: Las funciones comerciales podrían ser llamadas com_loquesea, las de bases de datos bd_loquesea, las de tratamiento de archivos file_loquesea. Esto nos permitirá reconocerlas enseguida cuando leamos el script sin tener que recurrir a nuestra oxidada memoria para descubrir su utilidad.

No obstante, antes de lanzarnos a crear nuestra propia función, merece la pena echar un vistazo a la documentación para ver si dicha función ya existe o podemos aprovecharnos de alguna de las existentes para aligerar nuestro trabajo. Así, por ejemplo, existe una función llamada header que crea un encabezado HTML configurable lo cual nos evita tener que crearla nosotros mismos.

Como puede verse, la tarea del programador puede en algunos casos parecerse a la de un coleccionista. Hay que ser paciente y metódico y al final, a base de trabajo propio, intercambio y tiempo podemos llegar poseer nuestro pequeño tesoro.

Nota: Si lo deseas puedes repasar todos los conceptos anteriores sobre las funciones, así como diversas otras cosas interesantes en el Videotutorial sobre las funciones en PHP.

Ejemplo de función

Vamos a ver un ejemplo de creación de funciones en PHP. Se trata de hacer una función que recibe un texto y lo escribe en la página con cada carácter separado por "-". Es decir, si recibe "hola" debe escribir "h-o-l-a" en la página web.

Nota: Para comprender este ejemplo necesitamos conocer el bucle for, que se explica en el capítulo Control del flujo en PHP: Bucles II.

La manera de realizar esta función será recorrer el string, caracter a caracter, para imprimir cada uno de los caracteres, seguido de el signo "-". Recorreremos el string con un bucle for, desde el caracter 0 hasta el número de caracteres total de la cadena.

El número de caracteres de una cadena se obtiene con la función predefinida en PHP strlen ( ), que recibe el string entre paréntesis y devuelve el número de los caracteres que tenga.

```
<html>
<head>
   <title>funcion 1</title>
</head>
<body>
<?
function escribe_separa ($cadena) {
   for ($i=0;$i<strlen ($cadena);$i++) {
```

```
      echo $cadena [$i];
      if ($i<strlen ($cadena)-1)
        echo "-";
   }
}
escribe_separa ("hola");
echo "<p>";
escribe_separa ("Texto más largo, a ver lo que hace");
?>
</body>
</html>
```

La función que hemos creado se llama escribe_separa y recibe como parámetro la cadena que hay que escribir con el separador "-". El bucle for nos sirve para recorrer la cadena, desde el primero hasta el último caracter. Luego, dentro del bucle, se imprime cada caracter separado del signo "-". El if que hay dentro del bucle for comprueba que el actual no sea el último caracter, porque en ese caso no habría que escribir el signo "-" (queremos conseguir "h-o-l-a" y si no estuviera el if obtendríamos "h-o-l-a-").

En el código mostrado se hacen un par de llamadas a la función para ver el resultado obtenido con diferentes cadenas como parámetro. Podemos ver el script en marcha.

Funciones definidas por el usuario

Una función se define con la siguiente sintaxis:

```
function foo ($arg_1, $arg_2, ..., $arg_n) {
  echo "Función de ejemplo.\n"; return $retval;
}
```

Cualquier instrucción válida de PHP puede aparecer en el cuerpo de la función, incluso otras funiones y definiciones de clases.

En PHP3, las funciones deben definirse antes de que se referencien. En PHP4 no existe tal requerimiento.

PHP no soporta la sobrecarga de funciones, y tampoco es posible redefinir u ocultar funciones previamente declaradas.

PHP3 no soporta un número variable de parámetros, aunque sí soporta parámetros por defecto (ver Valores por defecto de de los parámetros para más información). PHP4 soporta ambos (ver Listas de longitud variable de parámetros y las referencias de las funciones func_num_args ( ), func_get_arg ( ), y func_get_args ( ) para más información).

Parámetros de las funciones

La información puede suministrarse a las funciones mediante la lista de parámetros, una lista de variables y/o constantes separadas por comas.

PHP soporta pasar parámetros por valor (el comportamiento por defecto), por referencia, y parámetros por defecto. Listas de longitud variable de parámetros sólo están soportadas en PHP4 y posteriores (ver Listas de longitud variable de parámetros y la referencia de las funciones func_num_args ( ), func_get_arg ( ), y func_get_args ( ) para más información). Un efecto similar puede conseguirse en PHP3 pasando un array de parámetros a la función:

```
function takes_array ($input) {
  echo "$input [0] + $input [1] = ", $input[0]+$input [1];
 }
```

Pasar parámetros por referencia

Por defecto, los parámetros de una función se pasan por valor (de manera que si cambias el valor del argumento dentro de la función, no se ve modificado fuera de ella). Si deseas permitir a una función modificar sus parámetros, debes pasarlos por referencia.

Si quieres que un parámetro de una función siempre se pase por referencia, puedes anteponer un ampersand (&) al nombre del parámetro en la definición de la función:

```
function add_some_extra (&$string) {
   $string .= ' y algo más.';
}
$str = 'Esto es una cadena, ';
add_some_extra ($str);
echo $str; // Saca 'Esto es una cadena, y algo más.'
```

Si deseas pasar una variable por referencia a una función que no toma el parámetro por referencia por defecto, puedes anteponer un ampersand al nombre del parámetro en la llamada a la función:

```
function foo ($bar) {
$bar .= ' y algo más.';
}
$str = 'Esto es una cadena, ';
foo ($str); echo $str; // Saca 'Esto es una cadena, ' foo (&$str); echo $str; //
```

Saca ’Esto es una cadena, y algo más.’

Parámetros por defecto

Una función puede definir valores por defecto para los parámetros escalares estilo C++:

function makecoffee ($type = "cappucino") { return "Hacer una taza de $type.\n"; } echo makecoffee ( ); echo makecoffee ("espresso");

La salida del fragmento anterior es:

Hacer una taza de cappucino. Hacer una taza de espresso.

El valor por defecto tiene que ser una expresión constante, y no una variable o miembro de una clase.

En PHP 4.0 también es posible especificar unset como parámetro por defecto. Esto significa que el argumento no tomará ningún valor en absoluto si el valor no es suministrado.

Destacar que cuando se usan parámetros por defecto, estos tienen que estar a la derecha de cualquier parámetro sin valor por defecto; de otra manera las cosas no funcionarán de la forma esperada. Considera el siguiente fragmento de código:

function makeyogurt ($type = "acidophilus", $flavour) { return "Haciendo un bol de $type $flavour.\n"; }
echo makeyogurt ("mora"); // No funcionará de la manera esperada

La salida del ejemplo anterior es:

Warning: Missing argument 2 in call to makeyogurt() in /usr/local/etc/httpd/htdocs/php3test/functest.html on line 41 Haciendo un bol de mora.

Y ahora, compáralo con:

function makeyogurt ($flavour, $type = "acidophilus") { return "Haciendo un bol de $type $flavour.\n"; }
echo makeyogurt ("mora"); // funciona como se esperaba

La salida de este ejemplo es:

Haciendo un bol de acidophilus mora.

Lista de longitud variable de parámetros

PHP4 soporta las listas de longitud variable de parámetros en las funciones definidas por el usuario. Es realmente fácil, usando las funciones func_num_args (), func_get_arg ( ), y func_get_args ( ).

No necesita de ninguna sintaxis especial, y las listas de parámetros pueden ser escritas en la llamada a la función y se comportarán de la manera esperada.

Devolver valores

Los valores se retornan usando la instrucción opcional return. Puede devolverse cualquier tipo de valor, incluyendo listas y objetos.

function square ($num) { return $num * $num; } echo square (4); // saca ’16’.

No puedes devolver múltiples valores desde una función, pero un efecto similar se puede conseguir devolviendo una lista.

```
Function small_numbers ( ) {
return array (0, 1, 2);
}
list ($zero, $one, $two) = small_numbers ( );
old_function
```

La instrucción old_function permite declarar una función usando una sintaxis idéntica a la de PHP/FI2 (excepto que debes reemplazar ’function’ por ’old_function’).

Es una característica obsoleta, y debería ser usada únicamente por el conversor PHP/FI2->PHP3.

Aviso

Las funciones declaradas como old_function no pueden llamarse desde el código interno de PHP. Entre otras cosas, esto significa que no puedes usarlas en funciones como usort ( ), array_walk ( ), y register_shutdown_function ( ). Puedes solventar esta limitación escribiendo un "wrapper" (en PHP3 normal) que a su vez llame a la función declarada como old_function.

Funciones variable

PHP soporta el concepto de funciones variable, esto significa que si una variable tiene unos paréntesis añadidos al final, PHP buscará una función con el mismo nombre que la evaluación de la variable e intentará ejecutarla. Entre otras cosas,

esto te permite implementar retrollamadas (callbacks), tablas de funciones y demás.

Ejemplo 12-1. Ejemplo de función variable

```
<?php
   function foo ( ) {
      echo "Dentro de foo ( )<br>\n";
    }
   function bar ($arg = ”) {
      echo "Dentro de bar ( );
      el parámetro fue ’$arg’.<br>\n";
    }
    $func = ’foo’;
    $func ( );
    $func = ’bar’;
     $func (’test’);
?>
```

Todas las Funciones de librería en PHP (450 Funciones)

FUNCIONES DE FECHA Y HORA

Gettimeofday --> Permite obtener la hora actual.
gmdate --> Da formato a una fecha/hora GMT/CUT.
gmmktime -> Obtiene el valor timestamp UNIX de una fecha GMT.
gmstrftime -> Con esta funciÃ³n da formato a una fecha/hora GMT/CUT segÃºn las convenciones locales.
microtime -> Devuelve el valor timestamp UNIX actual con microsegundos.
mktime -> Obtiene el timestamp UNIX de una fecha.

strftime -> Da formato a la hora o fehca local de acuerdo a a las convenciones locales.
time -> Devuelve la hora de la fecha actual en formato timestamp UNIX.
strtotime -> Permite procesar cualquier descripciÃ³n textual de fehca/hora en inglÃ©s, convirtiÃ©ndola a timestamp de UNIX.
getdate -> Podemos obtener informaciÃ³n de fecha y hora.
checkdate -> Esta funciÃ³n verifica que la fecha sea vÃ¡lida, y su sintaxis es la siguiente.

FUNCIONES PARA EL MANEJO DE ERRORES Y LOGS

error_log -> EnvÃa un mensaje de error a algÃºn lugar (telÃ©fono celular, pager, etc.).
error_reporting -> Establece que errores PHP son registrador.
restore_error_handler -> Regresa al error handler previo.
set_error_handler -> Establece un error de usuario.
trigger_error -> Genera una advertencia de error.
user_error -> Genera una advertencia de error.

FUNCIONES CON DIRECTORIOS

chroot -> Cambia el directorio raÃz.
dir -> Clase directorio.
closedir -> Cierra el puntero a un directorio abierto.
getcwd -> Obtiene el directorio de trabajo actual.
opendir -> Abre un puntero a un directorio.
readdir -> Lee los archivos de un directorio.
rewinddir -> Rebobina el puntero del directorio llevÃ¡ndolo a la posiciÃ³n del primer archivo del mismo.
scandir -> Lista los archivos y directorios ubicados en la ruta especificada.

FUNCIONES PARA EL MANEJO DE ARCHIVOS

basename -> Devuelve la ruta o path correspondiente al nombre del archivo.
chgrp -> Cambia el grupo de un archivo.
chmod -> Cambia permisos de un archivo.
chown -> Cambia el propietario de un archivo.
clearstatcache -> Limpia la cachÃ© de estado de un archivo.
copy -> Copia un archivo.
dirname -> Devuelve la parte de la ruta o path de un archivo correspondiente al directorio.
disk_free_space -> Indica el tamaÃ±o de espacio libre en un directorio.
disk_total_space -> Indicai el tamaÃ±o total de un directorio.
fclose -> Cierra el puntero a un archivo abierto.
feof -> Verifica si el puntero de un archivo ha llegado al final del mismo.
fflush -> VacÃa la salida hacia un archivo.
fgetc -> Obtiene un cÃ¡racter del archivo apuntado.
fgetcsv -> Obitene una lÃnea del archivo apuntado y extrae los campos CSV.
fgets -> Obtiene una linea del archivo apuntado.
fgetss -> Obitene una linea del archivo apuntado y quita las etiquetas html.
file_exists -> Verifica si un archivo existe.
file_get_contents -> Lee un archivo entero en una cadena.
file_put_contents -> Escribe una cadena sobre un archivo.
file -> Lee un archivo completo y lo coloca en un array.
fileatime -> Obtiene la fecha del Ãºltimo acceso a un archivo.
filectime -> Obtiene la fecha de cambio de inode del archivo.
filegroup -> Obtiene el grupo al cual pertenece el archivo.
fileinode -> Obtiene el inode de un archivo.
filemtime -> Obtiene la fecha de Ãºltima modificaciÃ³n de un archivo.
fileowner -> Obtiene el propietario de un archivo.
fileperms -> Obtiene los permisos de un archivo.

filesize -> Obtiene el tamaÃ±o de un archivo.

filetype -> Obtiene el tipo de archivo de un archivo.

flock -> Bloqueo de archivo portable y asesorado.

fnmatch -> Compara un nombre de archivo contra un patrÃ³n.

fopen -> Abre un archivo o una url.

fpasstrhu -> Saca todos los datos restantes del archivo apuntado.

fputs -> Escribe en el archivo apuntado.

fread -> Lee archivos en plano binario.

fscanf -> Procesa la entrada desde un archivo de acuerdo con un formato.

fseek -> SitÃºa el puntero en una posiciÃ³n del archivo.

fstat -> Obtiene informaciÃ³n sobre un archivo usando un apuntador de archivo abierto.

ftell -> Pregunta sobre la posiciÃ³n del apuntador de lectura/escritura de un archivo.

ftruncate -> Trunca un archivo a la longitud dada.

fwrite -> Escribe archivo en plano binario.

glob -> Encuentra nombres de ruta coincidentes con un patrÃ³n.

is_dir -> Informa si el archivo dado es un directorio.

is_executable -> Informa si el archivo nombrado es ejecutable.

is_file -> Informa si el archivo nombrado es un archivo regular.

is_readable -> Informa si el archivo nombrado se puede leer.

is_uploaded_file -> Informa si el archivo fue cargado a travÃ©s de HTTP_POST.

is_writable -> Indica si el nombre de archivo se puede escribir.

is_writeable -> Informa si se puede escribir en el archivo indicado.

link -> Crea un enlace.

inkinfo -> Consigue informaciÃ³n sobre un enlace.

lstat -> Da informaciÃ³n sobre un archivo o enlace simbÃ³lico.

mkdir -> Crea un directorio.

move_uploaded_file -> Mueve un archivo cargado a una nueva ubicaciÃ³n especÃfica.

parse_ini_file -> Procesa un archivo de configuraciÃ³n.

pathinfo -> Indica informaciÃ³n sobre la ruta o path de un archivo.
pclose -> Cierra el archivo de proceso apuntado.
popen -> Abre el archivo de proceso apuntado.
readfile -> Muestra el contenido de un archivo.
readlink -> Devuelve el objeto de un enlace simbÃ³lico.
realpath -> Devuelve el nombre de ruta absoluto simplificado.
rename -> Renombra un archivo.
rewind -> Rebobina la posiciÃ³n del apuntador al archivo a la primera posiciÃ³n del mismo.
rmdir -> Borra un directorio.
set_file_buffer -> Fija el buffer de archivo del archivo apuntado.
stat -> Da informaciÃ³n sobre un archivo.
symlink -> Crea un enlace simbÃ³lico.
tempnam -> Crea un archivo de nombre Ãºnico.
tmpfile -> Crea un archivo temporal.
touch -> Cambia la fecha de modificaciÃ³n de un archivo.
umask -> Cambia la umask actual.
unlink -> Borra un archivo.

FUNCIONES HTTP

header -> EnvÃa una cabecera http.
headers_list -> Nos devuelve una lista de cabeceras.
headers_sent -> Verifica si ya se han enviado cabeceras, y donde.
setcookie -> EnvÃa una cookie.

FUNCIONES DE IMÁGENES

(Para varias de estas funciones debes tener instaladas las librerÃas GD).
GetImageSize -> Muesta el tamaÃ±o de una imagen Gif, JPG o PNG.
ImageArc -> Dibuja una elipse parcial.

ImageChar -> Dibuja un caracter de forma horizontal.

ImageChatUp -> Dibuja un caracter de forma vertical.

ImageColorAllocate -> Define un color para una imagen.

ImageColorAt -> Obtiene el Ãndice de color de un pÃxel.

ImageColorClosest -> Obtiene el Ãndice del color mÃ¡s cercano al color especificado.

ImageColorExact -> Devuelve el Ãndice del color especificado.

ImageColorResolve -> Devuelve el Ãndice del color especificado o su alternativa mÃ¡s cercana.

ImageColorSet -> Establece el color para el Ãndice de la paleta especificada.

ImageColorsForIndex -> Obtiene los colores de un Ãndice.

ImageColorsTotal -> Encuentra el nÃºmero de colores de una imagen.

ImageColorTransparent -> Define un color como transparente.

ImageCopyResized -> Copia y redimensiona una parte de una imagen.

ImageCreate -> Crea una nueva imagen.

ImagreCreateFromGif -> Crea una nueva imagen a partir de un archivo a una URL.

ImageDashedLine -> Dibuja una linea de forma discontinuo.

ImageDestroy -> Destruye una imagen para liberar memoria.

ImageFill -> Rellea una imagen con el color especificado.

ImageFilledPolygon -> Dibuja un polÃgono con relleno.

ImageFilledRectangle -> Dibuja un rectÃ¡ngulo con relleno.

ImageFillToBorder -> Relleno de un color especÃfico.

ImageFontHeight -> Devuelve la altura de una fuente.

ImageFontWidth -> Devuelve el ancho de una fuente.

ImagenGif -> Envia una imagen al navegador web o a un archivo segÃºn los parÃ¡metros que reciba.

ImageInterface -> Activa o desactiva el entrelazado.

ImageLine -> Dibuja una línea.

ImageLoadFont -> Carga una fuente nueva.

ImagePolygon -> Dibuja un polÃgono.

ImagePSBBox -> Devuelve el borde que rodea un rectÃ¡ngulo de texto, usando fuentes PostScript Type1.

ImagePSEncodeFont -> Cambia el vector de codificaciÃ³n de caracteres de una fuente.

ImagePSFreeFont -> Libera la memoria usada por un fuente PostScript Type1.

ImagePSLoadFront -> Carga una fuente PostScript Type1 desde un archivo.

ImagePSText -> Dibuja una cadena de texto sobre una imagen usando una fuente PostScript Type1.

ImageRectangle -> Dibuja un rectÃ¡ngulo.

ImageSetPixel -> Dibuja un pixel.

ImageString -> Dibuja una cadena de texto horizontalmente.

ImageStringUp -> Dibuja una cadena de texto verticalmente.

ImageSX -> Obtiene el ancho de una imagen en pÃxeles.

ImageSY -> Obtiene el alto de una imagen en pixeles.

ImageTTFBox -> Devuelve un cuadro que rodea al texto usando fuentes Truetype.

ImageTTFText -> Escribe texto en la imagen usando fuentes TrueType.

FUNCIONES MATEMÁTICAS

abs -> Valor absoluto.

acos -> Arco coseno.

acosh -> Coseno hiperbÃ³lico inverso.

asin -> Arco seno.

asinh -> Seno hiÃ©rbÃ³lico inverso.

atan2 -> Arco tangente de dos variables.

atanh -> Atangente hiperbÃ³lica inversa.

base_convet -> Convierte un nÃºmero entre bases arbitrarias.

BinDec -> Binario decimal.

ceil -> Redondea fracciones hacia arriba.

cos -> Coseno.

cosh -> Coseno hiperbÃ³lico.

DecBin -> Decimal a binario.

DecHex -> Decimal a hexadecimal.

DecOct -> Decimal a octal.

deg2rad -> Convierte el nÂ´muero en grados a su equivalente en radianes.

exp -> e elevado a...

floor -> Redondea fracciones hacia abajo.

fmod -> Devuelve el residuo de punto flotante (mÃ³dulo) de la divisÃ³n de los argumentos.

getrandmax -> Muestra el mayor valor aleatorio posible.

HexDec -> Hexadecimal a decimal.

hypot -> Devuelve sqrt(num1*num1 + num2*num2).

is_finite -> Encuentra si un valor es un nÃºmero finito legal.

is_infinite -> Encuentra si un valor es infinito.

is_nan -> Encuentra si un valor es un nÃºmero.

lcg_value -> Generador lineal congruente combinado.

log10 -> Logaritmo en base-10.

log1p -> Devuelve log (1 + numero), computado en una forma que es precisa, incluso, cuando el valor es cercano a cero.

log -> Logaritmo natural.

max -> Encuentra el valor mayor.

min -> Encuentra el valor menor.

mt_gentrandmax -> Muestra el mayor valor aleatorio posible.

mt_rand -> Genera un valor aleatorio mejorado.

mt_srand -> Introduce la semilla del generador de nÃºmeros aleatorios mejorado.

OctDec -> Octal a decimal.

pi -> Devuelve el valor de pi.

pow -> ExpresiÃ³n exponencial.

rad2deg -> Converte el nÃºmero en radianes a su equivalente en grados.

rand -> Genera un valor aleatorio.

round -> Redondea un float.

sin -> Seno.
sinh -> Seno hiperbÃ³lico.
sqrt -> RaÃz cuadrada.
srand -> Introduce la semilla del generador de nÃºmeros aleatorios.
tan -> Tangente.
tanh -> Tangente hiperbÃ³lica.

FUNCIONES PARA LAS BASES DE DATOS MYSQL

mysql_affected_rows -> Devuelve el nÃºmero de filas afectadas de la Ãºltima operaciÃ³n MySQL.
mysql_change_user -> Cambia el usuario conectado en la conecciÃ³n activa.
mysql_client_encoding -> Devuelve el nombre del juego de carcateres.
mysql_close -> Cierra una conecciÃ³n con MySQL.
mysql_conecct -> Abre una conecciÃ³n con un servidor MySQL.
mysql_create_db -> Crea una base de datos MySQL.
mysql_data_seek -> Mueve el puntero interno.
mysql_db_name -> Obtiene el nombre de una base de datos.
mysql_db_query -> EnvÃa una sentencia SQL al servidor MySQL.
mysql_drop_db -> Borra una base de datos del servidor MySQL.
mysql_erno -> Informa el nÃºmero de mensaje de error de la Ãºltima operaciÃ³n MySQL.
mysql_error -> Devuelve el texto del mensaje de error de la Ãºltima operaciÃ³n MySQL.
mysql_escape_string -> Escapa una cadena para su uso en mysql_query.
mysql_fetch_array -> Extrae la fila de resultado como una matriz asociativa.
mysql_fetch_assoc -> Recupera una fila de resultado como una matriz asociativa.
mysql_fetch_field -> Extrae la informaciÃ³n de una columna y la devuelve como un objeto.
mysql_fetch_lengths -> Devuelve la longitud de cada salida en un resultado.
mysql_fetch_object -> Extrae una fila de resultado como un objeto.

mysql_fetch_row -> Devuelve una fila de resultado como matriz.
mysql_field_flags -> Devuelve los flags asociados con el campo especificado en un resultado.
mysql_field_len -> Devuelve la longitud de un campo especificado.
mysql_field_name -> Devuelve el nombre del campo especificado como un resultado.
mysql_field_seek -> Asigna el puntero del resultado al offset del campo especificado.
mysql_field_table -> Devuelve el nombre de la tabla donde estÃ¡ el campo especificado.
mysql_field_type -> Devuelve el tipo del campo especificado en un resultado.
mysql_free_result -> Libera la memoria del resultado.
mysql_get_client_info -> Obtiene informaciÃ³n del Cliente MySQL.
mysql_get_host_info -> Obtiene informaciÃ³n de la mÃ¡quina donde reside el servidor MySQL.
mysql_get_proto_info -> Obtiene informaciÃ³n del protocolo MySQL.
mysql_get_server_info -> Obtiene informaciÃ³n del servidor MySQL.
mysql_info -> Obtiene informaciÃ³n sobre la consulta mÃ¡s reciente.
mysql_insert_id -> Devuelve el indentificador generado en la Ãºltima llamada.

INSERT

mysql_list_dbs -> Lista todas las bases de datos disponibles en el servidor MySQL.
mysql_list_fields -> Lista todos los campos del resultado de MySQL.
mysql_list_processes -> Lista todos los procesor MySQL.
mysql_list_tables -> Lista las tablas de una base de datos seleccionada previamente.
mysql_num_fields -> Devuelve el nÃºmero de campos de un resultado.
mysql_num_rows -> Devuelve el nÃºmero de filas obtenidas de un resultado.
mysql_pconnect -> Abre una conecciÃ³n de forma persistente al servidor MySQL.

mysql_ping -> EfectÃºa un chequeo de respuesta sobre una conexiÃ³n de servidor.
mysql_query -> EnvÃa una sentencia SQL a MySQL.
mysql_real_escape_string -> Escapa los caracteres especiales de una cadena para su uso en una sentencia MySQL.
mysql_result -> Devuelve datos sobre un resultado.
mysql_select_db -> Selecciona una base de datos MySQL.
mysql_stat -> Obtiene el estado actual del sistema.
mysql_tablename -> Devuelve el nombre de la tabla de un campo.
mysql_thread_id -> Devuelve el ID del hilo actual.
mysql_unbuffered_query -> EnvÃa una consulta SQL al MySQL, sin recuperar ni colocar en bÃºfer las filas de resultado.

FUNCIONES DE RED

checkdnsrr -> Comprueba los registros DNS correspondientes a nombres de mÃ¡quinas en Internet o direcciones IP.
dns_get_record -> Recupera los registros de recursos DNS asociados con un nombre de dominio.
fscokopen -> Abre una conecciÃ³n de dominio Internet o UNIX via sockets.
gethostbyaddr -> Muestra el nombre de un servidor mediante su direcciÃ³n IP.
gethostbyname -> Obtiene la direccion IP correspondiente al nombre de un servidor.
gethostbynamel -> Obtiene una lista de direcciones IP correspondiente a los servidores.
getmxrr -> Obtiene los registros MX correspondientes a un dominio.
ip2long -> Convierte una cadena que contiene una direcciÃ³n con puntos del Protocol en una direcciÃ³n apropiada.
long2ip -> Convierte una direcciÃ³n de red Interenet a una cadena de formato estandar en Internet con puntos.
syslog -> Genera un mensaje del sistema.

FUNCIONES DE INFORMACIÓN SOBRE PHP

assert -> Revisa si la aserciÃ³n es evaluada a FALSE.
dl -> Carga una extensiÃ³n de PHP en tiempo de ejecuciÃ³n.
extension_lodaded -> Indica si una extensiÃ³n ha sido cargada.
get_cfg_var -> Obtiene el valor de una opciÃ³n de configuraciÃ³n de PHP.
get_current_user -> Obtiene el nombre del propietario del programa actual.
get_defined_constants -> Devuelve un array con los nombres de funciones de un mÃ³dulo.
get_include_path -> Indica la opciÃ³n de configuraciÃ³n inlcude_path actual.
get_included_files -> Devuelve un array con los nombres de los archivos incluidos o requeridos en un programa.
get_loaded_extensions -> Devuelve un array con los nombres de todos los mÃ³dulos compilados y cargados.
get_magic_quotes_gpc -> Obtiene el valor de la configuraciÃ³n activa actual de las comillas mÃ¡gicas gpc.
get_magic_quotes_runtime -> Obtiene el valor de la configuraciÃ³n activa actual de magic_quote_runtime.
get_required_files -> Alias de get_included_files.
getenv -> Muestra el valor de una variable de entorno.
getlastmod -> Obtiene la fecha y hora de la Ãºltima modificaciÃ³n de una pÃ¡gina.
getmyid -> Obtener el GID del propietario de un programa.
getmyinode -> Recupera el inodo del script actual.
getmypid -> Obtiene el ID del proceso PHP.
getmyuid -> Indica el UID del propietario del script PHP.
getopt -> Obtiene opciones de la lista de argumentos desde la lÃnea de comandos.
getrusage -> Muestra el consumo actual de recursos.
ini_alter -> alias de ini_set ( ).
ini_get_all -> Muestra todas las opciones de configuraciÃ³n.

ini_get -> Recupera el valor de una opciÃ³n de configuraciÃ³n.
ini_restore -> Restablece el valor de una opciÃ³n de configuraciÃ³n.
ini_set -> Establece el valor de una opciÃ³n de configuraciÃ³n.
main -> PÃ¡gina predeterminada para main ( ).
memory_get_usage -> Muestra la cantidad de memoria para PHP.
php_ini_scanned_files -> Devuelve la lista de los archivos .ini procesador del directorio .ini adicional.
php_logo_guif -> Obtiene el guid logo.
php_sapi_name -> Devuelve el tipo de interfaz entre el servidor web y PHP.
php_uname -> Muestra informaciÃ³n indicando el sistema operativo donde fue compilado PHP.
phpcredits -> Imprime los crÃ©ditos de los creadores y colaboradores php.
phpinfo -> Muestra una importante cantidad de informaciÃ³n de PHP.
phpversion -> Indica la versiÃ³n actual de PHP.
putenv -> Permite establece el valor de una variable de entorno.
restore_include_path -> Restablece el valor de la opciÃ³n de configuraciÃ³n include_path.
set_include_path -> Establece la configuraciÃ³n de include_path.
set_magic_quotes_runtime -> Establece el valor de la configuraciÃ³n activa actual de magic_quotes_runtimes.
set_time_limit -> Limita el tiempo mÃ¡ximo de ejecuciÃ³n de un programa. Por defecto son 30 segundos.
version_compare -> Compara dos cadenas de nÃºmero de versiÃ³n 'PHP-EstÃ¡ndar'.
zend_logo_guid -> Obtiene el guid zend.
zend_version -> Obtiene la versiÃ³n del motor zend actual.

FUNCIONES DE EJECUCIÓN DE PROGRAMAS

escapeshellarg -> Escapa una cadena a ser usada como argumento del intÃ©rprete de comandos.

escapeshellcdm -> Enmascara los metacaracteres del intÃ©rprete de ordenes.
exec -> Permite ejecutar un programa externo.
passthru -> Ejecuta un programa externo y muestra su salida literal.
proc_close -> Cierra un proceso abierto por proc_open ( ) y devuelve el cÃ³digo de salida del proceso.
proc_get_status -> Obtiene informaciÃ³n sobre un proceso abierto por proc_open ( ).
proc_open -> Ejecuta un comando y abrir apuntadores de archivo para entrada/salida.
proc_terminate -> Mata un proceso abierto por proc_open ( ).
shell_exec -> Ejecuta un comando mediante el intÃ©rprete de comandos y devolver la salida completa como una cadena.
system -> Ejecuta un programa externo y muestra su salida.

FUNCIONES PARA EL MANEJO DE SESIONES

session_cache_expire -> Informa la caducidad actual del cachÃ©.
session_cache_limiter -> Lee y/o cambia el limitador del cachÃ© actual.
session_decode -> Decodifica los datos de una sesiÃ³n a partir de una cadena codificada previamente.
session_destroy -> Destruye todos los datos de una sesiÃ³n, pero no a la sesiÃ³n en sÃ.
session_encode -> Codifica los datos de la sesiÃ³n en una cadena que luegro podrÃ¡ ser decodificada.
session_get_cookie_params -> Obtiene los parÃ¡metros de la cookie de la sesiÃ³n actual.
session_id -> Lee y/o cambia el id de la sesiÃ³n actual.
session_is_registered -> Comprueba si una variable estÃ¡ registrada en la sesiÃ³n actual.
session_module_name -> Lee y/o cambia el mÃ³dulo de la sesiÃ³n actual.
session_name -> Lee y/o cambia el nombre de la sesiÃ³n actual.

session_regenerate_id -> Regenera el ID de la sesiÃ³n actual.
session_register -> Permite registrar mÃ¡s de una variable global en la sesiÃ³n actual.
session_save_path -> Lee y/o cambia la ruta donde se guardan los datos de la sesiÃ³n actual.
session_set_cookie_params -> Cambia los parÃ¡metros de la cookie de la sesiÃ³n.
session_set_save_handler -> Establece unas funciones para el almacenamiento de los datos de la sesiÃ³n a nivel de usuario.
session_start -> Inicia una sesiÃ³n.
session_unregister -> Desregistra una variable de la sesiÃ³n actual.
session_unset -> Elimina todas las variables de la sesiÃ³n.
session_write_close -> Escrie los datos de la sesiÃ³n y la finaliza.

FUNCIONES DE CADENAS

AddCSlashes -> Marca una cadena con barras al estilo del C de la forma \.
AddSlashes -> Marca una cadena con barras.
bin2hex -> Convierte datos binarios en su representaciÃ³n en hexadecimal.
chop -> Elimina los espacios en blanco al final de una cadena.
chr -> Devuelve un carÃ¡cter especÃfico sobre un nÃºmero dado.
chunk_split -> Divide una cadena en trozos mÃ¡s pequeÃ±os.
convert_cyr_string -> Convierte de un juego de caracteres CirÃlico a otro.
count_chars -> Muestra informaciÃ³n sobre los caracteres usados en una cadena.
crc32 -> Calcula el polinomio crc32 de una cadena.
crypt -> Encripta una cadena mediante DES.
echo -> Imprime una o mÃ¡s cadenas.
explode -> Divide una cadena por otra.
fprintf -> Escribe una cadena con formato por una secuencia.
get_html_translation_table -> Devuelve la tabla de traducciÃ³n utilizada por htmlspecialchars ( ) y htmlentities ( ).

hebrev -> Convierte hebreo lÃ³gico a texto visual.
hrebevc -> Convierte ebrero lÃ³gico a texto visual con conversiÃ³n de saltos de lÃ-nea.
html_entity_decode -> Convierte todas las entidades HTML a sus respectivos caraceteres.
htmlentitites -> Convierte los caracteres aplicables a entidades HTML.
implode -> Une elementos de un array mediante una cadena.
join -> Une elementos de una tabla en una cadena.
levenshtein -> Calcula la distancia levenshtein entre dos cadenas.
localconv -> Muestra informaciÃ³n sobre el formato numÃ©rico.
ltrim -> Elimina espacios en blanco del principio de una cadena.
md5_file -> Calcula el resumen criptogrÃ¡fico md5 de un nombre de archivo dado.
md5 -> Calcula el hash md5 de una cadena.
metaphone -> Calcula la 'metafona' de una cadena.
money_format -> Permite dar formato a un nÃºmero como una cadena de moneda.
nl_langinfo -> Consulta informaciÃ³n sobre el lenguaje y la localidad.
nl2br -> Convierte nuevas lÃneas a saltos de lÃneas HTML.
number_format -> Formatea un nÃºmero con los miles agrupados y separadores decimales.
ord -> Devuelve al valor ASCII de un cÃ¡racter. FunciÃ³n inversa de char.
parse_str -> Divide la cadena en variables.
print -> Imprime en pantalla una cadena.
printf -> Imprime una cadena con formato.
quoted_printable_decode -> Convierte una cadena con marcaciÃ³n imprimible a una cadena de 8 bits.
rtrim -> Borra los espacios en blanco al final de la cadena.
setlocale -> Fija la informaciÃ³n de la localidad.
sha1_file -> Calcula el resumen criptogrÃ¡fio sha1 de un archivo.
sha1 -> Calcula el resumen criptogrÃ¡fio sha1 de una cadena.
similar_text -> Calcula la similitud entre dos cadenas.

soundex -> Calcula la clave soundex en una cadena.

sprintf -> Devuelve una cadena con formato, pero no la imprime.

sscanf -> Trocea la entrada de una cadena segÃºn un formato dado.

str_ireplace -> VersiÃ³n 'case insensitive' de str_replace ( ).

str_pad -> Rellena una cadena con otra hasta una longitud definida.

str_repeat -> Repite una cadena.

str_replace -> Sustituye todas las de una cadena en otra.

str_rot13 -> Realiza la transofrmaciÃ³n rt13 sobre una cadena.

str_split -> Permite mezclar aleatoreamente una cadena.

str_word_count -> Muestra informaciÃ³n sobre las palabras usadas en una cadena.

strcasecmp -> Comapra cadenas insensibles en mayÃºsculas y minÃºsculas y seguras en modo binario.

strchr -> Encuentra la primera aparici Ã³n de un caracter en una cadena.

strcmp -> Compara cadenas con seguridad binaria.

strcoll -> Compara cadenas sobre la base de la localidad.

strcspn -> Encuentra la longitud del elemento inicial que no coincide con la mÃ¡scara.

strip_tags -> Elimina las etiquetas HTML y PHP de una cadena.

stripcslashes -> Desmarca la cadena marca don AddCSlashes ( ).

stripos -> Encuentra la posici Ã³n de la primera ocurrencia de una cadena, insensible a mayÃºsculas y minÃºsculas.

stripslashes -> Desmarca la cadena marcada con AddSlashes ( ).

stristr -> Igual que strrstr, pero sin tener en cuenta mayÃºsculas o minÃºsculas.

strlen -> Muestra el largo de una cadena.

strnatcasecmp -> Compara cadenas insensibles a mayÃºsculas y minÃºsculas usando un algoritmo de "orden natural".

strnatcmp -> Compara cadenas usando un algoritmo de "orden natural".

strncasecmp -> Compara de los primeros n caracteres de cadenas.

strncmp -> Compara de los n primeros caraceteres de cadenas, son seguridad binaria.

strpos -> Encuentra la posiciÃ³n de la primera apariciÃ³n de una cadena en otra cadena.
strrchr -> Encuentra la Ãºltima apariciÃ³n de una cadena en otra cadena.
strrev -> Invierte el orden de una cadena.
strripos -> Encuentra la posiciÃ³n de la Ãºltima ocurrencia de una cadena en otra, insensible a mayÃºsculas y minÃºsculas.
strrpos -> Encuentr la posiciÃ³n de la Ãºltima apariciÃ³n de un carÃ¡cter en una cadena.
strspn -> Encuentra la longitud del segmento inicial que coincide con la mÃ¡scara.
strstr -> Encuentra la primera apariciÃ³n de una cadena.
strtok -> Didive una cadena en elementos.
strtolower -> Pasa a minÃºsculas una cadena.
strtoupper -> Pasa a mayÃºsculas una cadena.
strtr -> Traduce ciertos caracteres.
substr_compare -> Compara 2 cadenas, segura con material binario, opcionalmente insensible a mayÃºsculas y minpusculas, a partir de un desplezamiento y hasta un lÃmite de caracteres.
substr_count -> Cuenta el nÃºmero de apariciones de una subcadena en una cadena.
substr_replace -> Sustituye un texto en una parte de una cadena.
substr -> Devuelve parte de una cadena.
trim -> Elimina espacios del principio y del fin de una cadena.
ucfirst -> Pasa a mayÃºsculas el primer caracter de una cadena.
ucwords -> Pasa a mayÃºsculas la primera letra de cada palabra de una cadena.
vprintf -> Imprime una cadena con formato.
vsprintf -> Devuelve una cadena con formato.
wordwrap -> Corta una cadena en un nÃºmero dado de caracteres usando un caracter de ruptura de cadenas.

FUNCIONES PARA EL TRATAMIENTO DE URL

base64_decode -> Decodifica datos cifrados con el tipo MIME base64.
base64_enocde -> Codifica datos en MIME base64.
get_meta_tags -> Extrae todo el contenido de atributos de etiquetas meta de un archivo y devuelve una matriz, por ejemplo keywords o description.
http_build_query -> Genera una cadena de consulta codificada estilo URL.
parse_url -> Analiza una URL y nos devuelve sus componentes.
rawurldecode -> Decodifica cadenas codificas previamente estilo URL.
rawurlencode -> Codifica una cadena al estilo URL de acuerdo con el RFC 1738.
urldecode -> Decodifica URL cifradas en una cadena de texto.
urlencode -> Codifia una URL en una cadena de texto.

FUNCIONES DE VARIABLES

doubleval -> Obtiene el valor double de una variable.
empty -> Indica si una variable estÃ¡ definida.
floatval -> Obtiene el valor flotante de una variable.
get_defined_vars -> Devuelve un array con todas las variables definidas.
get_resource_type -> Devuelve el tipo de recurso.
gettype -> Muestra el tipo de una variable.
import_request_variables -> IMporta variables GET/POST/Cookie en el contezto global.
intval -> Recupera el valor entero de una variable.
is_array -> Verifica si una variable es un array.
is_bool -> Verifica si una variable es de tipo booleana.
is_callable -> Verifica que los contenidos de una variable puedan ser llamados como una funciÃ³n.
is_double -> Verifica si una variable es un valor double.
is_float -> Verifica si una variable es un flotante.
is_int -> Verifica si una variable es un valor entero.
is_integer -> Verifica si una variable es un valor entero.

is_long -> Verifica si una variable es un valor entero.
is_null -> Verifica si una variable es nula.
is_numeric -> Verifica si una variable es un nÃºmero o una cadena nÃºmerica.
is_object -> Verifica si una variable es un objeto.
is_real -> Verifica si una variable es un nÃºmero real.
is_resource -> Verifica si una variable es un recurso.
is_scalar -> Verifica si una variable es un escalar.
is_string -> Verifica si una variable es una cadena de caracteres.
isset -> Determina si una variable estÃ¡ definida.
print_r -> Imprime informaciÃ³n legible para humanos sobre una variable.
serialize -> Genera una representaciÃ³n apta para almacenamiento de un valor.
settype -> Permite establecer el tipo de una variable.
strval -> Obtiene una cadena de caracteres a partir de una variable.
unserialize -> Crea u valor PHP a partir de una representaciÃ³n almacenada.
unset -> Destruye una variable pasada como parÃ¡metro.
var_dump -> Vuelca la informaciÃ³n sobre una variable.
var_export -> Imprime o devuelve la representaciÃ³n de cadena de una variable.

FUNCIONES PARA EXPRESIONES REGULARES

ereg_replace -> Reemplaza expresiones regulares.
ereg -> Busca coincidencia de expresiones regulares.
eregi_replace -> Reemplaza expresiones regularse sin diferencias mayÃºsculas ni minÃºsculas.
eregi -> Busca coincidencias de expresiones regulares sin diferencias mayÃºsculas ni minÃºsculas.
split -> Divide la cadena en elementos de un array.
spliti -> Separa una cadena en un matriz mediante una expresiÃ³n regular, no sensible a mayÃºsculas ni minÃºsculas.
sql_regcase -> Construye una expresiÃ³n regular para buscar coincidencias sin diferencias mayÃºsculas ni minÃºsculas

Ejemplos de funciones de librería

- **abs (valor absoluto)**

```
<?php
  $abs=abs (-876);
  echo”$abs”
?>
```

- **acos (arco coseno)**

```
<?php
 $arcos=acos (1);
 echo”$arcos”;
?>
```

- **asin (arco seno)**

```
<?php
 $arsin=asin (1);
 echo”$arsin”;
?>
```

- **bindec (binario a decimal)**

```
<?php
echo bindec (00111110111);
echo bindec (10101001);
?>
```

- **decbin (decimal a binario)**

```
<?php
echo decbin (14);
echo decbin (22);
?>
```

- **ceil (redondeo de fracciones hacia arriba)**

```
<?php
```

```
echo ceil (2.7);
echo ceil (3.78);
?>
```

- **dechex (decimal a hexadecimal)**

```
<?php
echo dechex (17);
echo dechex (78);
?>
```

- **decot (decimal a octal)**

```
<?php
echo decot (27);
echo decot (378);
?>
```

- **deg2rad (gradoas a radianes)**

```
<?php
echo deg2rad (27);
echo deg2rad (38);
?>
```

- **exp (exponencial)**

```
<?php
echo exp (2.1);
echo exp (3.8);
?>
```

- **floor (redondea fracciones hacia abajo)**

```
<?php
echo floor (2.7);
echo floor (3.1);
?>
```

- **fmod (devuelve el residuo de modulo de la división)**

```
<?php
```

```
$x=2.3;
$y=1.1;
echo fmod ($x,$y);
?>
```

- **hypot (calcula la hipotemusa de un triángulo cuadrado)**

```
<?php
función hypo ( ) {
  $sum=0;
  Foreach (func_get_args ( ) as $dimension) {
    if (!¡s_numeric ($dimension))return -1;
   $sum+=pow ($dimension, 2);
  }
  Return sqrt ($sum);
}
printf hypo (2,2,3);
?>
```

- **is_nan (encuentra si un valor no es número)**

```
<?php
$nan=2.7;
var_dump ($nan,is_nan($nan));
?>
```

- **pow (eleva un número a lo que quieras)**

```
<?php
echo pow (2,7);
echo ceil (3,78);
?>
```

- **rad2deg (convierte de radianes a grados)**

```
<?php
echo rad2deg (M_PI_3);
?>
```

- **rand (genera un número aleatorio entero)**

```
<?php
echo rand ( ).”\n”;
echo rand ( ).”\n”;;
echo rand (2,31);
?>
```

- **round (redondea un float)**

```
<?php
echo round (2,7);
echo round (3,34);
?>
```

- **sin (seno)**

```
<?php
echo sin (deg2rad (34));
echo sin (deg2rad (4));
?>
```

- **cos (coseno)**

```
<?php
echo cos (deg2rad (134));
echo cos (deg2rad (14));
?>
```

- **sqrt (raíz cuadrada)**

```
<?php
echo sqrt (32);
echo sqrt (4);
?>
```

- **tan (tangente)**

```
<?php
echo stan (M_PI_3);
?>
```

- **pi (obtiene el valor de pi)**

```
<?php
echo pi ( );
echo M_`PI;
?>
```

- **octdec (octal a decimal)**

```
<?php
echo octdec (‘23’);
echo octdec (‘13’);
?>
```

- **min (encontrar el valor mas bajo)**

```
<?php
echo min (1,5,7,3,9);
echo min (34,6,1,8);
?>
```

- **max (encontrar el valor mas grande)**

```
<?php
echo max (1,5,7,3,9);
echo max (34,6,1,8);
?>
```

- **log (logaritmo natural)**

```
<?php
echo log (9);
echo log (34);
?>
```

- **is_infinite (encuentra si un valor es infinito)**

```
<?php
if (!¡s_defined (‘is_infinite’)) {
  función is_infinite ($val) {
     return (is_float($val) and (“$val”==’INF’ or “$val”)
```

```
?>
```

- **hexdec (hexadecimal a decimal)**

```
<?php
var_dump (hexdec (“see”));
var_dump (hexdec (“ee”));
?>
```

Ambos imprimen int (238)

- **bcadd (añade dos números arbitrarios)**

```
<?php
$x=3.4;
$y=4;
echo bcadd ($x,$y);
echo bcadd ($x,$y,3);
?>
```

- **bccomp (compara dos números arbitrarios)**

```
<?php
echo bccomp (1,5,);
echo bccomp (3.4,6,1);
?>
```

- **bcdiv (divide dos números arbitrarios)**

```
<?php
echo bcdiv (1,5);
echo bcdiv (34,6,1);
?>
```

- **bcmul (multiplica dos números arbitarios)**

```
<?php
echo min (1,5.73,9);
echo min (23,6);
?>
```

- **bcsub (resta un número de otro arbitrario)**

```
<?php
echo bcsub (4,3.4);
echo bcsub (1,8);
?>
```

- **gmp_gcd (calcula máximo común divisor)**

```
<?php
$gcd=gmp_gcd (2,3);
echo gmp_gcd ($gcd);
?>
```

- **gmp_fact (factorial)**

```
<?php
echo gmp_fact (12);
?>
```

- **gmp_neg (número negativo)**

```
<?php
$neg=gmp_neg (-3);
echo gmp_strval ($neg);
?>
```

- **gmp_prob_prime (dice si el número es primo)**

```
<?php
echo gmp_prob_prime (8);
?>
```

- **acosh (arco coseno hiperbolico)**

```
<?php
echo acosh (1);
?>
```

- **asinh (arco seno hiperbolico)**

```
<?php
echo asinh (1);
?>
```

- **atanh (arco tangente hiperbolico)**

```
<?php
echo atanh (1,3);
?>
```

- **gmp_perfect_square (cuadrado perfecto)**

```
<?php
Var_dump (gmp_perfet_square(7));
?>
```

- **gmp_div_r (residuo de la división de números)**

```
<?php
$div=gmp_div_r (103,34)
echo gmp_strval ($div);
?>
```

FORMULARIOS CON PHP

Los formularios no forman parte de PHP, sino del lenguaje estándar de Internet, HTML. Vamos a dedicar en este capítulo algunas líneas al HTML, para entrar posteriormente a tratarlos con PHP.

Todo formulario comienza con la etiqueta <FORM ACTION="lo_que_sea.php" METHOD="post/get"> . Con. Con ACTION indicamos el script que va procesar la información que recogemos en el formulario, mientras que METHOD nos indica si el usuario del formulario va ha enviar datos (post) o recogerlos (get). La etiqueta <FORM> indica el final del formulario.

A partir de la etiqueta <FORM> vienen los campos de entrada de datos que pueden ser:

Cuadro de texto

<input type="text" name="nombre" size="20" value="jose">

Cuadro de texto con barras de desplazamiento

<textarea rows="5" name="descripcion" cols="20">Es de color rojo</textarea>

Casilla de verificación

<input type="checkbox" name="cambiar" value="ON">

Botón de opción

<input type="radio" value="azul" checked name="color">

Menú desplegable

<select size="1" name="dia">
<option selected value="lunes">lunes</option>
<option>martes</option>
<option value="miercoles">miércoles</option>
</select>

Boton de comando

<input type="submit" value="enviar" name="enviar">

Campo oculto

<input type="hidden" name="edad" value="55">

Este último tipo de campo resulta especialmente útil cuando queremos pasar datos ocultos en un formulario.

Como habrás observado todos lo tipos de campo tienen un modificador llamado name, que no es otro que el nombre de la variable con la cual recogeremos los datos en el script indicado por el modificador ACTION de la etiqueta FORM FORM, con value establecemos un valor por defecto.

A continuación veamos un ejemplo, para lo cual crearemos un formulario en HTML como el que sigue y lo llamaremos formulario.htm.

```
<HTML>
<BODY>
<FORM METHOD="post" ACTION="mis_datos.php">
<input type="hidden" name="edad" value="55">
<p>Tu nombre <input type="text" name="nombre" size="30" value="jose"></p>
<p>Tu sistema favorito
<select size="1" name="sistema">
<option selected value="Linux">Linux</option>
<option value="Unix">Unix</option>
<option value="Macintosh">Macintosh</option>
<option value=&qmargin-left: 75"><option value="Windows">Windows</option>
</select></p>
<p>¿Te gusta el futbol? <input type="checkbox" name="futbol" value="ON"></p>
<p>¿Cual es tu sexo?</p>
<blockquote>
<p>Hombre<input type="radio" value="hombre" checked name="sexo"></p>
<p>Mujer <input type="radio" name="sexo" value="mujer"></p>
</blockquote>
<p>Aficiones</p>
<p><textarea rows="5" name="aficiones" cols="28"></textarea></p>
```

```
<p><input type="submit" value="Enviar datos" name="enviar">
<input type="res-left: 50"> <input type="reset" value="Restablecer" name="B2"></p>
</FORM>
</BODY>
<HTML>
```

Y ahora creemos el script PHP llamado desde le formulario mis_datos.php.

Todos los datos se encuentran en la variable $_POST, ya que el formulario está enviado por el método post.

```
<?PHP;
if (isset ($_POST ['enviar']) {
echo "Hola <b>" . $_POST ['nombre'] . "</b> que tal estás<BR>n";
echo "Eres " . $_POST ['sexo'] . "<BR>n";
echo "Tienes " . $_POST ['edad'] . "<BR>n";
echo "Tu sistema favorito es " . $_POST ['sistema'] . "<BR>n";
if (isset($_POST ['futbol']) {
echo "Te gusta el futbol <BR>n";
} else codigo" style="margin-left: 50">} else {
echo "NO te gusta el futbol <BR>n";
}
if ($_POST ['aficiones'] != "") {
echo "Tus aficiones son: <BR>n";
echo nl2br ($_POST ['aficiones']);
} else {
echo "NO tienes aficiones <BR>n";
}
}
echo "<a href='formulario.htm'>VOLVER AL FORMULARIO</a>"
```

?>

Una vez rellenados los datos del formulario, pulsamos el botón Enviar datos, con lo que el campo enviar toma lo que su etiqueta value indica, es decir enviar="Enviar datos". En nuestro script lo primero que evaluamos es que se haya enviado el formulario, y para ello nada mejor que comprobar que la variable $enviar no está vacía. Le ponemos el signo dólar delante a enviar, ponemos el signo dólar delante a enviar, ya que en PHP todas las variables se les refiere con este signo.

Hay que tener en cuenta que si fusionáramos el código de ambos ficheros, nos ahorraríamos uno, pero no, también se puede hacer en dos como lo estamos haciendo. Si la variable $enviar está vacia, enviamos el formulario.

```
<?PHP;
if ($enviar) {
echo "Hola <b>" . $nombre . "</b> que tal estás<BR>n";
echo "Eres " . $sexo . "<BR>n";
echo "Tienes " . $edad . "<BR>n";
echo "Tu sistema favorito es " . $sistema . "<BR>n";
if ($futbol) {
echo "Te gusta el futbol <BR>n";
} else {
echo "NO te gusta el futbol <BR>n";
}
if ($aficiones != "") {
< stuot;)>
echo "Tus aficiones son: <BR>n";
echo nl2br ($aficiones);
} else {
echo "NO tienes aficiones <BR>n";
```

```
}
echo "<a href='$PHP_SELF'>VOLVER AL FORMULARIO</a>"
} else {
<HTML>
<BODY>
<FORM METHOD="post" ACTION="<?PHP echo $PHP_SELF ?>">
<input type="hidden" name="edad" value="55">
<p>Tu nombre <input type="text" name="nombre" size="30" nombre" size="30"
value="jose"></p>
<p>Tu sistema favorito
<select size="1" name="sistema">
<option selected value="Linux">Linux</option>
<option value="Unix">Unix</option>
<option value="Macintosh">Macintosh</option>
<option value="Windows">Windows</option>
</select></p>
<p>¿Te gusta el futbol? <input type="checkbox" name="futbol" value="ON"></p>
<p>¿Cual es tu sexo?</p>
<blockquote>
<p>Hombre<input type="radio" value="hombre" checked name="sexo"></p>
<p>="codigo" style="margin-left: 100"><p>Mujer <input type="radio" name="sexo"
value="mujer"></p>
</blockquote>
<p>Aficiones</p>
<p><textarea rows="5" name="aficiones" cols="28"></textarea></p>
<p><input type="submit" value="Enviar datos" name="enviar">
<input type="reset" value="Restablecer" name="B2"></p>
</FORM>
</BODY>
</HTML>
<?PHP
```

```
} //fin IF
?>
```

La variable de entorno $PHP_SELF, es una variable de entorno que nos devuelve el nombre del script que estamos ejecutando. Y por último, hacer notar el uso de la función nl2br ( ), nl2br ( ), con la cuál sustituimos los retornos de carro del texto, los cuáles no reconocen los navegadores, por la etiqueta
 .

Programa que saca el cuadrado de un número

```
<html>
   <head>
      <title>numero elevado</title>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
   </head>
   <body>
<?php
if ($_SERVER['REQUEST_METHOD']=='POST')
{
   $numero = floatval ($_POST ['numero']);
   $cuadrado=$numero*$numero;
   echo "Valor de cuadrado: $cuadrado<br>\n";
}
?>
      <form method="post">
         <table style="text-align: left; margin-left: auto; margin-right: auto;" border="1" cellpadding="1" cellspacing="1">
            <tbody>
               <tr><td>Ingrese el valor de numero</td><td><input name="numero"></td></tr>
```

```
                <tr align="center"><td colspan="2" rowspan="1"><input
value="Procesar" type="submit"></td></tr>
            </tbody>
          </table>
        </form>
    </body>
</html>
```

Programa que metiendo cualquiera de dos números nos da la suma

```
<!DOCTYPE HTML>
<html lang="es-ES">
<head>
        <meta charset="UTF-8">
        <title>Suma de 2 numeros en PHP</title>
</head>
<body>
<H2>Suma de 2 numeros en PHP</H2>
<form action="suma.php" method="POST">
        <table>
        <tr>
                <td><input type="text" name="numero1"></td>
        </tr>
        <tr>
                <td><input type="text" name="numero2"></td>
        </tr>
        <tr>
                <td> <input type="submit" value="sumar"> </td>
        </tr>
</table>
</form>
```

```
</body>
</html>
<?php
        if($_POST)
        {
                $num1 = $_POST
                ['numero1'];
                $num2 = $_POST
                ['numero2'];
                $suma = $num1
                + $num2;
                echo "La suma de ".$num1." y ".$num2." es ".$suma;
        }
```

30 PROGRAMAS PROPUESTOS

1. Sacar el volumen de un prisma triangular con unas de sus caras rectangulares con medidas de 4 de ancho y 7 de largo.
2. Sacar el volumen de un cono truncado con cualquier tipo de medidas.
3. Conociendo dos lados de un triángulo rectangular sacar el tercer lado.
4. Conociendo el volumen de una piscina circular y conociendo el radio de la base sacar la altura de la piscina, sabiendo que el grosor de la piscina es de 2.5 centímetros.
5. Si don Martin invirtió 93 504 pesos en la compra de animales y tiene una ganancia de 32.3% del capital en un un año. Suponiendo que la ganancia es constante, sacar la ganancia ¿qué ganancia tuvo en 7 meses?, ¿En 8 años con 2 meses y 7 días?
6. Sacar la distancia de dos puntos en R^3
7. Sacar la pendiente de dos puntos en R^3

8. Sacar la suma de la serie ∑(1+(n∗√n)^{(1/3)}) des de n=2 hasta n=5.
9. Calcular la factorial de cualquier número con el ciclo do while.
10. Si meto al banco 1200 pesos con un 13% de interés mensual sacar el dinero que tendré en año y medio.
11. Sacar la suma de la sucesión (1/2)-(2/3)+(3/4)-(4/5).
12. Sacar la suma de la serie ∑(2^n)/(n+1) desde n=1 hasta n=3.
13. Sacar los ángulos de un triángulo rectángulo conociendo 2 de sus lados.
14. Sacar la suma de la tabla del 7 y dividirla entre el factorial del 7. Con el ciclo for.
15. Multiplicar la suma de los números pares del 1 al 25 por la suma de los números impares del 3 al 18.
16. Sacar la cantidad de agua que tiene un cilindro con radio de la base 3 m. y altura 17.45 m. conociendo que contiene un 37% del volumen total.
17. Sacar los múltiplos del 7 del 7 al 57 y decir si la suma de dichos números es número impar o par. Análogamente con el factorial de dicho número.
18. Sacar la longitud de una manzana sabiendo que mide 19 tablas con una distancia de un metro la primera tabla, la mitad de la primera tabla mide la segunda, la mitad de la segunda tabla mide la tercera y así sucesivamente hasta la tabla 89.
19. Calcular la longitud de un vector en R^3
20. Calcular el ángulo formado por dos vectores dados.
21. Calcular el área de cualquier triangulo rectangular conociendo dos de sus ángulos.
22. Tenemos un deposito primario de 1700 y sabemos que el interés mensual es de 2.3% calcular la ganancia obtenida en 2 años con 3 meses.
23. Lo anterior pero usando el ciclo while.
24. Sacar el número mayor y menor de números usando etiquetas y usando el ciclo for.
25. Dame el volumen de la esfera sabiendo su radio.
26. Saca la tabla del 8 con el ciclo while.
27. Saca la mediana desde el 3 hasta el 23.

28. Saca el promedio de 7 numeros con el ciclo for y calcule su factorial.
29. Saca las raíces de x^2+x+1=0.
30. Conociendo x sacar x^4,x^6,x^-5,x^-1 usando ciclo for.

BIBLIOGRAFÍA

Sæther Bakken, Stig. Aulbach, Alexander. Manual php. Publicado 08-07-2002. 1719 Páginas.

Webgrafia

www.wikipedia.com
www.aprenderaprogramar.com
http://www.bisente.com/documentos/mysql-postgres.html
http://definicion.de/pagina-web/#ixzz309K1RQOy
http://es.tldp.org/Manuales-LuCAS/manual_PHP/manual_PHP/
http://linux.ciberaula.com/articulo/linux_apache_intro
www.alegsa.com
http://www.mundomanuales.com/manuales/PHPManualCompleto.pdf
http://www.matematica.ciens.ucv.ve/files/Manuales/Manuales/Programacion%20Web%20-%20Introducci%F3n%20al%20PHP.pdf
http://gobiernoti.wordpress.com/2011/10/04/tipos-de-redes-informaticas/
http://www.postgresql.org.es/sobre_postgresql
http://www.foro-creativo.com/viewtopic.php?f=13&t=198
http://oldblog.jesusyepes.com/administracion-de-sistemas/el-servidor-web-perfecto-macos-mamp/
http://webplusplus.blogspot.mx/2011/11/comparativa-python-vs-php-vs-java.html
http://www.aulaclic.es/flashMX/t_17_1.htm
http://www.monografias.com/trabajos7/html/html.shtml

http://www.comocreartuweb.com/curso-php-y-msql/que-es-el-php/escribir-codigo.html
http://www.desarrolloweb.com/en-directo/introduccion-a-codex-7289.html

CAPÍTULO 3

Dr. Luis manuel Martínez Hernández
Catedrático de la Universidad Pedagógica de Duango y
La Universidad Juárez del Estado de Durango

M.C. María Elizabeth Leyva Arellano
Catedrático de la Universidad Juárez del Estado de Durango

Dr. Arturo Barraza Macias
Catedrático de la Universidad Pedagógica de Durango

EL CONCEPTO DE NETWORKING

En su nivel más elemental, una red consiste en dos computadoras conectadas mediante un cable para que puedan compartir datos. Todo el Networking, no importa cuán sofisticado, procede de ese simple sistema. Mientras la idea de dos computadoras conectadas por cable puede no parecer extraordinaria, en retrospectiva, fue un gran logro en comunicaciones.

Networking surge de la necesidad de compartir datos en una forma oportuna. Las computadoras personales son buenas herramientas de trabajo para producir datos, hojas de cálculo, gráficos y otros tipos de información, pero no te permiten compartir rápidamente los datos que has producido. Sin una red, los documentos tienen que ser impresos para que otros los editen o los usen. En el mejor de los casos, entregar archivos en diskettes a otros para que los copien a sus computadoras. Si otros hacen cambios en el documento no hay manera de mezclarlos. Esto fue, y todavía es, llamado trabajo en un entorno aislado (stand alone).

Networking trabajo en red compartición de datos, impresoras, modems, faxes, gráficos. Lan, varias pc´s que corresponden a una única ubicación física. Solo se utiliza un medio (cable). Wan, redes distintas.

Por qué usar una red en términos económicos compartir hardware. En términos de datos compartir aplicaciones (un schedule o agenda).

Si un trabajador aislado conectase su computadora a otras computadoras, podría compartir los datos en las otras computadoras e impresoras. Un grupo de computadoras y otros aparatos conectados juntos es llamado una red, “network”, y el concepto de computadoras conectadas compartiendo recursos es llamado “networking”.

Las computadoras que son parte de una red pueden compartir datos, mensajes, gráficos, impresoras, fax, modems y otros recursos hardware. Esta lista está creciendo constantemente con las nuevas vías encontradas para compartir y comunicarse por medio de computadoras.

REDES DE ÁREA LOCAL

Las redes empezaron siendo pequeñas, con quizás diez computadoras conectadas junto a una impresora. La tecnología limitaba el tamaño de la red, incluyendo el número de computadoras conectadas, así como la distancia física que podría cubrir la red. Por ejemplo, en los primeros años de los 80 el más popular método de cableado permitía como 30 usuarios en una longitud de cable de alrededor de 200 metros (600 pies). Por lo que una red podía estar en un único piso de oficina o dentro de una pequeña compañía. Para muy pequeñas empresas hoy, ésta configuración es todavía adecuada. Este tipo de red, dentro de un área limitada, es conocida como una red de área local (Lan).

LA EXPANSIÓN DE LAS REDES

Las primeras Lans no podían soportar adecuadamente las necesidades de grandes negocios con oficinas en varios lugares. Como las ventajas del Networking llegaron a ser conocidas y más aplicaciones fueron desarrolladas para entorno de red, los empresarios vieron la necesidad de expandir sus redes para mantenerse competitivos. Las redes de hoy han construido bloques de grandes sistemas. Así como el ámbito geográfico de la red crece conectando usuarios en diferentes ciudades o diferentes estados, la Lan crece en una Red de Área Amplia (Wan). El número de usuarios en la red de una compañía puede ahora crecer de 10 a miles. Hoy, la mayoría de los negocios más importantes almacenan y comparten vastas cantidades de datos cruciales en un entorno de red, que es por

lo que las redes son actualmente tan esenciales para los empresarios como las mecanógrafas y los archivos lo fueron.

INTERNET PROTOCOL

Vamos a usar el término “IP” para referirnos a las redes diseñadas para trabajar con TCP/IP. IP es el protocolo a nivel de red de la familia de protocolos TCP/IP, usados en Internet. Es una práctica común usar el término “IP” cuando nos referimos a direcciones, enrutamiento y otros elementos a nivel de red. La distinción muchas veces no es lo suficientemente clara. Así que, en la práctica, los términos Internet TCP/IP e IP pueden parecer incluso intercambiables.

Los términos “paquete” y “datagrama” también suelen parecer intercambiables. Conceptualmente, un “paquete” es la unidad física de más bajo nivel, mientras que “datagrama” se refiere a la unidad de datos a nivel IP. Sin embargo, en la mayoría de las redes no se pueden distinguir porque coinciden, así que la gente suele usar los dos términos indistintamente.

Otro término "conflictivo" es el de pasarela ("gateway") y enrutador ("router"). Pasarela es el término original usado en Internet. Sin embargo, la comunidad OSI empezó a usar esta palabra con un significado distinto, así que la gente empezó a usar enrutador para evitar dicha ambigüedad. Nosotros, no obstante, seguiremos usando el término gateway.

QUE ES UNA RED

Las organizaciones implementan redes primariamente para compartir recursos y habilitar comunicación online. Los recursos incluyen datos, aplicaciones y periféricos. Un periférico es un dispositivo como un disco externo, impresora, ratón, modem o joystick. Comunicación OnLine incluye enviar mensajes de acá para allá, o e-mail.

Impresoras y otros Periféricos

Antes de la llegada de las redes, la gente necesitaba su propia e individual impresora, plotter y otros periféricos. Antes de que existieran las redes, la única forma de compartir una impresora era que la gente hiciera turno en la computadora conectada a la impresora.

Las redes ahora hacen posible para mucha gente compartir ambos, datos y periféricos, simultáneamente. Si mucha gente usa una impresora pueden utilizar la impresora disponible en la red.

Datos

Antes de que existieran las redes, la gente sólo quería compartir información, misma que estaba limitada a:

Decir a los otros la información (comunicación por voz)
Escribir memos.
Poner la información en un diskette, llevar físicamente el disco a otra computadora y entonces copiar los datos en esa computadora.

Las redes pueden reducir la necesidad de comunicación en papel y hacer más cercano cualquier tipo de dato disponible para cada usuario que lo necesite.

Aplicaciones

Las redes pueden ser usadas para estandarizar aplicaciones, como un procesador de textos, para asegurar que todos en la red están usando la misma aplicación y la misma versión de esa aplicación. Estandarizarse en unas aplicaciones puede

simplificar el soporte. Es más fácil conocer una aplicación muy bien que intentar aprender cuatro o cinco diferentes. Es también más fácil ocuparse de sólo una versión de una aplicación y ajustar todas las computadoras de la misma forma. Algunos empresarios invierten en redes a causa de los programas de e-mail y agendas. Los encargados pueden usar esas utilidades para comunicar rápida y efectivamente con un gran número de gente para organizar y manejar una compañía entera mucho más fácilmente de lo que era antes posible.

PRINCIPALES TIPOS DE REDES

En general, todas las redes tienen ciertos componentes, funciones y prestaciones en común.

Esto incluye:

Servidores. Computadoras que proporcionan recursos compartidos a los usuarios de la red.
Clientes. Computadoras que acceden a los recursos compartidos de la red, provistos por un servidor.
Medio. La vía por la que las computadoras están conectados.
Datos Compartidos. Archivos proporcionados por los servidores a través de la red.
Impresoras y otros periféricos compartidos. Otros recursos proporcionados por los servidores.
Recursos. Archivos, impresoras y otros items para ser usados por los usuarios de la red.

Incluso con esas similitudes, las redes pueden ser divididas en dos amplias categorías:

Peer-to-Peer (Pares a Pares)

Basadas en Servidor.

La distinción entre Peer-to-Peer y basada en servidor es importante porque cada una tiene diferentes capacidades. El tipo de red que usted implemente dependerá de numerosos factores, incluyendo el tamaño de la organización, nivel de seguridad requerido, tipo de negocio, nivel de soporte administrativo disponible, volumen de tráfico de red, necesidades de los usuarios de red y presupuesto de la red.

Se tiene en cuenta el costo del servidor. Peer Server.
En Peer no se puede expandir a varias Lan, trabajo en grupo.

REDES PEER-TO-PEER

En una red Peer-to-Peer no hay servidores dedicados o jerarquía entre las computadoras. Todos las computadoras son iguales y además son conocidas como pares (peers). Normalmente, cada computadora funciona como un cliente y como un servidor, y no hay una asignada a ser un administrador responsable de red en su totalidad. El usuario en cada computadora determina que datos en su computadora serán compartidos en la red.

Tamaño. Las redes Peer-to-Peer son llamadas también Grupos de Trabajo. El término implica un pequeño grupo de gente. En una red Peer-to-Peer, hay, típicamente, menos de 10 computadoras en la red.

Costo. Las redes Peer-to-Peer son relativamente simples. Debido a que cada computadora funciona como un cliente y un servidor, no hay necesidad de un potente servidor central, o de los otros componentes requeridos para una red de alta capacidad. Las redes Peer-to-Peer pueden ser menos caras que las redes basadas en servidores.

Sistemas Operativos Peer-to-Peer. En una red Peer-to-Peer, el software de red no necesita tener el mismo nivel de prestaciones y seguridad que el software diseñado para redes de servidor dedicado. Los servidores dedicados funcionan solo como servidores y no son usados como un cliente o estación de trabajo. En sistemas operativos como NT Workstation, MS Windows para Trabajo en Grupo y Windows-95, Peer-to-Peer networking está implementado dentro del sistema operativo. No se requiere software adicional para establecer una red Peer-to-Peer.

Implementación. En un entorno Peer-to-Peer típico, hay un número de cuestiones que tienen solución estandar. Estas implementaciones incluyen:

Computadoras situadas en las mesas de los usuarios.

Los usuarios actúan como su propio administrador y planean su propia seguridad. Es usado un simple y fácilmente visible sistema de cableado, que conecta computadora a computadora en la red.

Donde es apropiada la Peer-to-Peer. Son buena elección para entornos donde: hay menos de 10 usuarios. Los usuarios están situados todos en el mismo área común. La seguridad no es un problema. La organización y la red tendrán un crecimiento limitado dentro de un previsible futuro.

Considerando estas guias, hay veces que una red Peer-to-Peer será una mejor solución que una red basada en servidor.

Consideraciones Peer-to-Peer. Mientras una red Peer-to-Peer puede cumplir las necesidades de organizaciones pequeñas, éste tipo de propuesta puede ser inapropiada en ciertos entornos. Las siguientes áreas de networking ilustran algunas cuestiones Peer-to-Peer que un planificador de red deberá resolver antes de decidir qué tipo de red implementar.

Administración. La administración de una red abarca una variedad de tareas incluyendo el manejo de usuarios y seguridad. Hacer disponibles los recursos. Aplicaciones y datos. Instalar y actualizar software de aplicación.

En una típica red Peer-to-Peer no hay un encargado del sistema que supervise la administración de la red completa. Cada usuario administra su propia computadora.

Compartiendo recursos. Todos los usuarios pueden compartir cualquiera de sus recursos de la forma que escojan. Esos recursos incluyen datos en directorios compartidos, impresoras, tarjetas de fax, etc.

Requerimientos de servidor. En un entorno Peer, cada computadora puede usar un gran porcentaje de sus recursos para soportar al usuario local (el usuario en la computadora). Usar recursos adicionales para soportar cada usuario remoto accediendo a sus recursos (un usuario accediendo al servidor sobre la red.) Una red basada en servidor necesita más potencia, los servidores dedicados a satisfacer las demandas de todos los clientes de la red.

Seguridad. La seguridad consiste en poner un password en un recurso, como es un directorio que está siendo compartido en la red. Debido a que todos los usuarios Peer-to-Peer establecen su propia seguridad, y las comparticiones pueden existir en cualquier computadora no solo en un servidor centralizado, el control centralizado es muy dificil. Esto tiene un gran impacto en la seguridad de la red porque algunos usuarios pueden no implementar seguridad. Si la seguridad es importante, se debería considerar una red basada en servidor.

Enseñanza. Debido a que cada computadora en un entorno Peer-to-Peer puede actuar como servidor y cliente, los usuarios deberían ser entrenados antes de que pudieran ser capaces de funcionar propiamente como usuario y administradores de su computadora.

REDES BASADAS EN SERVIDORES

En un entorno con más de 10 usuarios, una red Peer-to.Peer -con computadoras actuando como servidores y clientes- probablemente no será adecuada. Por consiguiente, la mayoría de las redes tienen servidores dedicados. Un servidor dedicado es uno que solo funciona como un servidor y no es usado como cliente o estación de trabajo. Los servidores son dedicados porque están optimizados para dar servicio rápidamente a las peticiones desde los clientes de red y para asegurar la seguridad de archivos y directorios. Las redes basadas en servidor han llegado a ser el modelo estándar para Networking y serán usados como un modelo primario. En cuanto las redes incrementan el tráfico y el tamaño, se necesitará más de un servidor. La diseminación de tareas entre varios servidores asegura que cada tarea será procesada de la manera más eficiente posible.

Servidores especializados

La variedad de tares que los servidores deben ejecutar es variada y compleja. Los servidores para redes grandes han llegado a ser especializados para acomodar las necesidades de expansión de los usuarios. Por ejemplo, en una red de Windows NT, los diferentes tipos de servidores.

Servidores de archivos e impresión

Los servidores de archivos e impresión manejan los accesos de usuarios y el uso de los recursos de archivos e impresoras. Por ejemplo, si usted está ejecutando una aplicación de proceso de textos, la aplicación podría ejecutarse en su computadora. El documento almacenado en el servidor de archivos e impresión es cargado en la memoria de su computadora para que usted pueda editarlo o usarlo

localmente. En otras palabras, los servidores de archivos e impresión son para almacenamiento de datos y archivos.

Servidores de Aplicación

Los servidores de aplicación hacen el lado del servidor de las aplicaciones cliente/servidor, así como los datos, disponible para los clientes. Por ejemplo, los servidores almacenan grandes cantidades de datos que están estructurados para hacer fácil su recuperación. Esto difiere de un servidor de archivos e impresión.

Con un servidor de archivos e impresión, el dato o archivo es descargado a la computadora que hace la petición. Con un servidor de aplicaciones, la base de datos permanece en el servidor y sólo los resultados de una petición son descargados a la computadora que la hace.

Una aplicación cliente funcionando localmente podría acceder a datos en el servidor de aplicaciones. En lugar de ser descargada la base de datos completa desde el servidor a su computadora local, sólo los resultados de su petición serán cargados en su computadora.

Servidores de Correo

Los servidores de correo manejan mensajería electrónica entre usuarios de la red.

Servidores de Fax

Los servidores de fax manejan tráfico de fax hacia adentro y hacia afuera de la red, compartiendo uno o más modem-fax.

Servidores de Comunicaciones

Los servidores de comunicaciones manejan flujo de datos y mensajes de correo entre el propio servidor de la red y otras redes, mainframes o usuarios remotos usando modems y líneas de teléfono para llamar y conectarse al servidor.

Los Servicios de Directorio sirven para ayudar a los usuarios a localizar, almacenar y asegurar la información en la red Windows NT combina computadoras en grupos lógicos, llamados dominios, que permiten a cualquier usuario en la red tener acceso a cualquier recurso en la misma.

Planear varios servidores viene a ser importante con una red en crecimiento. El planificador debe tener en cuenta cualquier crecimiento anticipado de la red para que la utilización de la red no se interrumpa si el rol de un servidor específico necesita ser cambiado.

La importancia del Software

Un servidor de red y el sistema operativo trabajan juntos como una unidad. No importa cuán potente o avanzado sea un servidor, es inútil sin un sistema operativo que pueda sacar provecho de sus recursos físicos. Ciertos sistemas operativos avanzados, como Windows NT Server, fueron diseñados para obtener ventaja del hardware de servidor más avanzado.

Windows NT lo hace de las siguientes formas:

Procesamiento Simétrico (SMP) Un sistema multiprocesador tiene más de un procesador. SMP permite que el sistema arranque y las aplicaciones necesitadas son distribuidas a través de todos los procesadores disponibles.

Soporte Multi-Plataforma Intel 386/486/Pentium, MIPS R4000, RISC, Digital Alpha.
Nombres de archivo y directorio largos. 255 caracteres.
Tamaño de archivo 16 EB (exabytes).

Tamaño de partición 16 EB.

VENTAJAS DE LAS REDES BASADAS EN SERVIDOR

Compartición de Recursos

Un servidor está diseñado para proveer acceso a muchos archivos e impresoras mientras se mantienen las prestaciones y la seguridad para el usuario. En servidores, la compartición de datos puede ser administrada centralmente y controlada. Los recursos están usualmente situados centralmente y son más fáciles de encontrar y soportar que los recursos en computadoras dispersas. Por ejemplo, en Windows NT, los recursos de directorio están compartidos a través del File Manager.

Seguridad

La seguridad es a menudo la razón primaria para elegir una propuesta basada en servidor para hacer Networking. En un entorno basado en servidor, como Windows NT, la seguridad puede ser manejada por un administrador que establece la política y la aplica a cada usuario en la red.

Backup

Debido a que los datos cruciales están centralizados en uno o unos pocos servidores, es fácil estar seguro de que los datos son puestos a salvo en intervalos regulares.

Redundancia

A través de sistemas de redundancia, los datos en cualquier servidor pueden ser duplicados y mantenidos en línea para que, si algo sucede al principal almacén de datos, una copia de backup de los datos pueda ser usada para recuperarlos.

Número de Usuarios

Una red basada en servidor puede soportar cientos de usuarios. Este tipo de red sería imposible de manejar como una red Peer-to-Peer, pero las actuales utilidades de monitorización y manejo de la red lo hacen posible para operar una red basada en servidor con un gran número de usuarios.

Consideraciones Hardware

El hardware de las computadoras Cliente puede estar limitado a las necesidades del usuario porque los clientes no necesitan memoria RAM y almacenamiento en disco adicional para proveer servicios de servidor. Una computadora cliente típico tiene al menos un 486 y de 8 a 16 Mb. de RAM.

COMBINACIÓN DE REDES

No es inusual para las redes modernas en entornos de negocios combinar las mejores características de las Peer-to-Peer y las basadas en servidor en una red.

En una red combinada, pueden trabajar juntas dos clases de sistemas operativos para proporcionar lo que muchos administradores sienten como la red más completa.

Un sistema operativo basado en servidor como Windows NT o Novell Netware es responsable de compartir las aplicaciones y los datos.

Las computadoras cliente pueden ejecutar un sistema operativo como Windows NT Workstation o Windows-95. Pueden acceder a recursos en el servidor designado y, simultáneamente, compartir sus discos duros y hacer disponibles sus datos personales.

Este tipo de redes es común, pero requiere planificación intensiva y entrenamiento para implementar propiamente y asegurar la seguridad adecuada.

CONSIDERACIONES DEL HARDWARE DEL SERVIDOR

Los recursos compartidos son el fundamento de las redes Peer-to-Peer y las basadas en servidor.

Las diferencias entre servidores Peer-to-Peer y servidores dedicados tendrán un impacto en los requerimientos hardware, cuantos usuarios son soportados en la red.

Los siguientes componentes de servidor requieren consideración cuidadosa:

Localización del recurso compartido:

Peer-to-Peer: En las computadoras de los usuarios.
Redes basadas en servidor: En los servidores dedicados.

RAM

Peer-to-Peer: Depende de las necesidades del usuario. NT Workstation requiere un mínimo de 12 Mb, con 16 Mb. recomendados. Windows-95 recomienda al menos 8 Mb. de Ram.

Redes basadas en servidor: Tanta como sea posible. Al menos 12 Mb. Los super servidores soportando cientos de usuarios no tienen menos de 64 Mb.

CPU

Peer-to-Peer: Depende de las necesidades del usuario. Debería ser al menos un 386. NT Workstation requiere un 80386/25 o superior o un procesador RISC soportado. Windows- 95 requiere un 386DX o superior.

Redes basadas en servidor: Depende del uso del servidor. Debería ser al menos un 486.

Servidores de alto rendimiento soportan múltiples procesadores.

Espacio de DISCO

Peer-to-Peer: Varía con las necesidades del usuario.

Redes basadas en servidor: Varia con las necesidades de la Organización. Tanto como sea posible, pero se debería planear su expansión. Se sugiere al menos 1 Gb. Para organizaciones pequeñas. Los super-servidores ya no se limitan a sí mismos a gigabytes. Se refieren a su capacidad en términos del número de unidades que pueden instalárseles.

DISEÑO DE LA RED

COMPOSICIÓN DE LA RED

El término topología, o más específicamente, topología de red, se refiere a la composición o distribución física de computadoras, cables y otros componentes en

la red. Topología es el término estándar que la mayoría de los profesionales usan cuando se refieren al diseño básico de una red. En adición a topología, esos preparativos pueden estar referidos a:

Distribución física
Diseño
Diagrama
Mapa

Una topología de red afecta a sus capacidades. El escoger una topología sobre otra, puede tener impacto sobre:

El tipo de equipamiento que la red necesita
Capacidades del equipamiento
Crecimiento de la red
Forma en que es manejada la red.

Desarrollar el sentido de cómo son usadas las diferentes topologías es clave para comprender las capacidades de los diferentes tipos de redes.

Las computadoras tienen que estar conectadas en orden a compartir recursos o realizar otras tareas de comunicaciones. La mayoría de las redes usan cable para conectar una computadora con otra.

Sin embargo, no es tan simple como sólo enchufar una computadora en un cable conectando otras computadoras. Diferentes tipos de cables, combinados con tarjetas de red diferentes, sistemas operativos de red y otros componentes, requieren distintos tipos de soluciones.

Una topología de red implica un número de condiciones. Por ejemplo, una topología particular puede determinar no sólo el tipo de cable usado sino como está transcurriendo el cable a través de los suelos, techos y paredes.

La topología también puede determinar cómo se comunican las computadoras en la red.
Diferentes topologías requieren diferentes métodos de comunicación, y esos métodos tienen una gran influencia en la red.

TOPOLOGÍAS ESTANDAR

Para elegir una topología hay que tener en cuenta la influencia de muchas cosas, como el uso intensivo de las aplicaciones o el número de computadoras.

Todos los diseños de red parten de tres topologías básicas:

Bus
Estrella
Anillo

Si las computadoras están conectadas en una fila a lo largo de un único cable (segmento), la topología está referida como de bus. Si las computadoras están conectadas a segmentos de cable que se ramifican desde un único punto o hub, la topología es conocida como una estrella. Si las computadoras están conectadas a un cable que forma un bucle, la topología es conocida como anillo (ring).

Mientras esas tres topologías básicas son simples en sí mismas, sus versiones en el mundo real a menudo combinan características de más de una topología y puede ser complejo.

BUS

La topología de bus es conocida también como un bus lineal. Este es el método más simple y más común de interconectar computadoras (networking). Consiste en un simple cable llamado trunk o troncal (también backbone o segmento) que conecta todas las computadoras en la red a una línea única.

Comunicación en el Bus

Las computadoras en una topología de bus se comunican direccionando datos a una computadora en particular y poniendo esos datos en el cable en forma de señales electrónicas. Para comprender cómo se comunican las computadoras en un bus, usted necesita estar familiarizado con tres conceptos:

1. Envío de la señal
2. Rebote de la señal
3. El terminador.

Enviando la señal

Los datos de la red en forma de señales electrónicas son enviados a todas las computadoras en la red; sin embargo, la información es aceptada sólo por la computadora que coincide en su dirección con la codificada en la señal original. Sólo una computadora a la vez puede enviar mensajes. Sólo si el cable no está ocupado. Se tratan las direcciones origen y destino, que son las que están grabadas en el firmware de las tarjetas. Son las direcciones MAC, únicas en el mundo para cada tarjeta.

Debido a que sólo una computadora a la vez puede enviar datos en una red en bus, el rendimiento de la red está afectado por el número de computadoras enganchadas al bus. A más computadoras conectadas al bus, más computadoras

estarán esperando para poner datos en el mismo, y más lenta será la red. A más máquinas menos rendimiento.

No hay una medida estándar sobre el impacto del número de computadoras en una red dada. El total de ralentización de una red no está únicamente relacionado con el número de computadoras en la misma. Depende de numerosos factores, incluyendo:
Capacidad del hardware de las computadoras de la red.
Número de veces que las computadoras transmiten datos.
Tipo de aplicaciones siendo ejecutadas en la red.
Tipos de cable usados en la red.
Distancia entre las computadoras en la red.

El bus es una topología PASIVA

Las computadoras en un bus sólo escuchan los datos que estén siendo enviados por la red. Ellos no son responsables de mover los datos desde una computadora al siguiente. Si una computadora falla, no afecta al resto de la red. En una topología activa las computadoras regeneran las señales y mueven los datos a lo largo de la red.

Rebote de la señal

Debido a que los datos, o la señal electrónica, son enviados por toda la red, viajan desde un extremo del cable al otro. Si fuera permitido que la señal continuara ininterrumpidamente, podría rebotar para atrás y para delante a lo largo del cable e impedir a otros computadoras enviar señales. Por ello, la señal debe ser parada después de que haya tenido la oportunidad de alcanzar la dirección de destino apropiada.

El Terminador

Para parar el rebote de la señal, un componente llamado terminador, es situado en cada extremo del cable para absorber las señales libres. Absorbiendo las señales se limpia el cable para que otros componentes puedan enviar datos.

Cada fin de cable en la red debe estar conectado a algo. Por ejemplo, un fin de cable puede estar enchufado en una computadora o en un conector para extender la longitud del cable.

Cualquier final abierto, no enchufado a algo, debe ser terminado para prevenir el rebote de la señal. Rebote de la señal signal bounds terminadores, porque si no hay colisiones. Sirve con una sola toma de tierra.

Un backbone sirve para unir dos buses (es el mismo bus), por ejemplo, el trozo de cable que une un piso con otro.

Rompiendo la comunicación en la red

Un corte en el cable puede ocurrir si está físicamente roto en dos trozos o si un fin del cable está desconectado. En cualquier caso, uno o más finales del cable no tienen un terminador, la señal rebota, y toda la actividad de la red parará. A esto se llama “caer” la red.

Las computadoras en la red todavía serán capaces de funcionar en modo local, pero mientras el segmento esté roto, no serán capaces de comunicarse con los otros.

Expansión de la LAN

Cuando la ubicación de la red se hace más grande, la LAN podrá necesitar crecer. El cable en la topología bus puede ser extendido por uno de los dos métodos siguientes.

Un componente llamado conector “barrilete” puede conectar dos trozos de cable juntos para hacer un cable más largo. Sin embargo, los conectores debilitan la señal y deberían ser usados escasamente. Es mucho mejor comprar un cable continuo que conectar varios trozos pequeños con conectores. De hecho usar demasiados conectores puede evitar que la señal sea correctamente recibida.

Un componente llamado repetidor se puede usar para conectar dos cables. Un repetidor, amplifica la señal antes de enviarla por su camino. Un repetidor es mejor que un conector o un trozo largo de cable porque permite que la señal viaje más lejos y todavía sea recibida correctamente.

Anillo

La topología en anillo conecta computadoras en un círculo único de cable, no hay finales terminados. Las señales viajan alrededor del bucle en una dirección y pasan a través de cada computadora. No como la topología de bus pasiva, cada computadora actúa como un repetidor para amplificar la señal y enviarla a la siguiente computadora.

Debido a que la señal pasa a través de cada computadora, el fallo de uno de ellos puede impactar en toda la red.

Paso de Testigo

Un método de transmitir datos alrededor de un anillo es llamado paso de testigo. El testigo es pasado desde una computadora a otra hasta que encuentra una que tiene datos para enviar. La computadora que envía modifica el testigo, pone una dirección electrónica en el dato y lo envía alrededor del anillo.

El dato pasa por cada computadora hasta que encuentra uno con una dirección que coincide con la almacenada en el dato.

La computadora receptor devuelve un mensaje al emisor indicando que el dato ha sido recibido. Después de la verificación, el emisor crea un nuevo testigo y lo libera en la red.

Podría parecer que el paso de testigo lleva mucho tiempo, pero actualmente viaja aproximadamente a la velocidad de la luz. Un testigo puede recorrer un anillo de 200 m. de diámetro más o menos en unas diez milésimas de segundo.

HUBS

Un componente de red que está llegando a ser un equipo estándar en más y más redes es el hub. Un hub es el componente central de una topología en estrella.

Hubs Activos

La mayoría de los hubs son activos en tanto que regeneran y retransmiten las señales igual que hace un repetidor. De hecho, dado que los hubs usualmente tienen 8 ó 12 puertas para conectar computadoras de red, son llamados a veces repetidores multipuerta. Los hubs activos requieren alimentación electrónica para funcionar.

Hubs Pasivos

Algunos tipos de hubs son pasivos, por ejemplo, paneles de cableado o bloques agujereados (punchdown blocks). Actúan como puntos de conexión y no amplifican o regeneran la señal; la señal pasa a través del hub. No requieren alimentación eléctrica.

Hubs Híbridos

Hubs avanzados que pueden acomodar diferentes tipos de cables se llaman hubs híbridos.

Una red basada en hubs puede ser expandida conectando más de un hub.

CONSIDERACIONES

Los hubs son versátiles y ofrecen varias ventajas sobre los sistemas que no los usan. En la topología estándar de bus lineal, una rotura en el cable hará caer la red. Con hubs, sin embargo, una rotura en alguno de los cables enganchados al hub afecta solo a ese segmento. El resto de la red permanece funcionando.

Ventajas

Cambiar o expandir sistemas de cableado como se necesite. Simplemente enchufe en otra computadora u otro hub.

El uso de puertas diferentes para acomodar una variedad de tipos de cable.

Monitorización centralizada de la actividad de la red y del tráfico. Muchos hubs activos tienen capacidad de diagnóstico para indicar cuando está trabajando, o no, una conexión.

VARIACIONES EN LAS PRINCIPALES TOPOLOGÍAS

Hoy, muchas topologías que están trabajando, son combinaciones del bus, la estrella, y el anillo.

Bus en Estrella

El bus en estrella es una combinación de las topologías de bus y de estrella. En una topología de bus en estrella hay varias redes con topología en estrella conectadas juntas con troncales de bus líneas.

Si una computadora, cae, no afectará al resto de la red. Las otras serán capaces de comunicarse. Si un hub cae, todas las computadoras en ese hub son incapaces de comunicarse. Si un hub está conectado a otros hubs, esas conexiones también estarán rotas.

Anillo en Estrella

El anillo en estrella, es similar al bus en estrella. Ambas, el anillo en estrella, y el bus en estrella, están centrados en un hub que contiene el anillo actual o el bus. Los hubs en un bus en estrella están conectados por troncales de bus lineales, mientras que los hubs en un anillo en estrella están conectados en un modelo de estrella por el hub principal.

SELECCIÓN DE UNA TOPOLOGÍA

Hay varios factores a considerar cuando se determina que topología cubre las necesidades de una organización. La tabla siguiente muestra algunas guías para seleccionarla:

Bus

Ventajas

Económico uso del cable
El cable es barato y fácil de trabajar
Simple, segura
Fácil de extender.

Desventajas

La red puede caer con tráfico fuerte
Los problemas son difíciles de aislar
La rotura del cable puede afectar a muchos usuarios.

Anillo

Ventajas

Acceso igual para todos las computadoras
Prestaciones uniformes a pesar de la existencia de muchos usuarios.

Desventajas

El fallo de una computadora puede impactar al resto de la red
Problemas difíciles de aislar
La reconfiguración de la red interrumpe las operaciones.

Estrella

Ventajas

Fácil de modificar y añadir nuevas computadoras
Monitorización y manejo centralizado
El fallo de una computadora no afecta al resto de la red.

Desventajas

Si el punto centralizado falla, la red falla.

CONEXIÓN DE LOS COMPONENTES DE UNA RED

PRINCIPALES TIPOS DE CABLES

La inmensa mayoría de las redes de hoy en día están conectadas por algún tipo de malla o cableado que actúa como el medio de transmisión en la red, transportando señales entre computadoras. Hay una variedad de cables que pueden cubrir las necesidades y los distintos tamaños de las redes, desde pequeñas a grandes.

El tema del cableado puede ser confuso, algo importante a considerar es el nombre Belden, el cual es un fabricante de cables, publica un catálogo que lista más de 2200 tipos de cables.

Afortunadamente, solo tres principales grupos de cables conectan la mayoría de las redes.

- Coaxial
- Par Trenzado
  - Par trenzado blindado (STP)
  - Par trenzado sin blindar (UTP)
- Fibra óptica.

COAXIAL

En un momento dado, el cable coaxial fue el cable de red más ampliamente utilizado. Había un par de razones para el amplio uso del coaxial.

- El coaxial era relativamente barato, ligero, flexible y fácil de trabajar con él. Era tan popular que llegó a ser un medio seguro y fácil de soportar en una instalación.
- En la forma más simple, el coaxial consiste en un núcleo hecho de cobre sólido envuelto por un aislamiento, un trenzado de metal escudándolo y una capa exterior.

Una capa de película metálica y otra capa de trenzado de metal escudando, se conoce como un doble aislamiento. Sin embargo, hay disponible un aislamiento de calidad para entornos sujetos a fuertes interferencias. El aislamiento de calidad consiste en dos capas de película metálica y dos capas de malla metálica.

El aislamiento se refiere al entretejido o malla de metal trenzado que rodea algunos tipos de cable. El aislamiento protege los datos transmitidos absorbiendo señales electrónicas dispersas, llamadas “ruido”, para que no entren en el cable y distorsionen los datos.

El núcleo del cable coaxial transporta las señales electrónicas que conforman los datos. Este hilo del núcleo puede ser sólido o trenzado. Si el núcleo es sólido,

usualmente es cobre. El núcleo está envuelto por una capa de aislamiento que le separa de la malla. La malla trenzada actúa como tierra y protege el núcleo de ruido eléctrico y réplicas (Crosstalk). Las réplicas son desbordamientos de señal desde un cable cercano.

El núcleo conductor y la malla deben estar siempre separados el uno del otro. Si se tocan, el cable experimentará un corto, y fluirán ruido o señales dispersas en la malla, en el hilo de cobre. Esto podría destruir los datos.

El cable entero está rodeado por una capa no conductora, usualmente hecha de caucho, teflón o plástico.

El cable coaxial es más resistente a interferencias y atenuación que el cable de par trenzado.

Atenuación es la pérdida de fuerza en la señal, que empieza a ocurrir en cuanto la señal viaja a través del cable de cobre.

El trenzado, es como un manguito protector que puede absorber señales electrónicas dispersas para que no afecten al dato que está siendo enviado por el núcleo interior del cable. Por esta razón el coaxial es una buena elección para largas distancias y para fiabilidad soportando altos ratios de datos con equipo poco sofisticado.

Tipos de cable coaxial

Existen 2 tipos

- Thin (thinnet) (delgado)
- Thick (thicknet) (grueso)

El tipo depende de las necesidades de su red. Thinnet. Es un cable coaxial flexible, de alrededor de 0,25 pulgadas de grueso. Debido a que éste tipo de coaxial es flexible y fácil de trabajar con él, puede ser usado en prácticamente cualquier tipo de instalación de red.

Las redes que usan thinnet tienen el cable directamente conectado a una tarjeta de red.

El cable thinnet puede transportar la señal hasta aproximadamente 185 metros (sobre 607 pies) antes de que la señal empiece a sufrir por atenuación. Los fabricantes de cables han convenido en ciertas denominaciones para los diferentes tipos. El Thinnet está incluido en un grupo denominado familia RG-58 y tiene una impedancia de 50 ohmios. La impedancia es la resistencia, medida en ohmios, para alternar la corriente en un cable. La principal diferencia en la familia RG-58 es el núcleo central de cobre. Puede ser sólido o trenzado de cobre.

Núcleo trenzado RG-58 A/U
Núcleo sólido RG-58 /U
RG-58 c/u Especificación militar del RG-58 A/U
RG-59Transmisión en ancho de banda como TV
RG-6 Largo en diametro y prohibido para altas frecuencias como RG-59, pero también usado para transmisiones de ancho de banda.

RG-62 Redes ARC Net. Thicknet.- Es un cable relativamente rígido coaxial de alrededor de 0,5 pulgadas de diámetro. Es conocido como Ethernet Standard porque fue el primer tipo de cable usado con la popular arquitectura de red Ethernet. El núcleo de cobre es más grueso que el núcleo thinnet.

Cuanto más grueso es el núcleo de cobre, más lejos puede transportar señales el cable. Esto hace que el thicknet pueda llevar señales más lejos que el thinnet. Thicknet puede llevar la señal hasta 500 metros (sobre 1640 pies). Por lo tanto,

debido a la capacidad de thicknet de transportar transferencia de datos sobre largas distancias, es a veces usado como un backbone para conectar varias redes pequeñas basadas en thinnet.

Un aparato llamado transceptor ó “transceiver” conecta el coaxial thinnet al cable largo thicknet. El transceptor diseñado para Thicknet Ethernet incluye un conector conocido como tipo vampiro con un conector penetrante para hacer la conexión física al núcleo thicknet.

Este conector penetra a través de la capa de aislamiento y hace contacto directo con el núcleo conductor. La conexión desde el transceptor a la tarjeta de red se hace usando un cable transceptor (drop cable) para conectar a la puerta con el conector AUI (Attachment Unit Interface) en la tarjeta.

Un conector de puerto AUI para thicknet es conocido también como un conector DIX Digital Intel Xerox, o un conector DB-15.

El cable thicknet tiene marcas negras cada 2 metros, que es la distancia mínima a la que se puede poner un transceptor desde otro. Antes se usaba el thicknet como backbone, ahora fibra óptica.

Thinnet versus Thicknet

Como regla general, es más difícil trabajar con el cable thick. El cable thin es flexible, fácil de instalar y relativamente barato. El cable thick no se curva fácilmente y es, por lo tanto, difícil de instalar. Esto es una consideración cuando la instalación hay que hacerla por conductos y espacios estrechos. El cable thick es más caro que el thin, pero lleva la señal más lejos.

El hardware de conexión coaxial

Ambos cables, el thinnet y thicknet usan componentes de conexión conocidos como BNC (British Naval Conector), para hacer las conexiones entre el cable y las computadoras.

Hay varios componentes importantes en la familia BNC, incluyendo los siguientes:

- El conector de cable BNC. Se puede soldar o crimpar (ajuste por presión), al final de un cable.
- El conector T. Este conector junta la tarjeta de red en la computadora al cable de red.
- El conector de barrilete BNC. Es utilizado para unir dos tramos de cable thinnet para hacer un tramo más largo.
- El Terminador BNC. Cierra cada final de cable del bus para absorber las señales dispersas. Sin un terminador, una red en bus no funciona.

Cable Coaxial, códigos de fuego y calidad

El tipo de cable que debe usar depende de donde estarán los cables en su oficina. Los coaxiales son de:

- Polivinilo (PVC)
- Plenum

El polivinilo es un tipo de plástico usado para construir el aislamiento y la camisa del cable en la mayoría de los tipos de cable coaxial. El cable de PVC es flexible y puede ser fácilmente conducido por una oficina. Sin embargo, cuando arde, emite gases venenosos. Un plenum es el pequeño espacio en muchos edificios entre el falso techo y el falso suelo, es usado para que circule el aire frío y caliente. Los códigos de fuego son muy específicos con el tipo de cable que puede atravesar

esta área, debido a que cualquier humo o gas en el plenum puede mezclarse con el aire respirado en el edificio.

El cable Plenum se refiere al coaxial que contiene materiales especiales en su aislamiento y camisa. Esos materiales están certificados para ser resistentes al fuego y producir una mínima cantidad de humo. Puede ser usado en el plenum y en caminos verticales sin conducción. Sin embargo es más caro y menos flexible que el de PVC.

Puntos importantes a tener en cuenta del cable coaxial

- Considere estas cuestiones cuando tome la decisión del tipo de cable a usar.
- Use cable coaxial si necesita
- Un medio que puede transmitir voz, vídeo, y datos.
- Para transmitir datos a más largas distancias que un cable barato puede hacerlo.
- Una tecnología familiar que ofrece una razonable seguridad de datos.

CABLE DE PAR TRENZADO

En su forma más simple, el cable de par trenzado consiste en dos filamentos de hilo de cobre girados uno sobre otro. Hay dos tipos de cable de par trenzado: Par Trenzado no blindado (UTP) y Par Trenzado blindado (STP).

Un número de pares trenzados es agrupado a menudo junto y encerrado en una funda protectora para formar un cable. El actual número de pares en un cable varía. Los giros cancelan el ruido eléctrico desde los pares adyacentes y desde otras fuentes como motores, relés y transformadores.

UTP- Par trenzado no blindado (no aislado)

UTP usando la especificación 10 Base T es el tipo más popular de cable de par trenzado y llega rápidamente a ser el cable de LAN más popular.

La longitud máxima del segmento de cable es de 100 metros (alrededor de 328 pies).

UTP consiste en dos hilos de cobre aislados. Dependiendo de un uso particular, hay especificaciones UTP que gobiernan cuántos pares están permitidos por pie de cable. En Norte América, el cable UTP es el más utilizado para los sistemas telefónicos existentes y está aún instalado en muchos edificios de Oficinas.

UTP está especificado en el estándar EIA/TIA 568. EIA/TIA 568 usó UTP en la creación de estándares que aplicó a una variedad de edificios y situaciones de cableado, asegurando consistencia de productos para los clientes. Esos estándares incluyen cinco categorías de UTP:

Categoría 1. Esta se refiere al cable telefónico UTP tradicional que transporta voz pero no datos. La mayoría del cable telefónico anterior a 1983 era de Categoría 1.

Categoría 2. Esta categoría certifica el cable UTP para transmisión de datos hasta 4. Mbps (megabits por segundo). Consiste en 4 pares trenzados.

Categoría 3. Esta categoría certifica el cable UTP para transmisiones de datos hasta 10 Mbps. Consiste en 4 pares girados con 3 vueltas por pie.

Categoría 4. Esta categoría certifica el cable UTP para transmisiones de datos hasta 16 Mbps. Consiste en 4 pares trenzados.

Categoría 5. Esta categoría certifica el cable UTP para transmisión de datos hasta 100 Mbps. Consiste en 4 pares trenzados de cobre. De los 8 hilos solo se usan 4.

La mayoría de los sistemas telefónicos usan un tipo de UTP. De hecho, una razón por la que UTP es tan popular es porque muchos edificios están pre-cableados para sistemas telefónicos de par trenzado. Como parte de este pre-cableado, extra UTP es a menudo instalado para cumplir las futuras necesidades. Si el cable pre-instalado es de grado suficiente para soportar transmisión de datos, puede ser usado en una red de computadoras.

Se requiere precaución, sin embargo, porque el cable telefónico normal puede no tener los giros y otras características eléctricas requeridas para una transmisión de datos limpia y segura. Un problema potencial con todos los tipos de cables es el crosstalk (Diafonía). Se puede recordar que crosstalk es definido como señales que desde una línea se mezclan con señales de otra línea. UTP es particularmente susceptible al crosstalk. El aislamiento se usa para reducir crosstalk.

STP- Par trenzado aislado

STP usa un canutillo o camisa que envuelve la trenza de cobre que es de alta calidad, y más protectora que la del UTP. STP también usa un recubrimiento entre y alrededor de los pares y el giro interno de los mismos. Esto da a STP un excelente aislamiento para proteger los datos transmitidos de interferencias del exterior.

Esto es lo que hace que STP sea menos susceptible a interferencias eléctricas y soporte más altos ratios de transmisión a más largas distancias que UTP.

Componentes del cable de par trenzado

Hardware de conexión. Par trenzado usa conectores telefónicos RJ-45 para conectarse a una computadora. Es similar al conector telefónico RJ-11. Además

se parecen a primera vista, pero hay diferencias cruciales entre ellos. El RJ-45 es ligeramente más largo, y no cabe en el enchufe del RJ-11. El RJ-45 tiene 8 conexiones de cable, mientras que el RJ-11 solo cuatro.

Están disponibles varios componentes para ayudar a organizar grandes instalaciones UTP y hacer más fácil trabajar con el:

- Anaqueles de distribución (racks) y los racks en sí mismos.
- Los racks de distribución y los racks en sí mismos pueden crear más espacio para cables allí donde no hay mucho.
- Es una buena forma para centralizar y organizar una red que tenga un montón de conexiones.
- Paneles de expansión (Patch Panels)
- Hay varias versiones que soportan hasta 96 puntos y velocidades de transmisión de 100 Mbps.
- Latiguillos o ladrones. Conectores RJ-45 simples o dobles para los patch paneles o rosetas de pared y soportan ratios de 100 Mbps.
- Rosetas de pared. Soportan dos o más pares.

Puntos importantes a tener en cuenta en el Par Trenzado

Utilice cable de par trenzado si su red está constreñida por el presupuesto; si quiere una instalación relativamente fácil donde las conexiones de la computadora sean simples.

No use cable de par trenzado si debe estar absolutamente seguro de la integridad de los datos transmitidos a grandes distancias a altas velocidades.

CABLE DE FIBRA ÓPTICA

En este cable, las fibras ópticas transportan señales de datos digitales en forma de pulsos modulados de luz. Es una forma relativamente segura de enviar datos ya que no se envían impulsos eléctricos por el cable de fibra óptica. Esto hace que el cable de fibra óptica no pueda ser derivado y los datos robados, lo que es posible con cualquier cable basado en cobre transportando datos en forma de señales electrónicas.

El cable de fibra óptica es bueno para muy alta velocidad. Tiene alta capacidad de transmisión de datos debido a la ausencia de atenuación y la pureza de la señal.

Composición de la Fibra Óptica

La fibra óptica consiste en un cilindro de vidrio extremadamente fino, llamado núcleo, envuelto por una capa concéntrica de vidrio conocida como el “vestido”. Las fibras están hechas a veces de plástico. El plástico es fácil de instalar, pero no puede llevar los pulsos de luz tan lejos como el vidrio.

Cada filamento de vidrio pasa señales en una única dirección, por eso el cable consiste en dos filamentos en camisas separadas. Uno transmite y el otro recibe. Una capa de plástico reforzado envuelve cada filamento de vidrio mientras que fibras de Kevlar proporcionan resistencia. Las fibras de Kevlar en el conector de fibra óptica están situadas entre los dos cables, que están encapsulados en plástico.

Las transmisiones en cable de fibra óptica no están sujetas a interferencia eléctrica y son extremadamente rápidas (actualmente alrededor de 100 Mbps con ratios demostrados de hasta 200.000 Mbps).

Puede transportar la señal, el pulso de luz, a millas. La luz puede viajar en monomodo y multimodo, rebotando por las paredes del filamento.

Puntos importantes a tener en cuenta en el cable de Fibra Óptica.

- Use cable de fibra óptica, si necesita transmitir a muy altas velocidades sobre largas distancias en un medio muy seguro.
- No use cable de fibra óptica, si está bajo un presupuesto apretado; si no tiene experiencia disponible para instalar adecuadamente y conectar aparatos a él.

TRANSMISION DE LA SEÑAL

Se pueden usar dos técnicas para transmitir las señales codificadas por el cable, banda base (base band) y banda ancha (broad band).

Transmisión en Banda Base

Los sistemas de Banda Base usan señales digitales sobre una frecuencia simple. Las señales fluyen en forma de discretos pulsos de electricidad o luz. Con transmisión en banda base, la total capacidad de comunicación del canal se usa para transmitir una simple señal de datos.

La señal digital utiliza el completo ancho de banda del cable, que constituye un único canal.

Un total ancho de banda en el cable es la diferencia entre las frecuencias más altas y más bajas que son transportadas por el cable.

Cada dispositivo en una red de banda base transmite bidireccionalmente, y algunos pueden transmitir y recibir a la vez. Cuando la señal viaja a lo largo del

cable de red, gradualmente decrece en fuerza y puede distorsionarse. Si la longitud del cable es demasiada, el resultado es una señal que está distorsionada. La señal recibida puede ser irreconocible o mal interpretada.

Como una protección, los sistemas en banda base a veces usan repetidores para recibir una señal y retransmitirla con su fuerza y definición originales para incrementar la longitud practica del cable.

Banda Base: Digital, transmite en los dos sentidos, se ocupa todo el ancho de banda, a veces se usan repetidores.

Transmisión en Banda Ancha

Los sistemas en banda ancha usan señales analógicas y un rango de frecuencias. Con transmisión analógica, las señales son continuas y no discretas.

Las señales fluyen a través del medio físico en forma de ondas electromagnéticas u ópticas.

Con transmisión en banda ancha, el flujo de la señal es unidireccional. Si hay suficiente ancho de banda total, múltiples sistemas de transmisión analógica, como televisión por cable y transmisiones de red, pueden ser soportados simultáneamente en el mismo cable.

Cada sistema de transmisión está alojado en una parte del total de ancho de banda. Todos los dispositivos asociados con un sistema de transmisión, como todas las computadoras usando un cable de LAN, deben estar sintonizados para que usen sólo las frecuencias que están dentro del rango alojado.

Mientras los sistemas de banda base usan repetidores, los sistemas de banda ancha usan amplificadores para regenerar las señales analógicas a su fuerza original.

Debido a que el flujo de transmisión de la señal en banda ancha es unidireccional, debe haber dos caminos para el flujo de datos para que una señal alcance a todos los dispositivos.

Hay dos formas comunes de hacer esto:

Mid-Split (partir por la mitad) divide el ancho de banda en dos canales, cada uno usando una frecuencia diferente o un rango de frecuencias. Un canal se usa para transmitir señales, el otro para recibir.

En la configuración de doble cable, cada dispositivo está enganchado a dos cables. Un cable se usa para enviar y el otro para recibir.

Banda Ancha: Analógica, transmite en un sentido, se puede multiplexar, se usan amplificadores.

El sistema de cableado IBM

IBM desarrolló su propio sistema de cableado, completándolo con sus propios números, estándares, especificaciones y designaciones. Muchos de esos parámetros, sin embargo, son similares a las especificaciones no IBM. El sistema de cableado IBM fue introducido en 1984 para definir estos componentes:

- Conectores de cable
- Face plates (enchufes)
- Paneles de distribución
- Tipos de cable.

El componente del cableado IBM que es único es el conector IBM. El conector IBM es diferente a los BNC estándar y otros, debido a que no es macho ni hembra, sino hermafrodita.

El sistema IBM clasifica el cable por tipos. Por ejemplo, Categoría 3 (grado de voz UTP) es equivalente al tipo 3 IBM. Las definiciones de cable especifican, cuál debería ser el apropiado para una aplicación dada o entorno.

AWG. - El estándar en medición de cable.

En mediciones de cable, a menudo se ve la palabra calibre seguida por las iniciales AWG.

AWG es un sistema de medición para cable que especifica su espesor. Cuando el espesor del cable aumenta, el número AWG disminuye.

El hilo de teléfono se usa a menudo como un punto de referencia. Tiene un espesor de 22 AWG. Un hilo de 14 AWG debería ser más grueso que el telefónico y un 26 AWG más delgado.

Tipos de cable IBM

- Tipo 1. Par trenzado aislado (STP). Dos pares de hilos 22 AWG, envueltos por una capa exterior trenzada. Usado para computadoras y MAUs.
- Tipo 2. Cable de voz y datos. Cable protegido de voz y datos con dos pares trenzados de hilos 22 AWG para datos, una capa exterior trenzada y cuatro pares trenzados de 26 AWG para voz.
- Tipo 3. Cable de voz. Consiste en cuatro pares trenzados sólidos y no protegidos de cables 22 o 24 AWG.

- Tipo 4. No definido.
- Tipo 5. Fibra óptica. Dos fibras ópticas de 62,5/125 micrón multimodo. El estándar de la industria.
- Tipo 6. Cable Data Patch. Dos pares girados de 26 AWG y trenzados con una doble hoja y capa trenzada.
- Tipo 7. No definido.
- Tipo 8. Cable de alfombra. Metido en una camisa plana para utilizar bajo la moqueta. Dos pares trenzados protegidos de 26 AWG.
- Tipo 9. Plenum. Resistente al fuego. Dos pares trenzados protegidos.

Seleccionando el Cableado

Para determinar qué cableado es el mejor, para un lugar en particular, necesita contestar a las siguientes preguntas:

- Cuán pesado será el tráfico en la red.
- Cuáles son las necesidades de seguridad de la red.
- Cuáles son las distancias que debe recorrer el cable.
- Cuáles son las opciones de cable.
- Cuál es el presupuesto del cableado.

La mayoría de los cables protegen contra el ruido eléctrico interno y externo, aunque sea muy lejos y muy rápido, podrán transportar una señal limpia. Sin embargo, cuanta mayor velocidad, claridad y seguridad, más caro será.

Puntos importantes a tener en cuenta en el cableado IBM

Como con muchos componentes de red, hay ofertas o descuentos con el tipo de cable que puede comprar. Si trabaja para una gran organización y escoge el cable más barato, los clientes estarán contentos al principio, pero pronto tendrá noticias

de que la red no es adecuada en la velocidad de transmisión y en seguridad de datos.

El cableado depende de las necesidades de cada lugar en particular. El cable que compre para instalar una LAN en un negocio pequeño tiene requerimientos diferentes que los de una organización grande.

Algunas de las consideraciones que afectan al precio del cable y prestaciones, incluyen:

Logística de instalación

¿Cuán fácil es instalar el cable y con qué trabajo? En una instalación pequeña donde las distancias son cortas y la seguridad no es un tema principal, no tiene sentido el escoger cable grueso, incómodo y caro.

Aislamiento

El nivel de aislamiento requerido puede ser un costo añadido. Casi todas las redes deberían estar usando algún tipo de cable protegido. Cuanto más “ruidosa” es el área en la que está el cable, más aislamiento se requerirá. El cable Plenum es más caro y mejor.

Crosstalk

El Crosstalk y el ruido pueden causar serios problemas en redes grandes donde la seguridad de datos es crucial. El cable barato tiene poca resistencia a los campos eléctricos externos generados por líneas de corriente, motores, relés y transmisores de radio. Esto los hace susceptibles al ruido y al crosstalk.

Velocidad de Transmisión (parte del ancho de banda)

Los ratios de transmisión son medidos en megabits por segundo (Mbps). Un punto de referencia estándar para las actuales transmisiones en LAN sobre hilo de cobre es 10 Mbps, sin embargo, recientes estándares permiten ahora 100 Mbps.

El cable grueso (thick) transmite datos a más larga distancia que el cable fino. Pero el cable grueso, como el thicknet, es más difícil de trabajar que cables más delgados como el thinnet.

El cable de fibra óptica transmite a más de 100 Mbps, puede ser más rápido que el de cobre, pero requiere conocimientos para instalar y es relativamente caro.

Costo

El mejor cable, que transmite datos de forma segura sobre largas distancias, es más caro que el cable fino, que es fácil de instalar y de trabajar con él.

Atenuación

Atenuación es la razón por la que las especificaciones de cable recomiendan ciertos límites de longitud en diferentes tipos de cables.

Si una señal sufre demasiada atenuación, puede no ser comprendida por la computadora receptor. La mayoría de las redes tienen chequeo de errores que generan una retransmisión si la señal es demasiado débil para ser entendida, pero las retransmisiones llevan tiempo y ralentizan la red.

TABLA COMPARATIVA DE LOS DIFERENTES TIPOS DE CABLES

	THINNET COAXIAL 10 BASE 2	THICKNET COAXIAL	PAR TRENZADO TWISTED-PAIR	FIBRA ÓPTICA

		10 BASE 5	10 BASE T	
COSTO DEL CABLE	MÁS QUE EL PAR TRENZADO	MÁS QUE EL THINNET	MÁS BARATO	MÁS CARO
LONGITUD DEL CABLE UTILIZADO	185 m. o 607 pies	500 m. o 1640 pies	100 m. 328 pies	2 km. o 6562 pies
RATIOS DE TRANSMISIÓN	10 Mbps.	10 Mbps.	10 Mbps. 4-100 Mbps.	100 Mbps. o más
FLEXIBILIDAD	RAZONABLEMENTE FLEXIBLE	EL MENOS FLEXIBLE	EL MÁS FLEXIBLE	NO FLEXIBLE
FACILIDAD DE INSTALACIÓN	FÁCIL	FÁCIL	MUY FÁCIL, POSIBLEMENTE YA ESTÉ INSTALADO	DIFÍCIL DE INSTALAR
SUSCEPTIBILIDAD A INTERFERENCIAS	BUENA RESISTENCIA	BUNA RESISTENCIA	SUSCEPTIBLE	SUSCEPTIBLE
PRESENTACIONES ESPECIALES	COMPONENTES MÁS BARATOS QUE EL PAR TRENZADO	COMPONENTES MÁS BARATAS QUE EL PAR TRENZADO	HILO TELEFÓNICO. A MENUDO INSTALADO EN EDIFICIOS.	NO SOPORTA VOZ, DATOS Y VIDEO.
USOS PREFERIDOS	INSTALACIONES MEDIAS/GRANDES CON ALTAS NECESIDADES DE SEGURIDAD		UTP-PEQUEÑOS LUGARES EN PRESUPUESTO. STP-TOKEN_RING EN CUALQUIER TAMAÑO	INSTALACIONES DE CUALQUIER TAMAÑO, CON ALTA VELOCIDAD, ALTA SEGURIDAD E INTEGRIDAD

COMUNICACIONES DE RED SIN CABLES

El entorno sin cables está emergiendo como una opción de redes, cuando la tecnología madure, los vendedores ofrecerán más productos a precios atractivos, que incrementaran las ventas y la demanda. Cuando la demanda crezca, el entorno sin cables crecerá y mejorará.

La frase entorno sin cableado es engañosa porque implica una red completamente libre de cables. En la mayoría de casos esto no es verdad. Las redes sin cables consisten actualmente en componentes sin hilos comunicándose en una red que usa cables, en una red con mezcla de componentes, llamada híbrido.

Mejoras por la ausencia de cables

La idea de redes sin cables atrae la atención porque sus componentes pueden:

- Proporcionar conexiones temporales a una red existente, con cables.
- Ayudar a proporcionar backup a una red existente.
- Proporcionan un cierto grado de portabilidad.
- Extender la red más allá de los límites del cobre o incluso de los cables de fibra óptica.

Usos para las redes sin cables

La dificultad de implementar cable es un factor que continua empujando a los entornos sin cable hacia una mayor aceptación. Son especialmente útiles para las redes:

- Áreas muy concurridas como vestíbulos y recepciones.
- Gente que esté constantemente en movimiento como médicos y enfermeras en un hospital.
- Áreas aisladas en edificios.
- Departamentos donde los cambios físicos son frecuentes.
- Estructuras, como edificios históricos, donde cablear es difícil.

TIPOS DE REDES SIN CABLE

Pueden ser divididas en 3 categorías basadas en su tecnología:

- Redes de área local
- Redes de área extensa
- Computadoras móviles.

La principal diferencia entre esas categorías son las facilidades de transmisión.

Las redes sin cables LAN y LANs extensas usan transmisores y receptores propiedad de la compañía en la que la red opera. Las computadoras móviles usan proveedores públicos como AT &T, MCI, SPRINT PRODYGY y las compañías telefónicas locales y sus servicios públicos, para transmitir y recibir señales.

REDES DE ÁREA LOCAL

Una red inalámbrica parece y actúa como una red cableada, excepto por el medio por el cual se transmite, requiere de una tarjeta de red inalámbrica, con un transceptor, que se instala en cada computadora, y los usuarios se comunican con la red como si estuvieran en computadoras con cable.

Puntos de Acceso

El transceptor (transceiver), a veces llamado punto de acceso, difunde y recibe señales desde las computadoras y a las computadoras de alrededor y pasa los datos de acá para allá entre las computadoras sin cable y la red cableada.

Estas LANs sin cables usan pequeños transceptores colocados en las paredes para conectarse a la red cableada. Los transceptores establecen radio contacto con dispositivos de red portátiles.

Esto no es una verdadera LAN sin hilos porque usa un transceiver puesto en la pared para conectarse a una LAN estándar cableada.

Técnicas de transmisión

Las LAN sin hilos usan cuatro técnicas para transmitir datos:

- Infrarrojos
- Láser
- Banda estrecha de radio (frecuencia única)
- Amplio espectro de radio.

Infrarrojos

Todas las redes sin hilos por infrarrojos operan usando un rayo de luz infrarroja para transportar los datos entre dispositivos. Estos sistemas necesitan generar señales muy fuertes, debido a que las señales de transmisión dispersas son susceptibles a la luz desde fuentes como ventanas.

Este método puede transmitir señales en altos ratios debido al alto ancho de banda de la luz infrarroja. Una red de infrarrojos puede emitir a 10 Mbps.

Hay 4 tipos de redes de infrarrojos:

- Redes en línea de vista (Line-of-sight). Como implica su nombre, esta versión transmite solo si el transmisor y el receptor se ven limpiamente.
- Redes por dispersión de infrarrojos. (Scatter). Esta tecnología emite transmisiones para que reboten en las paredes y techos y eventualmente contacten con el receptor. Tiene un área efectiva de unos 100 pies y tiene una señal lenta para el rebote.
- Redes por reflexión. (Reflective). En esta versión de redes por infrarrojos, los tranceptores ópticos situados cerca de las computadoras transmiten hacia un punto común que redirige las transmisiones a la computadora apropiada.

- Telepunto óptico de banda ancha. Esta versión proporciona servicios de banda ancha. Esta red sin hilos es capaz de manejar requerimientos de alta calidad multimedia que pueden coincidir con los proporcionados por una red de cable.

Mientras la velocidad de los infrarrojos y su conveniencia están generando interés, el infrarrojo tiene dificultad transmitiendo a distancias más largas de 100 pies. Está sujeto también a interferencias por la fuerte luz ambiental que se encuentra en muchos entornos de trabajo.

Láser

La tecnología láser es similar a la de infrarrojos en que requiere una línea directa de visión y una persona o cosa que rompa el láser puede bloquear la transmisión.

Espectro sencillo de radio

Es similar a transmitir desde una emisora de radio. El usuario sintoniza el emisor y el transmisor a una cierta frecuencia. Esto no requiere una línea de visión porque el rango de difusión es 5000. Sin embargo, debido a que la señal es de alta frecuencia, no puede traspasar acero o paredes gruesas.

Los clientes se suscriben a este método desde un servicio proporcionado por Motorola.

Es relativamente lento, en un rango de 4,8 Mbps.

Radio de amplio espectro

La radio de amplio espectro emite señales en un rango de frecuencias. Esto ayuda a evitar los problemas de comunicación de espectro sencillo.

Las frecuencias disponibles están divididas en canales. Los adaptadores de amplio espectro sintonizan en un canal especifico por una determinada longitud de tiempo y entonces cambian a un canal diferente.

Una secuencia de saltos determina el “timing”.

Las computadoras en la red están todas sincronizadas al salto de tiempo.

Para evitar que usuarios no autorizados escuchen la transmisión, el emisor y el transmisor utilizan un código.

La típica velocidad de 250 Kbps hace este método mucho más lento que los otros. Sin embargo, algunas implementaciones pueden ofrecer velocidades de 2 Mbps. Sobre distancias de 2 millas al exterior y 400 pies en interior.

Esta es un área donde la tecnología actualmente proporciona una verdadera red sin hilos.

Por ejemplo, 2 ó más computadoras equipadas con tarjetas Xircom Credit Card Netware y un sistema operativo como Windows 95 o Windows NT pueden actuar como una red peer- to-peer sin cables que los conecten. Sin embargo, si tienes una red existente basada en servidor NT puede enlazar la red de más arriba en ésta añadiendo un Netware Access Point a uno de las computadoras en la red de NT.

Transmisión Punto a Punto

Este método de comunicación de datos no entra en las presentes definiciones de redes. Usa una tecnología punto a punto que transfiere datos desde una computadora a otra de forma opuesta a la comunicación entre varias

computadoras y periféricos. Sin embargo están disponibles componentes adicionales como transceptores “single” y “host”.

Estos pueden ser implementados en cualquier computadora aislada u computadoras que ya están en red para formar una red con transferencia de datos sin hilos. Esta tecnología incluye transferencia de datos serial sin hilos que:

- Usa un enlace de radio punto a punto, para transmisión de datos rápida, y libre de error.
- Penetra paredes, techos y suelos.
- Soporta ratios desde 1,2 a 38,4 Kbps. Y hasta 200 pies en interiores o un tercio de milla con transmisión por “Line-of-site”.

Este tipo de sistema puede transferir datos entre computadoras y entre otros dispositivos como impresoras o lectores de código de barra.

Redes de área local extensa

Otros tipos de componentes sin cables son capaces de hacer trabajos en las LAN extensas de forma similar a sus paralelos con cable.

- Bridges (puentes) para LAN sin cable, por ejemplo, pueden conectar redes alejadas hasta 3 millas.
- Conectividad Multipunto sin hilos.
- Un componente llamado bridge sin hilos ofrece una forma fácil de conectar edificios sin usar cables.

Así como un puente para personas proporciona un camino entre los puntos, un puente “bridge” sin cables provee un camino para datos entre edificios.

El AIRLAN/BRIDGE PLUS, por ejemplo, usa tecnología de radio de amplio espectro para crear un backbone sin hilos para enlazar lugares sobre distancias más allá de la extensión de las LAN. Dependiendo de las condiciones, esto puede ser hasta 3 millas.

El costo de ése componente puede estar justificado debido a que elimina el gasto de líneas alquiladas.

El Bridge sin hilos de gran alcance

Si el Bridge sin hilos no alcanza lo suficiente, una organización podría considerar un bridge sin hilos de gran alcance. Estos también usan tecnología de radio de amplio espectro para proporcionar puenteo Ethernet y Token Ring hasta 25 millas.

Como con el puente original sin hilos, el costo del de gran alcance puede estar justificado por la eliminación de líneas T1 ó conexiones por microondas.

T1 es el servicio estándar de línea digital y proporciona ratios de transmisión de 1,544 Mbps. Este tipo de conexión transporta voz y datos.

COMPUTACIÓN MÓVIL

Las redes sin hilos con móviles implican a compañías telefónicas y servicios públicos para transmitir y recibir señales usando:

- Comunicación por paquetes de radio
- Redes celulares
- Estaciones en satélites.

Los empleados que viajan pueden usar esta tecnología con computadoras portátiles o PDA para intercambiar e-mail, archivos u otra información. Mientras ésta forma de comunicación ofrece ventajas, es lenta. Los ratios de transmisión van de 8 Kbps a 19,2 Kbps. Los ratios pueden ser incluso más lentos si se incluye corrección de errores.

La computación móvil incorpora adaptadores sin hilos que usan la tecnología celular telefónica para conectar computadoras portátiles con la red cableada. Las portátiles usan pequeñas antenas para comunicarse con antenas de radio en el área cercana.

Los satélites en órbita cercana a la tierra recogen las señales de baja intensidad desde los portátiles y componentes móviles de red.

Comunicación por paquetes de radio

Esta técnica rompe una transmisión en paquetes, similar a otros paquetes de red, que incluye:

- La dirección de origen
- La dirección de destino
- Información de corrección de errores.

Los paquetes son enviados a un satélite que los difunde. Solo los componentes con la dirección correcta pueden recibir los paquetes transmitidos.

Redes celulares

Paquete de datos celular digital (CDPD) usa la misma tecnología y alguno de los mismos sistemas de los teléfonos celulares. Ofrece transmisiones de datos de computadora sobre las redes analógicas de voz existentes entre llamadas de voz

cuando el sistema no está ocupado. Es una tecnología muy rápida que solo sufre retrasos inferiores al segundo, que hace que se suavice bastante en transmisiones en tiempo real.

Como en otras redes sin hilos, debe haber un enlace en una red cableada existente.

Estaciones en Satélites

Los sistemas de microondas son buenos para interconectar edificios en sistemas pequeños y cercanos como los que hay en un campus o un parque industrial.

Las microondas son el método más ampliamente usado en larga distancia en los Estados Unidos.

Es excelente para comunicación entre dos líneas de puntos a la vista como:

- Enlaces de satélite a tierra
- Entre dos edificios
- A lo largo de áreas grandes y llanas como lagos o desiertos.

Un sistema de microondas consiste en:

- Dos transceptores de radio, uno para generar (estación de transmisión) y otro para recibir (estación receptora) la transmisión.
- Dos antenas direccionales apuntando la una a la otra para implementar comunicación de las señales transmitidas por los transceptores. Estas antenas están instaladas a menudo en torres para darles más cobertura y elevarlas sobre cualquier cosa que pudiera bloquear sus señales.

ASIGNACIÓN DE DIRECCIONES Y ENRUTAMIENTO

Muchas de las decisiones que se necesitan para la configuración de una red IP depende del enrutamiento. En general, un datagrama IP pasa a través de numerosas redes mientras se desplaza entre el origen y el destino. Veamos un ejemplo típico:

Red 1
128.6.4
Red 2
128.6.21
Red 3
128.121
128 .6 .4.2 128 .6 .4.3 128 .6 .4.1 128 .6 .21.1 128 .121.50 .2
ordenador A
ordenador B
128 .6 .21.2 128 .121.50 .1
ordenador C
gateway R
gateway S

Este gráfico muestra tres computadoras, 2 gateways y tres redes. Las redes pueden ser Ethernet, Token Ring o de cualquier otro tipo. La red 2 podría ser una línea punto a punto que conecta los gateways R y S.

La computadora A puede enviar datagramas a la B directamente, usando la red 1. Sin embargo, no puede llegar a la computadora C directamente, puesto que no están en la misma red. Hay varias maneras de conectar redes. En el gráfico asumimos el uso de gateways (más adelante veremos otras alternativas). En este caso, los datagramas que van desde A a C deben ser enviados a través del

gateway R, red 2 y gateway S. Todas las computadoras que usan TCP/IP necesitan que se les suministre la información y algoritmos apropiados para que puedan saber cuándo un datagrama debe ser enviado a través de un gateway, y elegir el gateway apropiado.

El enrutado está íntimamente relacionado con la asignación de direcciones. Podemos apreciar que la dirección de cada computadora comienza con el número de la red a la que pertenece. Por tanto, 128.6.4.2 y 128.6.4.3 se encuentran en la red 128.6.4. Luego los gateways, cuyo trabajo es conectar dos redes, tienen una dirección de ambas redes. Por ejemplo, el gateway R conecta la red 128.6.4 y 128.6.21. Su conexión a la red 128.6.4 tiene la dirección 128.6.4.1. Su conexión a la red 128.6.21 tiene la dirección 128.6.21.2.

Debido a esta relación entre direcciones y redes, las decisiones de enrutado deben basarse estrictamente en el número de red de destino. La información de enrutamiento de la computadora A tendrá el siguiente aspecto:

red	gateway	métrica
128.6.4	-	0
128.6.21	128.6.4.1	1
128.121	128.6.4.1	2

En esta tabla, la computadora A puede enviar datagramas a las computadoras de la red 128.6.4 directamente, y para los datagramas a las computadoras de las redes 128.6.21 y 128.121 es necesario usar el gateway R. La "métrica" será usada por algún tipo de algoritmo de enrutamiento, como medida de la lejanía del destinatario. En nuestro caso, la métrica simplemente indica cuantos diagramas tiene que atravesar para llegar a su destino (conocida como "cuenta de saltos").

Cuando la computadora A está lista para enviar un datagrama se examina la dirección del destinatario. Comparamos el inicio de dicha dirección de red con las

direcciones de la tabla de enrutamiento. Las distintas entradas de la tabla indican si el datagrama debe ser enviado directamente, o a un gateway.

Un gateway consiste simplemente en una computadora conectada a dos redes diferentes, y está habilitado para enviar datagramas entre ellos. En muchos casos es más eficiente usar un equipo especialmente diseñado para desempeñar el papel de gateway. Sin embargo, es perfectamente posible usar una computadora, siempre y cuando tenga más de un interfaz de red y un software capaz de enviar datagramas.

Un gateway tiene varias direcciones, una para cada red a la que esté conectado. Aquí encontramos una diferencia entre IP y otros protocolos de red: cada interfase de una computadora tiene una dirección. Con otros protocolos, cada computadora tiene una única dirección, aplicable a todos sus interfases. Un gateway entre las redes 128.6.4 y 128.6.21 tendrá una dirección que comience por 128.6.4 (por ejemplo, 128.6.4.1). Esta dirección se refiere a su conexión a la red 128.6.4. También tendrá una dirección que comience con 128.6.21 (por ejemplo, 128.6.21.2). Esta se refiere a su conexión a la red 128.6.21.

El término "red" generalmente se suele identificar a dispositivos del tipo Ethernet, en la cual varias máquinas están conectadas. Sin embargo, también se aplica a líneas punto a punto. En el gráfico anterior, las redes 1 y 3 podrían estar en ciudades distintas; la red 2 podría ser una línea serie, un enlace satélite, u otro tipo de conexión punto a punto. Una línea punto a punto es tratada como una red que consta sólo de dos computadoras. Como cualquier otra red, una línea punto a punto tiene una dirección de red (en este caso, 128.6.21). Los sistemas conectados por la línea (gateways R and S) tienen direcciones en dicha red (en este caso, 128.6.21.1 y 128.6.21.2).

Es posible diseñar software que no necesite distintos números de red para cada línea punto a punto. En este caso, el interfase entre el gateway y la línea punto a

punto no tiene una dirección. Esta solución es apropiada cuando la red es tan grande que peligra el hecho de que nos quedemos sin direcciones. Sin embargo, tales "interfases anónimas" pueden dificultar bastante el manejo de la red. Puesto que no tienen dirección, el software de red no tiene manera de referirse a dicho interfase, y, por tanto, no es posible obtener información sobre el flujo y los errores de la interfase.

ELECCIÓN DE UNA ADECUADA ESTRUCTURA DE DIRECCIONES

Antes de comenzar a montar una estructura de IP, necesitamos uno o más números de red oficiales. Una dirección IP tiene un aspecto como el siguiente: 128.6.4.3. Esta dirección sólo podrá ser usada por una computadora de la Universidad Pedagógica de Durango. La primera parte de dicha dirección, 128.6, es un número de red asignado a dicha Universidad por una autoridad central. Por tanto, antes de asignarse direcciones a nuestras computadoras, deberemos obtener una dirección oficial de red.

Sin embargo, alguna gente configura sus redes usando, o bien, una dirección aleatoria o usando una dirección genérica suministrada por defecto en el equipo. Esta forma de trabajar podría funcionar en pequeñas redes, pero seguramente no lo hará en una mayor. Además, es posible que quisiéramos conectar nuestra red con la red de otra organización. Incluso si nuestra organización tuviese un gran control de seguridad, es posible que tuviéramos una computadora dedicada a la investigación que estuviese conectada a una universidad u otra organización investigadora. Esta universidad o entidad estaría seguramente conectada a una red de nivel nacional. Tan pronto como uno de nuestros datagramas salga de nuestra red local va a provocar un estado de confusión en la organización con la que nos comuniquemos, porque la dirección que aparece en nuestros datagramas está probablemente asignada oficialmente a alguien distinto.

La solución es simple: obtener una dirección propia desde el principio. Además, no cuesta nada. La decisión más importante que tenemos que hacer para configurar una red es, sin lugar a dudas, cómo asignar las direcciones IP a las computadoras. Esta elección debe de hacerse desde el punto de vista de cómo nuestra red puede crecer. Si no se hiciese así, es casi seguro que tendremos que cambiar las direcciones en un futuro. Y cuando tengamos varios cientos de computadoras, cambiar tantas direcciones es casi imposible.

Las direcciones son muy importantes porque los datagramas IP son enrutados en base a dicha dirección. Por ejemplo, las direcciones de la Universidad Pedagógica de Durango (UPD), tienen una estructura de dos niveles. Una dirección típica puede ser 128.6.4.3. La dirección 128.6 es la asignada a dicha Universidad. Visto desde el exterior, 128.6 es una simple red.

Cualquier datagrama enviado desde el exterior, que comience por 128.6, se dirigirá al Gateway más cercano de la Universidad Pedagógica de Durango. Sin embargo, dentro de la UPD dividimos el espacio de direcciones en "subredes". Usamos los siguientes 8 bits de dirección para indicar a qué subred pertenece la computadora. Así, 128.6.4.3 pertenece a la subred 128.6.4. Generalmente, las subredes se corresponden con redes "físicas" o reales, por ejemplo una red Ethernet; sin embargo, veremos algunas excepciones más adelante.

Cuando queremos configurar una red, hay varias decisiones de direccionamiento que debemos afrontar:

* ¿Dividimos nuestro espacio de direcciones?
* Si lo hacemos, ¿usamos subredes o direcciones de clase C?
* ¿Cómo debe ser de grande el espacio de direcciones que necesitamos?

SUBDIVISIÓN DE NUESTRO ESPACIO DE DIRECCIONES

No es absolutamente necesario usar subredes. Hay mecanismos que permiten actuar a un campus o compañía completa como una simple y gran Ethernet, así que no es necesario un enrutamiento interno. Si usamos estas tecnologías, entonces no necesitaremos dividir nuestro espacio de direcciones. En este caso, la única decisión que tenemos que tomar es la de qué clase de dirección debemos de usar. Sin embargo, recomendamos usar un enfoque de subredes o cualquier otro método de subdividir nuestro espacio de dirección en varias redes:

- En secciones posteriores discutiremos que los gateways internos son recomendables para todas las redes, más allá de su simplicidad.
- Incluso si no necesitamos gateways en estos momentos, podemos descubrir que tarde o temprano necesitaremos usarlos. De esta manera, probablemente tiene sentido asignar direcciones como si cada Ethernet o Token Ring fuera una subred separada. Esto permitirá hacer conversiones a subredes reales, si esto es necesario.
- Por razones de mantenimiento, es conveniente tener direcciones cuya estructura corresponda con la estructura de la red. Por ejemplo, si vemos un datagrama extraviado procedente del sistema 128.6.4.3, es de bastante ayuda saber que todas las direcciones que comienzan por 128.6.4 se encuentran en un determinado edificio.

SUBREDES Y MULTIPLES NUMEROS DE RED

Supongamos que estamos convencidos de que es una buena idea imponer alguna estructura en nuestras direcciones. La siguiente cuestión es cuál es la más adecuada. Hay dos enfoques básicos: subredes y múltiples números de red.

Los estándares de Internet especifican el formato de las direcciones. Para las direcciones que comienzan entre 128 y 191 (las más usadas actualmente), los dos

primeros octetos forman el número de red; por ejemplo, en 140.3.50.1, 140.3 es el número de red. Los números de red están asignados a una organización particular. ¿Qué hacemos con los dos siguientes octetos que le siguen?. Podríamos optar por hacer al siguiente octeto un número de subred, u otro esquema completamente distinto. Los gateways dentro de nuestra organización deben configurarse para conocer qué esquema de división de redes estamos usando. Sin embargo, fuera de la organización nadie sabrá si 140.3.50 es una subred y 140.3.51 es otra; simplemente, fuera se sabe que 140.3 es una organización. Desafortunadamente, esta habilidad de añadir una estructura adicional a las direcciones, mediante el uso de subredes, no estaba presente en las especificaciones originales y, por tanto, un software antiguo sería incapaz de trabajar con subredes. Si una parte importante del software que hemos de usar tiene este problema, entonces no podremos dividir nuestra red en subredes.

Algunas organizaciones usan un enfoque distinto. Es posible que una organización use varios números de red. En lugar de dividir un simple número de red, por ejemplo 140.3, en varias subredes, como de 140.3.1 a 140.3.10, podríamos usar 10 números distintos de red. De esta manera haríamos una asignación desde 140.3 hasta 140.12. Todo el software IP sabrá que estas direcciones se corresponden con redes distintas.

A pesar de que usando números de red distintos todo funciona correctamente dentro de la organización, hay dos serias desventajas. La primera, y menos importante, es que se malgasta un gran espacio de direcciones. Hay solamente sobre unas 16.000 posibles direcciones de clase B. No queremos malgastar diez de ellas en nuestra organización, a no ser que sea bastante grande. Esta objección es “menos seria”, porque podríamos pedir una dirección C para este propósito y hay sobre 2 millones de direcciones C.

El problema más serio para usar varias direcciones de red, en lugar de subredes, es que sobrecarga las tablas de enrutamiento en el resto de Internet. Como

comentamos anteriormente, cuando dividimos nuestro número de red en subredes, esta división sólo es conocida dentro de la organización, pero no fuera. Los sistemas externos a la organización sólo necesitan una entrada en sus tablas para ser capaces de llegar. Por tanto, otras Universidades tienen entradas en sus tablas de enrutamiento para 128.6. Si usa un rango de redes en lugar de subredes, dicha división será visible en todo Internet. Si usamos los números 128.6 a 128.16, en lugar de 128.6, las otras universidades necesitarían tener una entrada para cada uno de estos números de red en sus tablas de enrutamiento. La mayoría de los expertos de TCP/IP recomiendan el uso de subredes, en lugar de múltiples redes. La única razón para considerar múltiples redes es el uso de un software que no puede manejar subredes. Esto era un problema hace algunos años, pero actualmente es poco frecuente.

Una última indicación sobre subredes: Las subredes deben ser "adyacentes". Esto significa que no podemos conectar la subred 128.6.4 con la subred 128.6.5 mediante otra red, como la 128.121. Por ejemplo, la Universidad Juárez del Estado de Durango (UJED), tiene su edificio central en el Centro de la Ciudad de Durango, además de ello tiene varias escuelas dentro y fuera de la mancha urbana, además de tener un campus en la ciudad de Gómez Palacio. Es perfectamente posible conectar redes en ciudades distintas que sean subred de 128.6. Sin embargo, en este caso, las líneas entre el edificio central de la UJED y la Ciudad de Gómez Palacio deben ser parte de 128.6. Supongamos que decidimos usar una red regional como la RegionaLnet para comunicarnos entre dos campus, en lugar de usar su propia línea.

Puesto que RegionaLnet tiene de número de red 128.121, los gateways y líneas de serie que usarían empezarían por 128.121. Esto viola las reglas. No está permitido tener gateways o líneas que son parte de 128.121 conectando dos partes de 128.6. Así, si queremos usar RegionaLnet entre nuestros dos campus, tendríamos que obtener diferentes números de red para los dos campus (Esta regla es un resultado de las limitaciones de la tecnología de enrutamiento.

Eventualmente podría desarrollarse un software para un gateway para manejar configuraciones cuyas redes no son contiguas).

COMO ASIGNAR LAS SUBREDES O LOS NUMEROS DE RED

Ahora, una vez decidido si vamos a usar subredes o múltiples números de red, tenemos que asignarlos. Normalmente es bastante fácil. Cada red física, ya sea Ethernet o Token Ring, se le asigna un número distinto de subred. Sin embargo, existen otras opciones.

En algunos casos, puede que tenga sentido asignar varios números de subred a una única red física. En la UJED hay una única Ethernet que ocupa tres edificios, usando repetidores. Está claro que a medida que vayamos añadiendo computadoras a esta Ethernet se irá dividiendo en varias Ethernets separadas. Para evitar tener que cambiar de direcciones cuando esto suceda, hemos asignado tres números de red distintas a esta Ethernet, una por edificio (esto podría ser útil, incluso, si no hubiésemos dividido la Ethernet con el fin de ayudar a localizarlos). Pero, antes de hacer esto, debemos estar muy seguros de que el software de todas las computadoras puede manejar una red que tiene tres números de red.

También hemos de elegir una "máscara de subred", que será usada por el software del sistema para separar la parte de subred del resto de la dirección. Hasta ahora hemos asumido que los dos primeros octetos son el número de red y el siguiente es el número de subred. Para las direcciones de clase B, el estándar especifica que los dos primeros octetos pertenecen al número de red. Y, por otro lado, tenemos libertad para establecer el límite del número de subred y el resto de la dirección. Es bastante usual utilizar un octeto de número de subred, pero no es la única posibilidad. Veamos de nuevo esta dirección de clase B, 128.6.4.3. Es fácil deducir que, si el tercer octeto es usado como número de subred, entonces habrá 256 posibles subredes y, en cada subred, habrá 256 posibles direcciones

(en realidad es más acertado decir que disponemos de 254, ya que no es buena idea usar 0 ó 255 como números de subred o dirección). Supongamos que sabemos que nunca vamos a tener más de 128 computadoras por subred, pero es probable que necesitemos más de 256 subredes (por ejemplo, un campus con una gran cantidad de pequeños edificios). En ese caso, podríamos establecer 9 bits como número de red, dejando 7 bits para el direccionamiento de cada subred. Esta decisión queda plasmada en una máscara de bits, usando unos para los bits usados por los números de red y de subred y ceros para los bits usados para el direccionamiento individual. La máscara de red más común es 255.255.255.0. Si elegimos 9 bits para el número de subredes y 7 para las direcciones, la máscara de subred sería 255.255.255.128.

Generalmente, es posible especificar la máscara de subred como parte de la configuración del software IP. Los protocolos IP también permiten a las computadoras que envíen un mensaje preguntando cuál es su máscara de subred. Si nuestra red soporta el envío de estos mensajes, y hay, al menos, una computadora o gateway de la red que conoce dicha máscara de subred, posiblemente será innecesario especificarlo en cada una de las restantes computadoras. Pero esta posibilidad puede traer muchos problemas. En caso de que nuestra implementación TCP/IP diera una máscara de subred errónea, se causaría una mala configuración en toda la red. Por lo tanto, es más seguro poner cada máscara de subred explícitamente en cada sistema.

TRABAJAR CON MÚLTIPLES SUBREDES "VIRTUALES" EN UNA RED

La mayoría del software está desarrollado bajo el supuesto de que cada red local tiene el mismo número de subred. Cuando existe un flujo hacia una máquina con un distinto número de subred, el software espera encontrar un gateway que pueda dirigirlo hacia esa subred.

Veamos detalladamente qué ocurre en este caso. Supongamos que tenemos las subredes 128.6.19 y 128.6.20 en la misma Ethernet. Consideremos las cosas que ocurren desde el punto de vista de una computadora con dirección 128.6.19.3. Dicha computadora no tendrá problemas para comunicarse con las máquinas de dirección 128.6.19.x. Estas máquinas están en la misma subred, y nuestra computadora simplemente deberá enviar los datagramas al 128.6.20.2. Puesto que esta dirección indica que está en una subred distinta, la mayoría del software esperará encontrar un gateway que haga de puente entre ambas subredes. Por supuesto, no existe un gateway entre las "subredes" 128.6.19 y 128.6.20, puesto que están en la misma Ethernet. De aquí se deduce que tenemos que encontrar una manera de indicarle al software que el 128.6.20 se encuentra realmente en la misma Ethernet.

La mayoría de las implementaciones TCP/IP pueden manejar más de una subred en la misma red. Por ejemplo, el Berkeley Unix nos permite hacerlo usando una ligera modificación del comando usado para definir gateways. Si, por ejemplo, queremos que para pasar de la subred 128.6.19 a la subred 128.6.4 se use el gateway con dirección 128.6.19.1, podemos usar el comando route add 128.6.4.0 128.6.19.1 1.

Esto indica que para llegar a la subred 128.6.4 el flujo debe ser enviado a través del Gateway 128.6.19.1. El "1" se refiere a la "métrica de enrutamiento". Si usamos la métrica "0", estamos diciendo que la subred de destino está en la misma red y, por consiguiente, no se necesita ningún gateway. En nuestro ejemplo, deberemos usar en el sistema 128.6.19.3 route add 128.6.20.0 128.6.19.1 0

La dirección usada en el lugar de 128.6.19.1 es irrelevante. La métrica "0" nos informa de que no va a usarse ningún gateway, luego no se usará dicha dirección. Sin embargo, deberá ampliarse una dirección legal de la red local.

Otra forma de trabajar con múltiples subredes

Hay otro modo de manejar varias subredes sobre una red física. Este método supone la desconfiguración de nuestros anfitriones o hosts y, por ello, es potencialmente peligrosa, si no sabemos exactamente lo que estamos haciendo. Sin embargo, puede resultar más cómodo cuando trabajamos con una gran cantidad de subredes en una red física. Un ejemplo de este tipo sería una instalación que use bridges, y use subredes simplemente por facilidades de administración. El truco está en configurar el software de nuestros hosts como si no usasen subredes. Así, nuestros hosts no harán ninguna distinción entre las distintas subredes y, por tanto, no habrá problemas para trabajar con todas ellas. Ahora, el único problema es cómo comunicarnos con subredes que no estén en esta red de múltiples subredes. Pero, si nuestros gateways manejan proxy ARP, ellos resolverán este problema por nosotros. Este enfoque está especialmente indicado cuando la misma red contiene múltiples subredes y, particularmente, si se van a añadir algunas más en un futuro. Desgraciadamente, tiene dos problemas:

- Si tenemos hosts con múltiples interfases, deberemos ser muy cuidadosos. En primer lugar, sólo debería haber máquinas con un interfase en la red con múltiples subredes. Por ejemplo, supongamos que disponemos de una red que consta de varias Ethernets conectadas mediante bridges; no podemos tener una máquina con interfases en dos de estas Ethernets, pero podemos tener un sistema con un interfase en esta red de subredes múltiples y otra en otra subred apartada de ésta. En segundo lugar, cualquier máquina con múltiples interfases deberá conocer la verdadera máscara de subred, y necesitará estar informada explícitamente de cuáles de las subredes están en la red de múltiples subredes.

Estas restricciones son consecuencia de que un sistema con múltiples interfases tiene que conocer qué interfase ha de usar en cada caso.

- También deberemos prestar atención a la facilidad ICMP de la máscara de subredes. Esta facilidad permite a los sistemas emitir una consulta para conocer cuál es la máscara de subred. Si la mayoría de los hosts piensan que la red no está dispuesta en subredes, pero los gateways y los hosts con varias interfases piensan lo contrario, tenemos aquí un foco potencial de confusión. Si un gateway o hosts con varios interfases envía una réplica a una ICMP de máscara de red, dando la verdadera máscara de subred, alguno de los restantes hosts puede interceptarlo. La situación contraria también sería posible. Esto significa que tendremos que:

* deshabilitar las réplicas a las ICMP de máscara de subred en todos aquellos sistemas que conocen la máscara real de subred (esto es especialmente fácil si solamente los gateways lo conocen);
* asegurar que nuestros hosts ignoran las réplicas ICMP.

A medida que establecemos una máscara de subred explícitamente, se supone que los hosts ignoran los ICMP de máscara de subred, así que deberemos ser capaces de establecer diferentes máscaras en diferentes hosts sin causar ningún problema, siempre y cuando podamos establecer la máscara explícitamente en todos ellos. Sin embargo, existen implementaciones IP que cambiarán su máscara de subred cuando vean una réplica de ICMP de máscara de subred.

Múltiples subredes: Consecuencias en el Broadcasting

Cuando tenemos más de una subred en una misma red física, hay que tener cuidado respecto a las direcciones de broadcasting. De acuerdo con los últimos estándares, hay dos formas distintas para que un host de la subred 128.6.20 pueda enviar un broadcast en la red local. Una es usar la dirección 128.6.20.255. La otra es usar la dirección 255.255.255.255. La dirección 128.6.20.255 dice, explícitamente, "todos los hosts de la subred 128.6.20"; la 255.255.255.255 expresa "todos los hosts de mi red local". Normalmente, ambas tienen el mismo

efecto. Pero no lo tienen cuando hay varias subredes en una red física. Si la red 128.6.19 está en la misma red, también recibirá el mensaje enviado a 255.255.255.255. Sin embargo, los hosts con números 128.6.19.x no escucharán los mensajes enviados a 128.6.20.255. El resultado es que ahí tenemos dos tipos distintos de direcciones de broadcast con dos significados distintos. Esto conlleva que debemos tener cuidado configurando el software de red, para asegurarnos de que nuestros broadcasting llegan a donde queremos que lo hagan.

Cuando solicitamos un número oficial de red se nos preguntará qué clase de número de red necesitamos. Las posibles respuestas son A, B y C. La decisión elegida limitará nuestro espacio de direcciones a usar. Las direcciones de clase A ocupan un octeto; las de clase B, dos octetos, y la clase C, tres octetos. Luego, hay más direcciones de clase C que direcciones de clase A, pero las de clase C no pueden tener muchos hosts. La idea que podemos sacar de lo anterior es que debería haber pocas grandes redes, un número moderado de redes de tamaño mediano y bastantes pequeñas redes. En la siguiente tabla observamos dicha distinción:

Clase	Rango	1er. Octeto	red	resto direcciones posibles
A	1 - 126	p	q.r.s	16777214
B	128 - 191	p.q	r.s	65534
C	192 - 223	p.q.r	s	254

Por ejemplo, la red 10 es de la clase A y por tanto tiene direcciones entre 10.0.0.1 y 10.255.255.254. Esto significa 2543, que son sobre unos 16 millones de posibles direcciones (realmente, la red 10 tiene algunas direcciones con octetos a cero, así que habrá algunas direcciones posibles más). La red 192.12.88, una dirección de clase C, tendrá sus hosts entre el 192.12.88.1 y el 192.12.88.254 y, por lo tanto, habrá 254 posibles hosts.

En general, deberemos elegir la clase menor que nos proporcione suficientes direcciones capaces de direccionar nuestra red, con sus posibles futuras ampliaciones. Aquellas organizaciones que usan computadoras en varios edificios, probablemente necesitarán una dirección de clase B, suponiendo que vamos a usar subredes (y si vamos a tratar con distintos números de red, deberíamos solicitar varias direcciones de clase C). Las direcciones de clase A, normalmente, sólo se usan en grandes redes públicas y algunas pocas redes de grandes corporaciones.

En la asignación de Direcciones IP, la autoridad máxima es la IANA (Internet Assigned Number Authority). A escala continental, la IANA delega grandes bloques de direcciones IP a los denominados registros regionales, de los que, de momento, existen tres en el mundo:

- El RIPE NCC (RIPE Network Coordination Center) es el registro delegado de Internet a nivel europeo y se encarga, entre otras tareas, de la asignación de bloques de direcciones IP a los proveedores de servicios Internet en Europa y su área de influencia.
- El AP-NIC lleva a cabo la tarea de asignación de bloques de direcciones IP a los proveedores de la región del Asia-Pacífico.
- El InterNIC se encarga de la asignación de bloques de direcciones IP a los proveedores de Internet en América del Norte y, de momento, en el resto del mundo.

Las organizaciones y usuarios finales han de obtener las direcciones IP necesarias para conectarse a Internet a través de su proveedor de acceso a Internet, quien a su vez las habrá obtenido bien de su proveedor de tránsito, bien del registro regional correspondiente.

LÍNEAS IP Y MICRO GATEWAYS: DIRECCIONES ASIGNADAS DINÁMICAMENTE

En la mayoría de los casos, cada una de las computadoras tendrá su propia dirección IP permanente. No obstante, hay algunas situaciones donde tiene más sentido asignar direcciones dinámicamente. La mayoría de los casos que manejan líneas IP constan de gateways destinados principalmente a microcomputadoras.

Líneas IP

Es posible usar IP sobre líneas telefónicas. Uno de los protocolos para hacer esto es el SLIP ("Serial line IP"). SLIP se usa frecuentemente en, al menos, dos circunstancias distintas:

* Como una alternativa barata a líneas punto a punto permanente, para aquellos casos en los que no está suficientemente justificado una línea dedicada.
* Como una manera de conectar individualmente un PC a una red, cuando se encuentran localizados en edificios que no tienen Ethernets o cualquier otro tipo LAN.

Vamos a usar el término "servidor SLIP" para referirnos a un sistema de computadora(s) que incluye una serie de modems, con los que otros sistemas pueden conectarse usando SLIP. Se trata de un sistema que proporciona un gateway de nuestra red para usuarios de PC, o para otras redes que se conectan usando SLIP.

Si tenemos varios PC's conectados mediante SLIP, muchas veces no es práctico usar una dirección IP propia para cada PC. Una de las razones puede ser que no haya suficientes direcciones. Para que el enrutamiento funcione correctamente, estos sistemas conectados deben tener sus direcciones en la misma subred que

el servidor SLIP. Por lo general, hay solamente del orden de 256 direcciones disponibles en cada subred. Si el número de PC's que pueden conectarse es mayor que esa cifra, no podremos asignarles su propia dirección. Si, además, tenemos servidores SLIP en más de una subred, la asignación permanente de direcciones se hace aún más complicada. Si un usuario es capaz de llamar a dos servidores, su PC necesitaría dos direcciones, una para cada subred.

Para solucionar estos problemas, la mayoría de las implementaciones SLIP asignan las direcciones dinámicamente. Cuando un PC se conecta con el servidor SLIP, el servidor busca una dirección IP que no se esté usando y se la asigna al PC. La forma más simple de manejar esto es dando a cada servidor SLIP un rango de direcciones IP que controle y pueda asignar.

Cuando usamos este esquema, el software SLIP debe comunicar al PC, de alguna manera, qué dirección debe usar. Si cada PC tiene una dirección permanente, tendríamos el problema contrario: cuando un PC se conecta con un servidor debe de haber algún método para que el PC comunique al servidor su dirección. Este problema debe ser estudiado cuidadosamente, porque en otro caso alguien podría usar la dirección de otro y tener acceso a sus archivos.

Desafortunadamente, no hay un estándar para manejar estos problemas de direccionamiento con SLIP. Hay varias implementaciones SLIP que lo hacen, pero todavía no hay un estándar.

Hasta que no se elabore éste, deberemos tener cuidado con el software SLIP. Tenemos que asegurarnos de que dicha asignación de dirección se lleva a cabo de la manera que queremos y que nuestro servidor SLIP y los PC's tienen claro la forma en que se asignan las direcciones.

Recomendamos dar direcciones permanentes a los PC's en aquellos casos en que las demás computadoras tienen que ser capaces de conocer con qué PC

están hablando. Este podría ser el caso de un PC para recibir correo privado, o cualquier otro servicio con transacciones delicadas. Y recomienda el direccionamiento dinámico cuando tenemos un gran número de PC's y las aplicaciones que utilizan para acceder a la red tienen sus propios mecanismos de seguridad.

Cuando usemos SLIP para conectar dos redes, hay que considerar tres elecciones para el manejo de direcciones (teniendo en cuenta que no todo el software SLIP puede controlar los tres apartados):

* Tratar a las conexiones SLIP como si se tratasen de líneas punto a punto que no están disponibles permanentemente. Si podemos conectar con más de una computadora, cada par de computadoras que se comunican tienen un número de red distinto del que ellos usarían cuando se comunican con el otro.
* Usar un software de enrutamiento que permita interfases anónimos. En este caso, no serían necesarias las direcciones.
* Asignar direcciones dinámicamente cuando la conexión está abierta, tan pronto como el PC haya contactado.

Si hacemos sólo una o dos conexiones a otro sistema, es bastante razonable usar un número de red para cada conexión. Este método es fácil de usar y limita los errores estadísticos.

Si tenemos muchas conexiones distintas, probablemente es mejor usar interfases anónimos.

Aunque si los sistemas de enrutamiento no lo soportan, debemos usar asignación dinámica.

Al igual que SLIP, PPP "Point to Point Protocol" es un protocolo serie distinto utilizado para enviar datagramas a través de una conexión serie, pero mejora

algunas de las carencias del anterior. El PPP permite a las partes comunicantes negociar opciones como las direcciones IP y el tamaño máximo de los datagramas al comenzar la conexión, y proporciona permisos de conexión a los clientes (autorizaciones). Para cada una de estas capacidades, el PPP tiene un protocolo concreto.

A continuación, citaremos los elementos básicos que constituyen el PPP. Esta descripción esta muy lejos de ser completa; si quiere saber mas sobre el PPP, lea sus especificaciones en el RFC 1548, asi como en la docena de RFCs que le acompañan.

En la parte más baja del PPP está el protocolo de Control de Conexión de Datos de Alto-Nivel, abreviadamente HDLC (en realidad, el HDLC es un protocolo mucho más general publicado por la Organización Internacional de Estándares, ISO) que define los límites de las tramas PPP individuales, y proporciona un control de errores de 16 bit. Al contrario de lo que ocurría en las encapsulaciones SLIP más antiguas, una trama PPP es capaz de llevar paquetes de otros protocolos distintos al IP, como los IPX de Novell o Appletalk. El PPP consigue esto añadiendo a la trama básica HDLC un campo de control que identifica el tipo de paquete contenido en la trama.

El LCP, Protocolo de Control de Enlace, es utilizado en la parte más alta del HDLC para negociar las opciones concernientes a la conexión de datos, tales como la Unidad Máxima de Recepción (MRU) que establece el tamaño máximo del datagrama que una de las partes de la conexión acepta recibir.

Micro gateways

Es perfectamente posible que un microcomputador forme parte de una red IP. Pero hay una tendencia de que los micros utilicen distintas tecnologías de red que la de los grandes sistemas.

Esto es debido a que muchos de los usuarios de micros empiezan a demandar un software de red diseñado específicamente para las necesidades de un micro, incluso para un particular tipo de micro. Muchos usuarios están interesados en usar TCP/IP sin tener que abandonar su red especial de micro, a la que están acostumbrados. Por esta razón, hay un creciente número de productos, especialmente gateways, que dan acceso a los PC's tanto a redes orientadas a micros como a TCP/IP.

En esta sección vamos a hablar del AppleTalk, de Apple, a modo de ejemplo. No obstante, existen productos similares para otros tipos de redes de micros. Hay que aclarar que el término AppleTalk se asocia a los protocolos de red de Apple, mientras que LocalTalk se asocia a una tecnología específica de par trenzado, en la que AppleTalk fue inicialmente implementada. Por tanto, el AppleTalk es análogo a los protocolos TCP/IP, mientras que LocalTalk es análogo a medio Ethernet.

Algunas compañías ofrecen gateways para conectar una red AppleTalk corriendo sobre LocalTalk, con redes IP corriendo sobre Ethernet. A pesar de que hay varios productos de este tipo, la mayoría de ellos incluyen los siguientes servicios:

* Las aplicaciones TCP/IP de un PC pueden conectarnos a sistemas TCP/IP de la Ethernet. Se definen utilidades especiales para permitirnos llevar datagramas IP desde el PC hasta el gateway, a través del LocalTalk. Las aplicaciones TCP/IP de PC han sido escritas usando unas librerías especiales que mezclan AppleTalk y TCP/IP. Las utilidades AppleTalk se necesitan para llevar los datagramas hasta el gateway, donde se transformarán en datagramas 100% TCP/IP, antes de dejarlos en la Ethernet.
* Se pueden escribir aplicaciones AppleTalk para grandes sistemas, de tal manera que un PC podrá usarlos como servidores. Dichas aplicaciones también han sido escritas haciendo uso de una librería especial que mezcla

AppleTalk y TCP/IP. Pero, en esta ocasión, son utilidades TCP/IP para dejar datagramas en el gateway, donde se transformarán totalmente en AppleTalk, antes de dejarlos en la AppleTalk y lleguen al PC.

* Una red IP de un campus o una corporación puede ser usada para conectar redes AppleTalk. Los gateways de cada Applet realizarán las conversiones necesarias antes de enviar los datagramas a la red IP.

Además, algunos gateways pueden hacer traducciones a nivel de aplicación. Por ejemplo, algunos gateways pueden hacer traducciones entre el sistema de archivos de Apple y el sistema de archivo de red de Sun (NFS). Esto permite a un PC acceder al sistema de archivos Unix, donde el PC usa el sistema de archivos Apple, y el acceso al sistema Unix se hace mediante el uso del sistema NFS, o sistema de archivos de red (Network File System), de Sun.

Desafortunadamente, la gran flexibilidad de estos productos se traduce en una gran complejidad. El tema de direcciones es especialmente complicado. Por las mismas razones que SLIP, y PPP estos gateways usan frecuentemente asignación dinámica de direcciones IP. Para ello asignaremos un rango de direcciones IP a cada gateway. Cuando un PC intenta abrir una conexión TCP/IP, el gateway se hace con una dirección IP libre y se la asigna al PC. Al igual que SLIP, en muchos casos necesitaremos elegir si queremos que las direcciones se asignen de esta manera, o bien queremos que cada PC tenga su propia dirección. Otra vez, la elección dependerá del número de PC's que tengamos y de si tenemos aplicaciones capaces de usar la dirección IP para identificar qué PC, en particular, es el que está conectado.

El direccionamiento es mucho más complejo, debido a que AppleTalk tiene su propia estructura de direcciones. Deberemos establecer una correspondencia entre direcciones AppleTalk y números de red IP. También habrá una correspondencia entre direcciones IP y AppleTalk, que se establecerá dinámicamente en los gateways.

SERVICIOS A NIVEL DE RED, NOMBRES

Si vamos a tener una red TCP/IP, hay algunas tareas importantes que realizar. Algunas de ellas son simplemente de tipo administrativo. La más importante es crear un registro central de nombres y direcciones IP. Existen organizaciones que realizan esta labor para toda la red Internet. Si estamos conectados a Internet, el administrador de nuestro sistema necesita registrarse a una de estas organizaciones, para que cualquier demanda por parte de otra institución sobre nuestros hosts sean dirigidos a nuestros servidores.

Queremos mantener una base de datos que contenga la información de cada sistema de la red.

Como mínimo, necesitaremos el nombre y la dirección IP de cada sistema. Probablemente, el registro central será el encargado de asignar las direcciones IP. Si nuestra red está estructurada en subredes, o si usamos varios números de clase C, el registro posiblemente asignará los números de red a las nuevas redes o subredes. Pero, habitualmente, se permitirá que los propios administradores de los hosts elijan el nombre del host. Sin embargo, el registro debe de, al menos, verificar que no haya nombres duplicados. Si estamos trabajando con una gran red, puede que sea buena idea delegar algunas de estas tareas a subregistros, posiblemente uno para cada departamento.

Se recomienda asignar las direcciones de la forma más simple: empezando por 1. Así, si nuestra red es la 128.6, podríamos asignar como 128.6.1 a la primera subred; 128.6.2, a la segunda, etc. La asignación de direcciones IP para hosts individuales podrían empezar por 2.

De esta manera reservamos la dirección 1 de cada subred para que sea usada por el Gateway correspondiente. Por consiguiente, el primer host de la subred 128.6.4

sería el 128.6.4.2; el siguiente sería 128.6.4.3, y así sucesivamente. Hay una razón básica para mantener las direcciones tan cortas como sean posibles. Si tenemos una gran organización, podríamos quedarnos sin números de subred. Si esto ocurriera, y nuestros hosts tienen números de red bajos, podríamos asignar otro bit para el direccionamiento de las subredes. Si, por ejemplo, usamos el tercer octeto como número de subred, en tanto en cuanto nuestros hosts tengan unos números inferiores a 128, podremos ampliar el número de red a 9 bits. Así, por ejemplo, la subred 128.6.4 podría dividirse en dos subredes distintas: 128.6.4.0 y 128.6.4.128. Si hubiésemos asignado a los hosts números por encima de 128, la división habría sido imposible.

La asignación de nombres de los hosts no es tan sistemática. Pueden ser cualquier expresión compuesta de letras, números y guiones. Es más seguro que el primer carácter sea una letra.

Y, desde el punto de vista de los usuarios, es recomendable que los nombres sean lo más cortos posibles (incluso hay software que tiene problemas trabajando con nombres más largos de 16 caracteres). Muchas veces, los departamentos o proyectos eligen un tema o nombre relacionado con ellos.

Si estamos conectados a Internet, nuestra organización necesitará un "nombre de dominio" (domain name). Al igual que en el caso del espacio de direcciones IP, la autoridad máxima del espacio de nombres de Internet (DNS, Domain Name System) es la IANA (Internet Assigned Number Authority). La raíz del DNS es gestionada por el InterNIC por delegación de la IANA. Bajo la raíz se encuentran los distintos dominios de primer nivel (Top Level Domains o TLD's) gestionados por distintos registros delegados de Internet. Algunos de ellos son:

Dominios "especiales" como COM, ORG, NET, EDU,. controlados por InterNIC (nodo central del Network Internet Center); y dentro de los dominios nacionales, el dominio ES, correspondiente a España, está delegado a ES-NIC.

A diferencia del número de red, podremos arreglárnosla sin él si la red está aislada. Si posteriormente lo necesitamos, es fácil de añadir un nombre de dominio (recomendamos usar un número de red desde el principio, porque cambiar números de red posteriormente puede ser traumático). Los nombres de dominio, normalmente, terminan en .EDU para las instituciones educativas, .COM, para las compañías, etc.

Si tenemos más de uno o dos sistemas, necesitaremos tener algún mecanismo para tener al día la información de los distintos hosts. El software TCP/IP necesita ser capaz de traducir nombres de hosts en direcciones IP. Cuando un usuario intenta conectarse con otro sistema, generalmente se referirá a él usando su nombre. El software tendrá que traducir el nombre en una dirección IP, para poder abrir la conexión. La mayoría del software incluye dos vías para hacer esta traducción: una tabla estática o un servidor de nombres. La solución de la tabla está indicada para pequeñas organizaciones, siempre y cuando no estén conectadas a otra red.

Simplemente se crea un archivo que lista los nombres y direcciones de todos los hosts.

Veamos parte de una tabla de este tipo:

HOST: 128.6.4.2, 128.6.25.2: ARAMIS.GROUCHO.EDU, ARAMIS: SUN-3-280: UNIX ::
HOST: 128.6.4.3: GAUSS.GROUCHO.EDU, GAUSS: SUN-3-180: UNIX ::
HOST: 128.6.4.4, 128.6.25.4: ATHOS.GROUCHO.EDU, ATHOS: SUN-4-280: UNIX ::

Como se puede apreciar, el formato es el siguiente: una línea para cada sistema y listar sus direcciones, nombres y otra información sobre él. En el ejemplo, tanto

ARAMIS como ATHOS están en dos redes, así que tienen dos direcciones. Además, ambos tienen un nombre principal, por ejemplo ARAMIS.GROUCHO.EDU, y apodos, por ejemplo ARAMIS. En caso de estar conectados a Internet, el nombre principal será el nombre de dominio completamente especificado. Se incluyen apodos cortos, para facilitar la tarea a nuestros usuarios. Hay otro formato muy frecuente para las tablas de hosts. Veamos un ejemplo:

128.6.4.2 aramis.groucho.edu aramis
128.6.25.2 aramis.groucho.edu aramis
128.5.4.3 gauss.groucho.edu gauss
128.6.4.4 ahtos.groucho.edu athos
128.6.25.4 athos.groucho.edu athos

En este formato, cada línea representa una dirección IP. Si el sistema tiene dos interfases, hay dos líneas de él en la tabla. Se debe procurar poner, en primer lugar, aquellas direcciones de uso más común. La documentación de su sistema le informará sobre el formato usado por él.

En la configuración más simple, cada computadora tiene su propia copia de la tabla de hostsen/etc/hosts. En caso de elegir esta configuración, deberemos establecer procedimientos para asegurarnos que todas las copias son actualizadas regularmente. En una red pequeña no es dificil mantener una tabla /etc/hosts en cada máquina, y modificarla al agregar, eliminar o modificar nodos. Aunque resulta complicado cuando hay muchas máquinas, ya que, en principio, cada una necesita una copia de /etc/hosts.

Una solución a esto es compartir ésta y otras bases de datos con el NIS, o sistema de información de red (Network Information System), desarrollado por Sun Microsystems y conocido también como páginas amarillas o YP. En este caso, las bases de datos como la de /etc/hosts se mantienen en un servidor NIS central y

los clientes accederán a ellas de forma transparente al usuario. En todo caso, esta solución sólo es aconsejable para redes pequeñas o medianas, ya que implican mantener un archivo central /etc/hosts que puede crecer mucho, y luego distribuirlo entre los servidores NIS.

En redes grandes, y todos aquellos que están conectados a Internet, debemos adoptar un nuevo sistema, el DNS o sistema de nombres por dominios (Domain Name System) diseñado por Paul Mockapetris. Técnicamente, el DNS es una inmensa base de datos distribuida jerárquicamente por toda la Internet; existen infinidad de servidores que interactúan entre si para encontrar y facilitar las aplicaciones a clientes que los consultan, la traducción de un nombre a su dirección de red IP asociada, con la cual poder efectuar la conexión deseada.

Cada parte de la base de datos está replicada en, al menos, dos servidores, lo que asegura una debida redundancia. Un servidor de nombres es un programa que se ejecuta en algunos de nuestros sistemas para tener conocimiento de los nombres. Cuando un programa necesita buscar un nombre, en lugar de hacerlo en una copia de la tabla de host, envía una petición al servidor de nombres. Este enfoque tiene dos ventajas:

* Para los grandes sistemas, es más fácil tener al día las tablas en algunos servidores de nombres que en todo el sistema.
* Si nuestra red está conectada a Internet, nuestro servidor de nombres será capaz de dialogar con los servidores de nombres de otras organizaciones, para buscar nombres de cualquier sitio.

Usar un servidor de nombres es el único camino para tener un acceso completo a la información del resto de los hosts de Internet.

Es importante comprender la diferencia entre un servidor de nombres y un resolvedor. Un servidor de nombres es un programa que tiene acceso a una base

de datos de hosts, y responde peticiones de otros programas. Un resolvedor es un conjunto de subrutinas que pueden cargarse con un programa. El resolvedor genera las peticiones que se enviarán al servidor de nombres, y procesa sus correspondientes respuestas. Cada sistema debería usar un resolvedor (en general, el resolvedor es cargado por cada programa que va a hacer uso de la red, puesto que sólo es un conjunto de subrutinas). Hay que recalcar que sólo se necesitarán unos pocos servidores de nombres. Mucha gente confunde los dos enfoques y llega a creer que es necesario tener un servidor de nombres en cada computadora.

Para usar un resolvedor, cada computadora necesitará un archivo de configuración, u otro método, para especificar la dirección del servidor de nombres al que enviar nuestras peticiones. Por regla general, se pueden declarar varios servidores de nombres, para el caso de que alguno de ellos no funcione correctamente. En el caso de que nuestro sistema no pudiera contactar satisfactoriamente con ningún servidor, la mayoría de nuestro software empezaría a fallar. Por tanto, hay que ser muy cuidadoso y declarar tantos servidores como podamos para intentar asegurar un buen funcionamiento.

Los servidores de nombres, generalmente, tienen un conjunto de opciones para su configuración. En lugar de dar algunos consejos sobre cómo configurar un servidor de nombres, vamos a recomendar dos documentos oficiales de los estándares de Internet. El RFC 1032 contiene las instrucciones sobre cómo conseguir un nombre de dominio del Centro de Información de Red, incluyendo los formularios necesarios. El RFC 1033 contiene las instrucciones sobre cómo configurar un servidor de nombres. Todos estos documentos son de tipo conceptual. Seguramente, también necesitará documentación sobre el software específico de su servidor de nombres.

En algunos casos, puede que se necesiten, a la vez, tablas y servidores de nombres. Si tenemos alguna implementación de TCP/IP que no incluyan resolvers,

estamos obligados a instalar tablas de hosts en estos sistemas. Si nuestra red está conectada a Internet, vamos a tener problemas con aquellos sistemas que no dispongan de resolvers, ya que Internet es demasiado grande para tener unas tablas de hosts de todos sus hosts. Por lo tanto, lo que se puede hacer es incluir una tabla de hosts con los hosts que realmente se tiene pensado usar. InterNIC tiene a su cargo una tabla de host que puede ser un buen punto de comienzo, aunque no es completa de ningún modo. Así que tendremos que añadir los hosts favoritos de los usuarios. Los sistemas que usan resolvers no tendrán este problema, puesto que un servidor de nombres es capaz de traducir cualquier nombre legal de host.

Los nombres de Hosts y la asignación de números son los únicos elementos que deben de tener una estructura centralizada. Sin embargo, puede haber otros elementos susceptibles de centralización. Es bastante frecuente tener una o dos computadoras que se hagan cargo de todo el correo electrónico. Si estamos conectados a Internet, es bastante simple establecer comunicaciones con otras computadoras de Internet. No obstante, hay muchas instituciones que quieren comunicarse con sistemas de otras redes, como Bitnet o Usenet. Hay gateways entre varias de estas redes. Pero la elección del gateway correcto, y transformar la dirección de correo electrónico correctamente, es una tarea muy especializada. Por esto, en muchas ocasiones se configura el software apropiado sólo en un lugar, y todo el correo externo (o todo el correo externo a hosts que no están en Internet) se dirige a este sistema.

CONFIGURANDO EL ENRUTAMIENTO DE CADA COMPUTADORA

Todas las implementaciones TCP/IP necesitan alguna configuración en cada host. En algunos casos, esto se hace durante la instalación del sistema de forma casi automática. En otros casos, mediante la configuración de ciertos programas o

archivos. Y, por último, otros sistemas obtienen la información de configuración a través de la red de un "servidor".

A pesar de que los detalles de la configuración pueden diferir bastante, existen ciertos datos que deben incluirse en todos los casos. Entre ellos:

- Parámetros que describan a una máquina en particular, como su dirección IP;
- Parámetros que describan la red, como su submáscara de red (si hubiera);
- Software de enrutamiento y las tablas que use;
- Otros programas necesarios para el funcionamiento de otras tareas de red.

Antes de que se instale una computadora en una red, un coordinador deberá asignarle un nombre de red y su dirección IP, como describimos anteriormente. Una vez otorgado un nombre y una dirección estamos en disposición de configurarlo. En numerosas ocasiones, lo que debemos hacer es poner la dirección y el nombre en un archivo de configuración. Sin embargo, algunas computadoras (especialmente aquellos que no disponen de un disco propio en el que dicha información pueda ser almacenada) deben obtener esta información a través de la red. En el momento en que un sistema arranca, se realiza un broadcast a la red con la petición "¿quién soy?". En el caso de poseer computadoras de este tipo, debemos asegurarnos de que nuestra red está preparada para responder adecuadamente. La pregunta lógica es: ¿cómo otro sistema sabe quién eres?.

Generalmente, esto se soluciona haciendo uso de las direcciones Ethernet (o las direcciones análogas para otro tipo de redes). Las direcciones Ethernet son asignadas por los fabricantes hardware. Está garantizado que sólo una máquina en todo el mundo tiene una determinada dirección Ethernet. Por lo general, dicha dirección está grabada en una ROM en la tarjeta Ethernet de la máquina. La máquina, probablemente, no conozca su dirección IP, pero sin duda conoce su dirección Ethernet. Por esta razón, la petición "¿quién soy?" incluye la dirección

Ethernet. Y habrá sistemas configurados para responder a estas peticiones, buscando en una tabla que hace corresponder a cada dirección Ethernet su dirección IP. Pero, por desgracia, deberemos configurar y actualizar esta tabla periódicamente. Para este fin se usa el protocolo de RARP (Reverse Address Resolution Protocol); existe además otro protocolo, el BOOTP o protocolo de arranque. En general, las computadoras están diseñadas de tal manera que muestran su dirección Ethernet por pantalla, tan pronto como se enciende la misma. Y, en la mayoría de los casos, disponemos de un comando que muestra esta información del interfaz Ethernet.

Generalmente, la máscara de subred debe especificarse en un determinado archivo (en los sistemas Unix, el comando "ifconfig", donde “if” significa interfase, se usa para especificar tanto la dirección Internet como la máscara de subred). No obstante, hay previsiones en los protocolos IP para permitir un broadcast de una computadora, preguntando por la máscara de red. La submáscara de red es un atributo de la red y, por ello, es el mismo para todos las computadoras de una determinada subred. No hay una tabla de subred independiente de la tabla de las correspondencias Ethernet/ Internet, usada para consulta de direcciones.

Idealmente, sólo determinadas computadoras contestan peticiones de la máscara de red, pero, en muchas implementaciones TCP/IP, están diseñadas de tal manera que si una computadora cree conocer la máscara de red debe contestar, y, por tanto, en estas implementaciones, la mala configuración de la máscara de subred en un sola computadora puede causar un mal funcionamiento de la red.

Por regla general, los archivos de configuración hacen, a grosso modo, las siguientes cosas:

* Cargar un driver especial para los dispositivos que sean necesarios (esto es bastante usual en los PC's, donde los accesos a red son controlados por una tarjeta controladora y un software que no forma parte del sistema operativo).

* Habilitar cada interfaz de red (Ethernet, líneas serie, etc.). Normalmente, esto conlleva la especificación de una dirección Internet y una máscara de red para cada uno, así como otras opciones especiales de cada dispositivo.
* Establecimiento de la información de enrutamiento de la red, tanto por comandos que establecen rutas, como ejecutando un programa que las obtiene dinámicamente.
* Activar el sistema de dominios (usado para buscar nombres y encontrar la correspondiente dirección Internet -mirar la sección del sistema de dominio, en la Introducción al TCP/IP). Los detalles dependerán del sistema de dominios usado. En la mayoría de los casos, sólo algunos hosts deberán ejecutar el servidor de nombres de dominios. Lo otros hosts, simplemente, necesitan archivos de configuración, que especifican dónde se encuentra el servidor más cercano.
* Establecer otro tipo de información necesaria para el sistema software, como por ejemplo, el nombre del propio sistema.
* Lanzar varios demonios ("daemons"). Hay programas que proveen de servicios de red a otros sistemas de la red, y a los usuarios de estos sistemas.

En el caso de los PC's, que en muchos casos no soportan el multiproceso, y dichos servicios, se establecen mediante los llamados "TSR", o mediante los drivers del dispositivo.

COMO ENRUTAR LOS DATAGRAMAS

Si nuestro sistema consiste en una simple Ethernet, o un medio similar, no será necesario prestar demasiada atención al enrutamiento. Pero, para sistemas más complejos, cada una de las máquinas necesita una tabla que contenga el gateway y el interfaz necesario para cada posible destino. Vimos un ejemplo simple en una sección anterior, pero ahora es necesario describir el modo como funciona el enrutamiento, con un poco más de detalle. En la inmensa mayoría de los sistemas, la tabla de enrutamiento tendrá un aspecto similar (este ejemplo ha sido

tomado de un sistema con Berkeley Unix, usando el comando "netstat -n -r"; algunas columnas que contienen información estadística han sido omitidas):

Destino	Gateway	Bandera	Interfase
128.6.5.3	128.6.7.1	UHGD	il0
128.6.5.21	128.6.7.1	UHGD	il0
127.0.0.1	127.0.0.1	UH	lo0
128.6.4	128.6.4.61	U	pe0
128.6.6	128.6.7.26	U	il0
128.6.7	128.6.7.26	U	il0
128.6.2	128.6.7.1	UG	il0
10	128.6.4.27	UG	pe0
128.121	128.6.4.27	UG	pe0
default	128.6.4.27	UG	pe0

El sistema del ejemplo está conectado a dos Ethernet:

Controlador	Red	Dirección	Otras Redes
il0	128.6.7	128.6.7.26	128.6.6
pe0	128.6.4	128.6.4.61	ninguna

La primera columna muestra el nombre de la interfase Ethernet; la segunda, es el número de red para esa Ethernet; la tercera columna es la dirección Internet de esa red, y, la última muestra otras subredes que comparten la misma red física.

Estudiemos la tabla de enrutamiento; por el momento, ignoraremos las tres primeras líneas. La mayor parte de la tabla consiste en un conjunto de entradas describiendo las redes. Para cada red, las otras tres columnas muestran a dónde deben ser enviados los datagramas destinados a dicha red. Si aparece la bandera

"G" en la tercera columna, los datagramas tienen que enviarse a través de un gateway; en caso de no aparecer, la computadora está directamente conectada a la red en cuestión. Así que los datagramas para dichas redes deben enviarse usando el controlador especificado en la cuarta columna. La bandera "U", de la tercera columna, sólo indica que la ruta especificada por esa línea está activa (generalmente, se asume que estará abierta, a no ser que se produzcan errores tras varios intentos).

Las tres primera líneas muestran "rutas a hosts", indicándose con "H" en la tercera columna.

Las tablas de enrutamiento, normalmente, tienen entradas para redes o subredes. Por ejemplo, la entrada

128.6.2 128.6.7.1 UG il0

indica que un datagrama para cualquier computadora de la red 128.6.2 (es decir, direcciones desde 128.6.2.1 hasta 128.6.2.254) debe enviarse al gateway 128.6.7.1, para llevarlo a cabo.

En algunas ocasiones, se establecen rutas para una computadora específica, en lugar de una red entera. En este caso, se usa una ruta al host. En la primera columna aparece una dirección completa, y la bandera "H" está presente en la columna tres; por ejemplo, la entrada

128.6.5.21 128.6.7.1 UHGD il0

indica que un datagrama, dirigido en concreto a la dirección 128.6.5.21, debe ser enviado al gateway 128.6.7.1. Al igual que en los enrutamientos a redes, la bandera "G" se usa cuando en el enrutamiento se ve involucrado un gateway, y la bandera "D" indica que el enrutamiento fue añadido dinámicamente, usando un

mensaje ICMP de redirección desde un gateway (más adelante daremos más detalles).

El siguiente enrutamiento es especial:

127.0.0.1 127.0.0.1 UH lo0

donde, 127.0.0.1 es el dispositivo de "lazo cerrado", o loopback. Cualquier datagrama enviado a través de este dispositivo aparece inmediatamente como entrada. Es muy útil para hacer pruebas. Las direcciones de "lazo cerrado" pueden, también, ser usadas para comunicar aplicaciones que están en la propia computadora (¿Por qué molestarnos en usar la red para comunicarnos con programas que se encuentran en la misma máquina?).

Por último, hay una ruta por defecto ("default"), como es default 128.6.4.27 UG pe0

Esta ruta será seguida por aquellos datagramas que no correspondan con ninguna de las anteriores. En nuestro ejemplo, se enviarán a un gateway de dirección 128.6.4.27.

Como último ejemplo veamos la tabla de enrutamiento de un sistema Linux conectado a Internet mediante una linea PPP, usando el comando "netstat -n -r"; algunas columnas que contienen información estadística han sido omitidas.

Destino	Gateway	Bandera	Interfase
172.16.1.33	0.0.0.0	UH	ppp0
128.0.0.1	0.0.0.0	U	l0
0.0.0.0	172.16.1.33	UG	ppp0

Hay que aclarar que 0.0.0.0 representa al enrutamiento por defecto, es el valor numérico de default. En este ejemplo, al sistema se le ha asignado la dirección IP 172.16.1.3 de forma dinámica, de manera que usa la línea PPP para conectarse con Internet, y 127.0.0.1 es el dispositivo loopback. Antes de la conexión PPP solamente estaba activo el dispositivo de “lazo cerrado”, pero una vez establecida la conexión PPP se activa el interfase ppp0 (0 indica un identificativo de interfase ppp; es decir, si hubiera otra línea ppp se etiquetaría como ppp1, etc), se usa el sistema del otro lado de la linea como un gateway por defecto, como se puede apreciar en la última línea.

En muchos sistemas, los datagramas son enrutados consultando la dirección de destino en una tabla como la que acabamos de describir. Si la dirección se corresponde con una ruta específica a un host, ésta será usada; en otro caso, si se corresponde con un enrutamiento a red, se usará ésta; y, si nada de lo anterior acontece, se usará el enrutamiento por defecto. En caso de no existir uno por defecto, aparecería un mensaje de tipo "red inalcanzable" ("network is unreachable").

En las siguientes secciones describiremos varias maneras de configurar estas tablas de enrutamiento. Generalmente, la operación de enviar datagramas no depende del método usado en la configuración de estas tablas. Cuando un datagrama va a ser enviado, su destino es consultado en la tabla. Los distintos métodos de enrutamiento son simplemente, más o menos, una serie de sofisticadas formas de configurar y mantener las tablas.

RUTAS FIJAS

La forma más fácil de configurar el enrutamiento es usar comandos que lo fijan. Nuestros archivos de inicialización contienen comandos que configuran el enrutamiento. Si es necesario algún cambio, deberá hacerse, normalmente, usando comandos que añaden y borran entradas de la tabla de enrutamiento

(cuando se realice un cambio, no debemos olvidar actualizar el archivo de inicialización también). Este método es práctico para redes relativamente pequeñas, especialmente cuando los cambios no son muy frecuentes.

Muchos computadoras configuran automáticamente algunas entradas de enrutamiento por nosotros. Unix añade una entrada para las redes a las que estamos directamente conectados.

Por ejemplo, un archivo de inicialización podría ser

```
ifconfig  ie0128.6.4.4   netmask   255.255.255.0
ifconfig  ie1128.6.5.35  netmask   255.255.255.0
```

Este especifica que hay dos interfases de red y sus direcciones en ellas. El sistema crea automáticamente estas entradas en la tabla de enrutamiento

```
128.6.4     128.6.4.4    U    ie0
128.6.5     128.6.5.35   U    ie1
```

y, en ésta, se especifica que los datagramas para las redes locales 128.6.4 y 128.6.5 deben ser enviados a las correspondientes interfases.

Además de éstos, el archivo de inicialización podría contener comandos para definir rutas a cualquier otra red a la que queramos acceder. Por ejemplo,

```
route add   128.6.2.0    128.6.4.1    1
route add   128.6.6.0    128.6.5.35   0
```

Estos comandos determinan que para alcanzar la red 128.6.2 debemos usar el gateway de dirección 128.6.5.1, y esa red 128.6.6 es, realmente, un número de red adicional para una red física conectada al interfase 128.6.5.35. Otro tipo de

software puede usar comandos distintos a estos casos. Unix se diferencia de ellos por el uso de una métrica, que es el número final del comando. La métrica indica cuántos gateways tiene que atravesar un datagrama para alcanzar su destino. Rutas de métrica 1 ó más indican que hay en el camino sólo un gateway hasta el destino. Rutas de métrica 0 indican que no hay ningún gateway implicado -es un número de red adicional para la red local-.

En último lugar, podemos definir un enrutamiento por defecto, usado cuando el destino no está listado explícitamente. Normalmente, se suele acompañar de la dirección de un gateway que tiene suficiente información como para manejar todos los posibles destinos.

Si nuestra red sólo dispone de un gateway, entonces sólo necesitaremos una sola entrada por defecto. En este caso, no deberemos preocuparnos más de la configuración del enrutamiento de los hosts (el gateway, en sí, necesitará más atención, como veremos). Las siguientes secciones nos ayudarán para configurar redes donde hay varios gateways.

RECONDUCIR EL ENRUTAMIENTO

La mayoría de los expertos recomiendan dejar las decisiones de enrutamiento a los gateways.

Por tanto, probablemente, será una mala idea tener largas tablas estáticas de enrutamiento en cada computadora. El problema está en que cuando algo cambia en la red tenemos que actualizar las tablas en demasiadas computadoras. Si el cambio ocurre debido a que cae una línea, el servicio no se restablecerá hasta que alguien se de cuenta del problema y cambie todas las tablas de enrutamiento.

La manera más fácil de tener actualizado el enrutamiento es depender sólo de un único gateway y actualizar su tabla de enrutamiento. Este gateway debe fijarse

como gateway por defecto (En Unix esto significa usar un comando como "route add default 128.6.4.27.1", donde 128.6.4.27 es la dirección del gateway). Como describimos anteriormente, el sistema enviará todos aquellos datagramas a dicho gateway cuando no haya una ruta mejor. En principio, parece que esta estrategia no es demasiado buena cuando poseemos más de un gateway; máxime, cuando todo lo que tenemos es sólo la entrada por defecto. ¿Cómo usaremos los otros gateways en los casos en los que éstos sean más recomendables? La respuesta es que los datagramas correspondientes serán redirigidos a estos gateways en estos casos.

Un "redireccionamiento" es una clase específica de mensaje ICMP (Internet Control Message Protocol), que contiene información del tipo "en el futuro, para llegar a la dirección XXXXX, intenta usar YYYYY en lugar de mí". Las implementaciones que cumplen completamente los protocolos TCP/IP usan estas técnicas de redireccionamiento para añadir entradas a las tablas de enrutamiento. Supongamos que una tabla inicialmente es como sigue: donde hay una entrada para la red local 128.6.4, y una entrada por defecto del gateway 128.6.4.27. Supongamos que hay también un gateway 128.6.4.30, que es el mejor camino para acceder a la red 128.6.7. ¿Cómo podemos llega a usar este camino? Supongamos que tenemos unos datagramas para enviar a 128.6.7.23. El primer datagrama llegará al gateway por defecto, puesto que es el único que aparece en la tabla de enrutamiento, y el gateway se dará cuenta de que el mejor camino debe pasar por 128.6.4.30 (hay distintos métodos para que un gateway determine que debe usarse otro para un mejor enrutamiento). Por tanto, 128.6.4.27 contestará con un mensaje de redireccionamiento especificando que los datagramas para 128.6.7.23 deben enviarse a través del gateway 128.6.4.30. El software TCP/IP añadirá una entrada a la tabla de enrutamiento 128.6.7.23 128.6.4.30 UDHG pe0.

De esta manera, los restantes datagramas al 128.6.7.23 se enviarán directamente al gateway apropiado. Esta estrategia sería perfecta si no fuera por los siguientes tres problemas:

* Necesita que cada computadora contenga la dirección de un gateway por defecto en los archivos de configuración.
* En caso de que un gateway falle, las entradas de las tablas de enrutamiento que usan dicho gateway no se eliminan.
* Si la red usa subredes y la implementación TCP/IP usada no las maneja, esta estrategia no podrá emplearse.

El alcance del problema depende del tipo de red de la que disponemos. Para redes pequeñas, apenas supondrá un problema cambiar los archivos de configuración de algunas máquinas. Sin embargo, para algunas organizaciones este trabajo es difícil de llevar a cabo. Si, por ejemplo, la topología de la red cambia y un gateway es eliminado, cualquier sistema que tenga dicho gateway por defecto deberá ser ajustado. Este problema será especialmente grave si el personal encargado del mantenimiento de la red es distinto del encargado de mantener a los sistemas individualmente. La solución más simple consiste en asegurarnos de que la dirección por defecto nunca cambiará. Por ejemplo, podríamos adoptar el convenio de que la dirección 1 de cada subred se corresponde con el gateway por defecto de cada subred; así, en la subred 128.6.7, el gateway por defecto sería siempre el 128.6.7.1. Si dicho gateway es eliminado, habrá que asignarle dicha dirección a algún otro gateway (siempre tendrá que haber, al menos, un gateway, puesto que si no es así estaremos completamente incomunicados).

Hasta ahora hemos visto cómo añadir rutas, pero no cómo deshacernos de ellas. ¿Qué ocurre si un gateway no funciona correctamente?. Nuestro deseo sería que se recondujera a un gateway operativo, pero desgraciadamente, un gateway en mal funcionamiento no tendrá en general esta capacidad de redireccionamiento.

La solución más obvia es usar gateways fiables. El redireccionamiento puede usarse para controlar distintos tipos de fallos.

La mejor estrategia para controlar gateways averiados es que nuestra implementación TCP/IP detecte las rutas que no tienen éxito. TCP mantiene varios contadores que permiten al software detectar cuándo una conexión se ha roto. Cuando esto ocurre, se puede marcar esta ruta como fallida y volver al gateway por defecto. Una solución similar puede usarse para manejar fallos en el gateway por defecto. Si configuramos dos gateways por defecto, entonces el software deberá ser capaz de cambiar el gateway cuando las conexiones en uno de ellos empiecen a fallar. Sin embargo, algunas implementaciones TCP/IP no pueden marcar rutas como fallidas y empezar a usar otras. En particular, Berkeley 4.2 Unix no lo hace; pero Berkeley 4.3 Unix sí, lo que empieza a hacerse cada vez más común. Hasta implementaciones de Unix para PC como Linux ya incorporan esta posibilidad (Linux en concreto puede controlar hasta cuatro gateways por defecto).

OTROS METODOS PARA QUE LOS HOSTS ENCUENTREN RUTAS

En tanto en cuanto las implementaciones TCP/IP manejan caídas de las conexiones adecuadamente, estableciendo una o más rutas por defecto en el archivo de configuraciones, se produce probablemente la forma más simple de controlar el enrutamiento. No obstante, hay otras dos técnicas de enrutamiento dignas de consideración para algunos casos especiales:

* espiar el protocolo de enrutamiento,
* usar un proxy ARP.

Espiar el enrutamiento

Los gateways, por regla general, tienen un protocolo especial que usan entre ellos. Hay que aclarar que el redireccionamiento no puede ser usado por los gateways,

ya que éste es simplemente el mecanismo por el cuál ellos informan a simples hosts que tienen que usar otro gateway. Los gateways deben tener una visión completa de la red y un método para calcular la ruta óptima a cada subred. Generalmente, los gateways mantienen esta visión mediante el intercambio de información entre ellos. Hay varios protocolos distintos de enrutamiento para este propósito. Una alternativa para que una computadora siga la pista a los gateways es escuchar los mensajes que se intercambian entre ellos. Hay software capaz de hacer esto para la mayoría de los protocolos. Cuando ejecutamos este software, la computadora mantendrá una visión completa de la red, al igual que los gateways. Este software normalmente está diseñado para mantener dinámicamente las tablas de enrutamiento de la computadora, así que los datagramas se enviarán siempre al gateway más adecuado. De hecho, el enrutamiento realizado es equivalente a ejecutar los comandos Unix "route add" y "route delete" a medida que la topología cambia. El resultado suele ser una completa tabla de enrutamiento, en lugar de una con unas rutas por defecto (este enfoque asume que los gateways mantienen entre ellos una tabla completa. Algunas veces los gateways tienen constancia de todas nuestras redes, pero usan una ruta por defecto para las redes ajenas al campus, etc.).

Ejecutando el software de enrutamiento en cada host resolveremos de alguna manera el problema de enrutamiento, pero hay algunas razones por las que normalmente no es recomendable, reservándola como última alternativa. El problema más serio incorpora numerosas opciones de configuración, que deben mantenerse en cada computadora. Además, los actuales gateways suelen añadir opciones cada vez más complejas. Por tanto, no es deseable extender el uso de este software en todos los hosts.

Hay otro problema más específico referido a las computadoras sin discos. Como es natural, una computadora sin discos depende de la red y de los servidores de archivos para cargar los programas y hacer swapping. No es recomendable que estas computadoras escuchen las emisiones de la red. Por ejemplo, cada gateway

de la red debe emitir sus tablas de enrutamiento cada 30 segundos. El problema es que el software que escucha estas emisiones debe ser cargado a través de la red. En una computadora ocupada, los programas que no son usados durante algunos segundos deben guardarse haciendo swapping o paginación. Cuando se activan de nuevo, han de recuperarse. Cuando una emisión de un gateway es enviada en la red, cada computadora activa su software de red para procesar dicha emisión, lo cual significa que todos ellos intentan hacer una recuperación al mismo tiempo y, por tanto, es probable que se produzca una sobrecarga temporal de la red.

Proxy ARP

Los proxy ARP son otra técnica para permitir a los gateways tomar todas las decisiones de enrutamiento. Son aplicables a aquellas redes que usan ARP (Address Resolution Protocol), o una técnica similar para corresponder las direcciones Internet con direcciones de redes específicas, como las direcciones Ethernet. Para facilitar la explicación, vamos a asumir redes Ethernet. Los cambios necesarios para otros tipos de redes consistirán en poner la correspondiente dirección de red, en lugar de "dirección Ethernet", y protocolo análogo a ARP para dicha red.

En muchos aspectos, los proxy ARP son semejantes al uso de una ruta por defecto y redireccionamiento, y la mayor diferencia radica en que tienen distintos mecanismos para comunicar rutas a los hosts. Con el redireccionamiento se usa una tabla completa de enrutamiento, de forma que en cualquier momento un host sabe a cual gateway debe enviar los datagramas. En cambio, los proxy ARP prescinden de las tablas de enrutamiento y hacen todo el trabajo a nivel de direcciones Ethernet. Los proxy ARP pueden usarse para todos los destinos, tanto para aquellos que están en nuestra red como para algunas combinaciones de destinos. El caso más sencillo de explicar es el de todas las direcciones; para ello ordenamos a la computadora que simule que todos las computadoras del mundo

están conectadas directamente a nuestra Ethernet local. En Unix, esto se hace usando el comando route add default 128.6.4.2 0 donde, 128.6.4.2 es la dirección IP de nuestro host. Como ya hemos visto, la métrica 0 provoca que todo aquello que se identifique con esta ruta se enviará directamente a la red local Ethernet. Alternativamente, otros sistemas nos permiten conseguir el mismo efecto fijando una máscara de red de ceros, en cuyo caso debemos asegurarnos de que no será alterada por un mensaje ICMP de máscara de subred debido a que un sistema conoce la verdadera máscara de red.

Cuando un datagrama va a ser enviado a un destino dentro de la Ethernet local, la computadora necesita conocer la dirección Ethernet del destino, y para ello, generalmente, se usa la llamada tabla ARP, que contiene las correspondencias entre las direcciones Internet y las direcciones Ethernet. Veamos un ejemplo típico de tabla ARP (en la mayoría de los sistemas se visualiza usando el comando "arp -a"):

FOKKER.GROUCHO.EDU (128.6.5.16) at 8:0:20:0:8:22 temporary
CROSBY.GROUCHO.EDU (128.6.5.48) at 2:60:8c:49:50:63 temporary
CAIP.GROUCHO.EDU (128.6.4.16) at 8:0:8b:0:1:6f temporary
DUDE.GROUCHO.EDU (128.6.20.16) at 2:7:1:0:eb:cd temporary
W2ONS.MIT.EDU (128.125.1.1) at 2:7:1:0:eb:cd temporary
OBERON.USC.EDU (128.125.1.1) at 2:7:1:2:18:ee temporary
gatech.edu (128.61.1.1) at 2:7:1:0:eb:cd temporary
DARTAGNAN.GROUCHO.EDU (128.6.5.65) at 8:0:20:0:15:a9 temporary

Como dijimos anteriormente, simplemente es una lista de direcciones IP y su correspondiente dirección Ethernet. El término "temporary" indica que la entrada fue añadida dinámicamente usando ARP, en lugar de ser puesta manualmente.

Si hay una entrada para una dirección determinada en la tabla ARP, los datagramas serán puestos en la Ethernet con su correspondiente dirección

Ethernet. Si esto no ocurre, se enviará una "petición ARP", solicitando que el host destino se identifique. La petición es, en efecto, una pregunta: "¿Puede decirme el host con dirección Internet 128.6.4.194 cuál es su dirección Ethernet?". Cuando llega una respuesta, esta se añade a la tabla ARP y los futuros datagramas con ese destino serán enviados directamente.

Este mecanismo fue diseñado inicialmente sólo para hosts que estuvieran directamente conectados a una simple Ethernet. Si necesitamos comunicarnos con un host que se encuentra en otra Ethernet, se supone que la tabla de enrutamiento lo dirigirá a un gateway. Dicho gateway, como es obvio, deberá tener una interfase en nuestra Ethernet. El host deberá averiguar la dirección de dicho gateway usando ARP. Este procedimiento es más útil que hacer que el ARP trabaje directamente con una computadora en una red lejana, puesto que no están en la misma Ethernet, no disponemos de una dirección Ethernet para poder enviar los datagramas y, al enviar "peticiones ARP" por ellas, nadie nos responderá.

Los proxy ARP se basan en la idea de que los conceptos actúen como proxyes de hosts lejanos. Supongamos que tenemos un host en la red 128.6.5, con direcciones (es la computadora A en diagrama siguiente), que va a enviar un datagrama al host 128.6.5.194 (la computadora C) que se encuentra en una Ethernet distinta (subred 128.6.4). Hay un gateway que conecta ambas subredes, de direcciones 128.6.5.1 (gateway R).

Red 1
128.6.5
Red 2
128.6.4
128 .6 .5.2 128 .6 .5.3 128 .6 .5.1 128 .6 .4.19 4
ordenador A ordenador B
128 .6 .4.1 ordenador C
gateway R

Ahora supongamos que la computadora A envía una petición ARP a la computadora C, pero C no es capaz de responder por sí misma. Al estar en redes distintas, C nunca verá la petición ARP; sin embargo, el gateway actuará en su lugar. En efecto, nuestra computadora pregunta: "¿Puede decirme el host con dirección de Internet 128.6.4.194 cuál es su dirección Ethernet?", y el gateway contesta: "Yo soy 128.6.4.194 es 2:7:1:0:eb:cd", donde 2:7:1:0:eb:cd es la dirección Ethernet del gateway. Este pequeño truco funciona correctamente y hace pensar a nuestro host que 128.6.4.194 está conectado a la Ethernet local con dirección 2:7:1:0:eb:cd, pero, por supuesto, no es cierto. Cada vez que enviamos un datagrama a 128.6.4.194, nuestro host lo envía a la dirección Ethernet especificada y, puesto que es la dirección del gateway R, llega hasta dicho gateway. Y es entonces cuando se envía a su destino.

Veamos que esto tiene el mismo efecto que tener una entrada en la tabla de enrutamiento diciendo que la ruta de 128.6.4.194 al gateway 128.6.5.1 es:

128.6.4.194 128.6.5.1 UGH pe0

Con la excepción de que, en lugar de tener el enrutamiento hecho a nivel de tabla de enrutamiento, se hace a nivel de tabla ARP.

Generalmente, es mejor usar tablas de enrutamiento, pero hay algunos casos en los que tiene sentido usar los proxyes ARP:

* cuando tenemos un host que no trabaja con subredes;
* cuando tenemos un host que no responde adecuadamente al redireccionamiento;
* cuando no queremos elegir un gateway determinado por defecto;
* cuando el software no es capaz de recuperarse de un enrutamiento fallido.

La técnica fue diseñada originariamente para trabajar con hosts que no soportan subredes.

Supongamos que tenemos una red dividida en subredes. Por ejemplo, hemos decidido dividir la red 128.6 en subredes, obteniendo las subredes 128.6.4 y 128.6.5. Supongamos también que tenemos un host que no trabaja con subredes y, por tanto, creerá que 128.6 es tan sólo una red.

Esto último significa que será difícil establecer las entradas para la tabla de enrutamiento para la configuración vista. No podemos decirle nada sobre la existencia del gateway, de forma explícita, usando "route add 128.6.4.0 128.6.5.1 1", puesto que, al considerar que toda la 128.6 es una simple red, no entenderá que intentamos enviarlo a una subred. En su lugar, interpretará este comando como un intento de configurar una ruta a un host de dirección 128.6.4.0. La única manera que podría hacerlo funcionar sería establecer rutas explícitas a los host, para cada host individual sobre cada subred. Tampoco podríamos depender del gateway por defecto y redireccionar. Supongamos que establecemos "route add default 128.6.5.1 1", en el que fijamos el gateway 128.6.5.1 por defecto; esto no podría funcionar para enviar datagramas a otras subredes. En el caso de que el host 128.6.5.2 quiera enviar un datagrama al 128.6.4.194, puesto que el destino es parte de 128.6, la computadora lo considerará en la misma red y no se preocupará por buscarle un gateway adecuado.

Los proxy ARP resuelven el problema haciendo ver el mundo de un modo simplista que espera encontrarse. Puesto que el host piensa que todas las restantes subredes forman parte de su propia red, simplemente usará una petición ARP para comunicarse con ellas, esperando recibir una dirección Ethernet que pueda usarse para establecer comunicaciones directas. Si el gateway ejecuta un proxy ARP, responderá con la dirección Ethernet del gateway. Por tanto, los datagramas serán enviados al gateway y todo funcionará correctamente.

Como se puede observar, no se necesita una configuración específica para usar una proxy ARP con hosts que no trabajan con subredes. Lo que necesitamos es que todos nuestros gateways ARP tengan implementado un proxy ARP. Para poder usarlos, deberemos especificar la configuración de la tabla de enrutamiento. Por defecto, las implementaciones TCP/IP esperarán encontrar un gateway para cualquier destino que esté en otra red y, para hacerlo, deberemos explícitamente instalar una ruta de métrica 0, como por ejemplo "route add default 128.6.5.2 0", o poner la máscara de subred a ceros.

Es obvio que los proxy ARP son adecuados cuando los hosts no son capaces de entender subredes. Generalmente, las implementaciones TCP/IP son capaces de manejar mensajes de redirección ICMP correctamente, y, por tanto, normalmente lo que se hará es configurar la ruta por defecto a algún gateway. Sin embargo, en caso de contar con una implementación que no reconoce los redireccionamientos, o no puede configurarse un gateway por defecto, podemos usar proxy ARP.

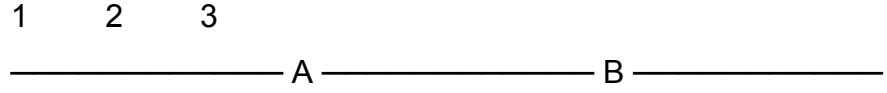
A veces se usa proxy ARP por conveniencia. El problema de las tablas de enrutamiento es que hay que configurarlas. La configuración más simple es fijar una ruta por defecto; pero, incluso en este caso, hay que incluir un comando equivalente al de Unix "route add default...". En el caso de que hubiese cambios en las direcciones de los gateways, deberíamos modificar este comando en todos los hosts. Si configuramos una ruta por defecto que depende de proxy ARP (con métrica 0), no deberemos cambiar los archivos de configuración cuando los gateways cambian. Con los proxy ARP, no hace falta poner ninguna dirección de un gateway. Cualquier gateway puede responder a una petición ARP, no importa cuál sea su dirección.

Para evitarnos tener que configurar los sistemas, algunas implementaciones TCP/IP usan ARP por defecto, cuando no tienen otra ruta. Las implementaciones más flexibles nos permiten usar estrategias mixtas. Así, si tenemos que especificar

una ruta para cada red en particular, o una ruta por defecto, se usará esa ruta, pero si no hay rutas para un destino lo tratará como si fuese local y usará una petición ARP. En tanto en cuanto sus gateways soporten proxy ARP, esto permitirá que los hosts alcancen cualquier destino sin necesitar ninguna tabla de enrutamiento.

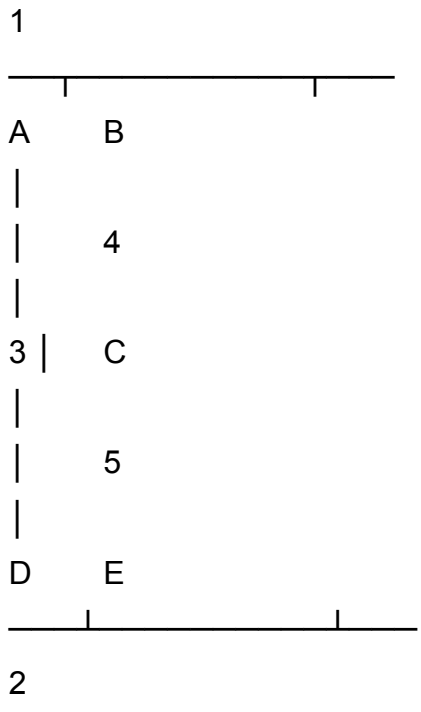
Finalmente, podríamos elegir usar una proxy ARP porque se recuperan mejor de los fallos. La elección dependerá en gran medida de la implementación.

En aquellas situaciones en las que hay varios gateways en una red, veamos cómo los proxy ARP permiten elegir el mejor. Como hemos mencionado anteriormente, nuestro computador simplemente envía un mensaje preguntando por la dirección Ethernet del destino. Suponemos que los gateways están configurados para responder a estos mensajes. Si hay más de un gateway, será necesaria una coordinación entre ellos. Conceptualmente, los gateways tendrán una visión completa de la topología de la red. Por consiguiente, serán capaces de determinar la mejor ruta desde nuestro host a cualquier destino. Si hay una coordinación entre los gateways, será posible que el mejor gateway pueda responder a la petición ARP. En la práctica no es siempre posible, por ello se diseñan algoritmos para evitar rutas malas. Veamos por ejemplo la siguiente situación:



donde, 1, 2 y 3 son redes; y A y B gateways conectando 2 con 1 ó con 3. Si un host de la red 2 quiere comunicarse con otro de la red 1 es bastante fácil para el gateway A decidirse a contestar, y el gateway B no lo hará. Veamos cómo: si el gateway B acepta un datagrama para la red 1, tendrá que remitirlo al gateway A para que lo entregue. Esto significaría que debería tomar un datagrama de la red 2 y enviarlo de vuelta a la red 2. Es muy fácil manejar las rutas que se dan en este

tipo de redes. Es mucho más difícil de controlar en una situación como la siguiente:



Supongamos que una computadora en la red 1 quiere enviar un datagrama a otro de la red 2.

La ruta vía A y D es probablemente la mejor, porque sólo hay una red (3) entre ambas.

También es posible la ruta vía B, C y E, pero este camino probablemente es algo más lento.

Ahora supongamos que la computadora de la red 1 envía peticiones ARP para alcanzar 2.

Seguramente A y B responderán a dicha petición. B no es tan buena como A, pero no hay tanta diferencia como en el caso anterior. B no devolverá el datagrama a 1. Además, no es posible determinar qué camino es mejor sin realizar un costoso análisis global de la red. En la práctica no disponemos de tanta cantidad de tiempo para responder a una petición ARP.

Establecer nuevas rutas tras fallos

En principio, IP es capaz de controlar líneas con fallos y caídas de gateways. Hay varios mecanismos para rectificar las tablas de enrutamiento y las tablas de ARP y mantenerlas actualizadas. Pero, por desgracia, muchas de las implementaciones TCP/IP no implementan todos estos mecanismos, por lo que deberemos estudiar detalladamente la documentación de nuestra implementación y, teniendo en cuenta los fallos más frecuentes, deberemos definir una estrategia para asegurar la seguridad de nuestra red. Las principales elecciones son las siguientes: espiar el protocolo de enrutamiento de los gateways, establecer una ruta por defecto y hacer uso del redireccionamiento y usar proxy ARP. Todos estos métodos tienen sus propias limitaciones dependiendo del tipo de red.

Espiar el protocolo de enrutamiento de los gateways es, en teoría, la solución más directa y simple. Si suponemos que los gateways usan una buena tecnología de enrutamiento, las tablas que ellos envían, deberían contener la información necesaria para mantener unas rutas óptimas para todos los destinos. Si algo cambia en la red (una línea o un gateway, falla), esta información deberá reflejarse en las tablas y el software de enrutamiento deberá ser capaz de actualizar adecuadamente las tablas de enrutamiento de los hosts. Las desventajas de esta estrategia son meramente prácticas, pero, en algunas situaciones, la robustez de este enfoque puede pesar más que dichas desventajas. Veamos cuáles son estas desventajas:

* Si los gateways usan un protocolo de enrutamiento sofisticado, la configuración puede ser bastante compleja, lo que se convierte en un problema ya que debemos configurar y mantener los archivos de configuración de cada host.
* Algunos gateways usan protocolos específicos de alguna marca comercial. En este caso, es posible que no encontremos un software adecuado para nuestros hosts.
* Si los hosts carecen de disco, puede que haya serios problemas a la hora de escuchar las emisiones.

Algunos gateways son capaces de traducir su protocolo interno de enrutamiento en otro más simple que puede ser usado por los hosts. Esta podría ser una forma de resolver las dos primeras desventajas. Actualmente no hay una solución definitiva para la tercera.

Los problemas de los métodos de rutas por defecto/redireccionamiento y de los proxy ARP son similares: ambos tienen problemas para trabajar con situaciones donde las entradas a las tablas no se usan durante un largo periodo de tiempo. La única diferencia real entre ellos son las tablas que se ven involucradas. Supongamos que un gateway cae, si alguna de las actuales rutas usan ese gateway no podrá ser usada. En el caso de que estemos usando tablas de enrutamiento, el mecanismo para ajustar las rutas es el redireccionamiento. Esto funciona perfectamente en dos situaciones:

Cuando el gateway por defecto no está en la mejor ruta. El gateway por defecto puede dirigirlo a un gateway mejor.

Cuando una línea distante o un gateway fallan. Si esto cambia la mejor ruta, el gateway actual nos dirigirá hacia el gateway que ahora es el mejor.

El caso que no está a salvo de problemas es cuando el gateway a que se le envía datagramas falla en ese momento. Puesto que está fuera de servicio, es imposible que redireccione a otro gateway. En muchos casos, tampoco estamos a salvo si el gateway por defecto falla, justo cuando el enrutamiento empieza a enviar al gateway por defecto.

La casuística de los proxy ARP es similar. Si los gateways se coordinan adecuadamente, en principio el gateway indicado responderá adecuadamente. Si algo en la red falla, el gateway que actualmente se está usando nos reconducirá a un nuevo y mejor gateway (normalmente es posible usar redireccionamiento para ignorar las rutas establecidas por el proxy ARP). Otra vez, el caso que no podemos proteger de fallos es cuando el gateway actual falla. No hay equivalencia al fallo de los gateways por defecto, puesto que cualquier gateway puede responder a una petición ARP.

Así que el gran problema es el fallo debido a que el gateway en uso no se puede recuperar, por el hecho de que el principal mecanismo para alterar las rutas es el redireccionamiento, y un gateway en mal funcionamiento no puede redirigir. Teóricamente, este problema podría solucionarse a través de la implementación TCP/IP, usando "timeout". Si una computadora no recibe respuesta una vez terminado el timeout, debería de cancelar la ruta actual y tratar de encontrar otra nueva. Cuando usamos una ruta por defecto, esto significa que la implementación TCP/IP puede ser capaz de declarar una ruta como fallida en base al timeout.

En caso de que se haya redirigido a un gateway distinto del de por defecto, y la ruta se declare fallida, el tráfico se devolverá al gateway por defecto. El gateway por defecto puede entonces empezar a manejar el tráfico, o redirigirlo a un gateway diferente. Para manejar los fallos del gateway por defecto es posible tener más de un gateway por defecto; si uno de ellos se declara fallido, se usará el otro. En conjunto, estos mecanismos nos salvaguardan de cualquier fallo.

Métodos similares pueden usarse en sistemas que dependen de proxy ARP. Si una conexión sobrepasa el timeout, la entrada de la tabla ARP usada se debe borrar. Esto causará una petición ARP, que podrá ser contestada por un gateway que funcione correctamente. El mecanismo más simple para llevar esto a cabo podría ser usar los contadores de timeout para todas las entradas ARP. Puesto que las peticiones ARP no son muy costosas en tiempo, cada entrada cuyo timeout concluya será borrada, incluso si estaba funcionando perfectamente. Así, su próximo datagrama será una nueva petición. Las respuestas, normalmente, son suficientemente rápidas para que el usuario no se de cuenta del retraso introducido.

Sin embargo, algunas implementaciones no usan estas estrategias. En Berkeley 4.2 no hay manera de librarse de ningún tipo de entrada, ni de la tabla de enrutamiento ni de la tabla ARP.

Estas implementaciones no invalidan las entradas, éstas fallan. Luego si los problemas de fallos de gateways son más o menos comunes, no habrá otra opción que ejecutar un software que escuche el protocolo de enrutamiento. En Berkeley 4.3, las entradas son eliminadas cuando las conexiones TCP fallan, pero no las ARP. Esto hace que la estrategia de la ruta por defecto sea más atractiva que la de proxys ARP, si usamos Berkeley 4.3. Si, además, incluimos más de una ruta por defecto se posibilitará la recuperación de fallos cuando falle un gateway por defecto. Si una ruta está siendo usada sólo por servicios basados en el protocolo UDP, no habrá una recuperación de fallos si el gateway implicado cae. Mientras que los servicios "tradicionales" TCP/IP hacen uso del protocolo TCP, algunos otros, como el sistema de archivos de red, no lo hacen. Por tanto, la versión 4.3 no nos garantiza una recuperación de fallos absoluta.

Por último, también podemos hablar de otras estrategias usadas por algunas antiguas implementaciones. Aunque están casi en desuso, vamos a describirlas de forma esquemática.

Estas implementaciones detectan un fallo de un gateway haciendo comprobaciones de qué gateways están en uso. Para ello, la mejor forma de hacer estas comprobaciones es hacer una lista de gateways que actualmente se estén usando (para lo que se ayuda de la tabla de enrutamiento) y cada minuto se envía una petición de "echo" a cada gateway de la citada lista; si el gateway no envía una respuesta se declara como fallido, y todas las rutas que hacen uso de él se reconducirán al gateway por defecto. Generalmente, se deberá de proporcionar más de un gateway por defecto, de manera que si el gateway por defecto falla se elige uno de los alternativos. En otros casos no es necesario especificar un gateway por defecto, ya que el software, aleatoriamente, eligirá un gateway que responda. Estas implementaciones son muy flexibles y se recuperan bien de los fallos, pero una gran red con esta implementación malgastará el ancho de banda con datagramas "echo" para verificar qué gateways funcionan correctamente. Esta es la razón por la que esta estrategia está en desuso.

PUENTES Y GATEWAYS

En esta sección vamos a tratar con más detalle la tecnología usada para construir grandes redes. Vamos a centrarnos especialmente en cómo conectar varias Ethernet, token rings, etc.

Hoy día la mayoría de las redes son jerárquicas. Los hosts están incluidos en una red de área local, como una Ethernet o un token ring. Estas redes se conectan entre sí mediante alguna combinación de redes principales o enlaces punto a punto. Una universidad puede tener una red como se muestra, en parte, a continuación:

Las redes 1, 2 y 3 están en un edificio. Las redes 4 y 5 están en edificios distintos del campus.

La red 6 puede estar en una localización más distante. El diagrama anterior nos muestra que las redes 1, 2 y 3 están conectadas directamente, y los mecanismos que manejan las conexiones se marcan con "x". El edificio A está conectado a otros edificios en el mismo campus por una red principal. El tráfico desde la red 1 a la red 5 tomará el siguiente camino:

- de 1 a 2 a través de la conexión entre estas redes;
- de 2 a 3 a través de su conexión directa;
- de 3 a la red principal;
- a través de la red principal, desde el edificio A al edificio donde la red 5 está emplazada;
- de la red principal a la red 5.

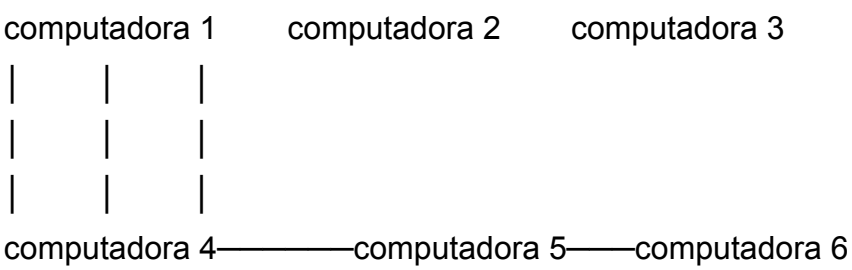
El tráfico hacia la red 6 debería pasar adicionalmente a través de la línea serie. Con la misma configuración, se usaría la misma conexión para conectar la red 5 con la red principal y con la línea serie. Así, el tráfico de la red 5 a la red 6 no necesita pasar a través de la red principal, al existir esa conexión directa entre la red 5 y la línea serie. En esta sección vamos a ver qué son realmente estas conexiones marcadas con "x".

DISEÑOS ALTERNATIVOS

Hay que hacer constar que hay distintos diseños alternativos al mostrado anteriormente. Uno de ellos es usar líneas punto a punto entre los hosts, y otro puede ser usar una tecnología de red a un nivel capaz de manejar tanto redes locales como redes de larga distancia.

Una red de líneas punto a punto

En lugar de conectar los hosts a una red local como una Ethernet, y luego conectar dichas Ethernets, es posible conectar directamente las computadoras a través de líneas serie de largo alcance. Si nuestra red consiste primordialmente en un conjunto de computadoras situadas en localizaciones distintas, esta opción tiene sentido. Veamos un pequeño diseño de este tipo:



En el primer diseño, la tarea de enrutamiento de los datagramas a través de red era realizada por unos mecanismos de propósito específico que marcábamos con "x". Si hay líneas que conectan directamente un par de hosts, los propios hosts harán esta labor de enrutamiento, al mismo tiempo que realizan sus actividades normales. A no ser que haya líneas que comuniquen directamente todos los hosts, algunos sistemas tendrán que manejar un tráfico destinado a otros. Por ejemplo, en nuestro diseño, el tráfico de 1 a 3 deberá pasar a través de 4, 5 y 6. Esto es perfectamente posible, ya que la inmensa mayoría de las implementaciones TCP/IP son capaces de reenviar datagramas. En redes de este tipo podemos pensar que los propios hosts actúan como gateways. Y, por tanto, deberíamos configurar el software de enrutamiento de los hosts como si se tratase de un gateway. Este tipo de configuraciones no es tan común como podría pensarse en un principio debido, principalmente, a estas dos razones:

* La mayoría de las grandes redes tienen más de una computadora por localización. En estos casos es menos caro establecer una red local en cada localización que establecer líneas punto a punto entre todas las computadoras;

* Las unidades de propósito especial para conectar redes son más baratas, lo que hace que sea más lógico descargar las tareas de enrutamiento y comunicaciones a estas unidades.

Por supuesto, es factible tener una red que mezcle los dos tipos de tecnologías. Así, las localizaciones con más equipos podría manejarse usando un esquema jerárquico, con redes de área local conectadas por este tipo de unidades, mientras que las localizaciones lejanas con una sola computadora podrían conectarse mediante líneas punto a punto. En este caso, el software de enrutamiento usado en las computadoras lejanas deberá ser compatible con el usado por las unidades conmutadoras, o bien tendrá que haber un gateway entre las dos partes de la red.

Las decisiones de este tipo generalmente se toman tras estudiar el nivel de tráfico de la red, la complejidad de la red, la calidad del software de enrutamiento de los hosts y la habilidad de los hosts para hacer un trabajo extra con el tráfico de la red.

Tecnología de los circuitos de conmutación

Otro enfoque alternativo al esquema jerárquico LAN/red principal es usar circuitos conmutadores en cada computadora. Realmente, estamos hablando de una variante de la técnica de las líneas punto a punto, donde ahora el circuito conmutador permite tener a cada sistema aparentar que tiene línea directa con los restantes. Esta tecnología no es usada por la mayoría de la comunidad TCP/IP debido a que los protocolos TCP/IP suponen que el nivel más bajo trabaja con datagramas aislados. Cuando se requiere una conexión continuada, el nivel superior de red la implementa usando datagramas. Esta tecnología orientada al datagrama no coincide con este sistema orientado a los circuitos de forma directa. Para poder usar esta tecnología de circuitos conmutadores, el software IP debe modificarse para ser posible construir circuitos virtuales de forma adecuada. Cuando hay un datagrama para un destino concreto se debe abrir un circuito

virtual, que se cerrará cuando no haya tráfico para dicho destino por un tiempo. Un ejemplo de este enfoque es la DDN (Defense Data Network).

El protocolo principal de esta red es el X.25. Esta red parece desde fuera una red distribuida X.25. El software TCP/IP trata de manejar la DDN mediante el uso de canales virtuales.

Técnicas similares podrían usarse con otras tecnologías de circuitos de conmutación, como, por ejemplo, ATT's DataKit, aunque no hay demasiado software disponible para llevarlo a cabo.

Redes de un solo nivel

En algunos casos, los adelantos en el campo de las redes de larga distancia pueden sustituir el uso de redes jerárquicas. Muchas de las redes jerárquicas fueron configuradas así para permitir el uso de tecnologías tipo Ethernet y otras LAN, las cuáles no pueden extenderse para cubrir más de un campus. Así que era necesario el uso de líneas serie para conectar las distintas LANs de varios lugares. Sin embargo, ahora hay tecnologías de características similares a Ethernet, pero que pueden abarcar más de un campus y, por tanto, pensar en una sola red de larga distancia que no hace uso de una estructura jerárquica.

Las principales limitaciones de este tipo de redes son cuestiones de rendimiento y flexibilidad.

Si una sola red es usada por todo el campus es muy fácil que se sobrecargue. Las redes jerárquicas pueden manejar un volumen de trabajo mucho mayor que las redes de un solo nivel. Además, el tráfico dentro de los departamentos tiende a ser mayor que el tráfico entre departamentos.

Veamos un ejemplo concreto. Supongamos que hay diez departamentos, cada uno de los cuales genera 1 Mbit/seg de tráfico. Supongamos que el 90% del tráfico se realiza entre sistemas del mismo departamento y el 10% restante hacia los demás departamentos. Si cada departamento tiene su propia red, éstas deberían ser capaces de manejar 1 Mbit/seg, al igual que la red principal que las maneja, para poder posibilitar el 10% que cada departamento destina a otros departamentos. Para resolver la misma situación con una red de un solo nivel, puesto que debe manejar simultáneamente los diez departamentos, se resuelve con una red que soporte 10 Mbit/seg.

Está claro que el ejemplo anterior está pensado para que el sistema jerárquico sea ventajoso o, al menos, que sea más fácil de llevar a cabo. Si el tráfico destinado a los otros departamentos fuese mayor, el ancho de banda de la red principal deberá ser mayor. Por ejemplo, si en un campus hay algunos recursos centralizados, como mainframes u otros grandes sistemas en un centro de cálculo. Si la mayoría del tráfico procede de pequeños sistemas que intentan comunicarse con el sistema central, entonces el argumento anterior no es válido. Aunque un enfoque jerárquico puede que todavía sea útil, sin embargo no reduce el ancho de banda requerido. Siguiendo con el ejemplo dado, si los diez departamentos se comunicasen primordialmente con los sistemas de la computadora central, la red principal deberá ser capaz de manejar 10 Mbit/seg. La computadora central debería de conectarse directamente a la red principal, o tener una red "departamental" con una capacidad de 10 Mbist/seg, en lugar de los 1 Mbit/seg de los otros departamentos.

La segunda limitación se refiere a consideraciones respecto a la fiabilidad, mantenibilidad y seguridad. Las redes de área amplia son más difíciles de diagnosticar y mantener que las redes de área local, porque los problemas pueden localizarse en el edificio donde la red se ubica.

Además, hacen que el tráfico sea más fácil de controlar. Por estas razones es más lógico manejar un tráfico local dentro del edificio y usar las redes de área amplia sólo para el tráfico entre edificios. No obstante, si se da el caso de que en cada localización hay sólo una o dos computadoras, no tiene sentido montar una red local en cada lugar y sí usar una red de un solo nivel.

Diseños mixtos

En la práctica, pocas redes se permiten el lujo de adoptar un diseño teóricamente puro. Es poco probable que una red grande sea capaz de evitar el uso de un diseño jerárquico. Supongamos que la configuramos como una red de un solo nivel. Incluso si la mayoría de los edificios tienen sólo una o dos computadoras, habrá alguna localización donde haya bastantes computadoras para justificar el uso de una red local. El resultado es una mezcla entre una red de un solo nivel y una red jerárquica. En la mayoría de los edificios sus computadoras están conectadas directamente a una red de área amplia, como una red de un solo nivel, pero en un edificio hay una red de área local usando su red de área amplia como red principal, a la cual se conecta a través de unidades conmutadoras.

Por otro lado, incluso los diseñadores de redes que defienden el uso de una enfoque jerárquico, en muchas ocasiones encuentran partes de redes donde simplemente no resulta económico instalar una red de área local, así que algunos hosts se enganchan directamente a la red principal, o bien se usa una línea serie.

Además de las razones económicas de la instalación en sí, hay que tener en cuenta que a la larga hay que valorar aspectos de mantenimiento, de manera que a veces es mejor hacer un desembolso económico en el diseño para ahorrarnos dinero en el mantenimiento futuro. Por tanto, el diseño más consistente será aquél que podamos ser capaces de mantener más fácilmente.

INTRODUCCIÓN A LAS DISTINTAS TECNOLOGÍAS DE CONMUTACIÓN

En esta sección discutiremos las características de varias tecnologías usadas para intercambiar datagramas entre redes. En efecto, trataremos de dar más detalles sobre esas "cajas negras" que hemos visto en las anteriores secciones. Hay tres tipos básicos de conmutadores, como repetidores, bridges (o puertas) y gateways (o pasarelas), o, alternativamente, switches de nivel 1, 2 y 3 (basándonos en el nivel del modelo OSI en el que operan). También hay que aclarar que hay sistemas que combinan características de más de uno de estos dispositivos, especialmente bridges y gateways.

Las diferencias más importantes entre estos tipos de dispositivos residen en el grado de aislamiento a fallos, prestaciones, enrutamiento y las facilidades que ofrecen para la administración de la red. Más adelante examinaremos esto con más detalle.

La diferencia mayor se encuentra entre los repetidores y los otros dos tipos de switches. Hasta hace relativamente poco tiempo, los gateways proporcionaban unos servicios muy distintos a los ofrecidos por los bridges, pero ahora hay una tendencia a unificar estas dos tecnologías.

Los gateways están empezando a adoptar un hardware de propósito específico que antes era característico de los bridges. Los bridges están empezando a adoptar un enrutamiento más sofisticado, características de aislamiento y de administración de redes que antes sólo se podían encontrar en los gateways. Incluso hay sistemas que pueden funcionar como bridge y gateway. Esto significa que la decisión crucial no es decidir si tenemos que usar un bridge o un gateway, sino qué características necesitamos en un switch y cómo éste afecta el diseño global de la red.

Repetidores

Un repetidor es un equipo que conecta dos redes que usan la misma tecnología. Recibe los paquetes de datos de cada red y los retransmite a la otra red. La red resultante se caracteriza por tener la unión de los paquetes de ambas redes. Para las redes Ethernet, o que cumplen el protocolo IEEE 802.3, hay dos tipos de repetidores (otras tecnologías de red no hacen estas distinciones).

Un repetidor trabaja a muy bajo nivel. Su objetivo principal es subsanar las limitaciones de la longitud del cable que provocan pérdidas de señal, dispersión temporal, etc. nos permiten construir redes más grandes y liberarnos de las limitaciones de la longitud del cable.

Podríamos pensar que un repetidor se comporta como un amplificador a ambos lados de la red, pasando toda la información contenida en la señal (incluso las colisiones) sin hacer ningún procesamiento a nivel de paquetes. No obstante, hay un número máximo de repetidores que pueden introducirse en una red. Las especificaciones básicas de Ethernet requieren que las señales lleguen a su destino dentro de un límite de tiempo, lo que determina que haya una longitud máxima de la red. Poniendo varios repetidores en el camino se introducen dificultades para estar dentro del límite (de hecho, cada repetidor introduce un retraso, así que de alguna manera se introducen nuevas dificultades).

Un "repetidor con buffer" trabaja a nivel de paquetes de datos. En lugar de pasar la información contenida en la señal, almacena paquetes enteros de una red en un buffer interno y, luego, lo retransmite a la otra red, por lo que no deja pasar las colisiones. Debido a que los fenómenos de bajo nivel, como las colisiones, no son repetidos, se puede considerar como si las dos redes continuasen separadas en lo que se refiere a las especificaciones Ethernet. Por tanto, no hay restricciones respecto al número de repetidores con buffer que se pueden usar.

De hecho, no es necesario que ambas redes sean del mismo tipo, pero han de ser suficientemente similares, de manera que tengan el mismo formato de paquete. Generalmente, esto significa que se emplean repetidores con buffer entre redes de la familia IEEE 802.x (asumiendo que elegimos la misma longitud para las direcciones y el mismo tamaño máximo para los paquetes), o entre dos redes de otra familia. Además, un par de repetidores con buffer pueden usarse para conectar dos redes mediante una línea serie.

Los repetidores con buffer y los repetidores básicos tienen una característica en común: repiten cada paquete de datos que reciben de una red en la otra. Y así ambas redes, al final, tienen exactamente el mismo conjunto de paquetes de datos.

PUENTS Y GATEWAYS

Un bridge se diferencia principalmente de un repetidor en que realiza algún tipo de selección de qué datagramas se pasan a las otras redes. Persiguen alcanzar el objetivo de aumentar la capacidad de los sistemas, al mantener el tráfico local confinado a la red donde se originan.

Solamente el tráfico destinado a otras redes será reenviado a través del bridge. Esta descripción también podría aplicarse a los gateways. Bridges y gateways se distinguen por la manera de determinar qué datagramas deben reenviarse. Un bridge usa sólo las direcciones del nivel 2 de OSI; en el caso de las redes Ethernet, o IEEE 802.x, nos referimos a las direcciones de 6 bytes de Ethernet o direcciones del nivel-MAC (el término "direcciones del nivel MAC" es más general. Sin embargo, con la intención de aclarar ideas, los ejemplos de esta sección se referirán a redes Ethernet y así sólo deberemos reemplazar el término "dirección Ethernet" por el equivalente de dirección de nivel MAC en cualquier otra tecnología). Un bridge no examina el datagrama en sí, así que no usa las

direcciones IP, o su equivalente para tomar las decisiones de enrutamiento. Como contraste, un gateway basa sus decisiones en las direcciones IP, o su equivalente en otros protocolos.

Hay varias razones por las que importa el tipo de dirección usada para tomar una decisión. La primera de ellas afecta a cómo interactúan dichos dispositivos conmutadores con los niveles superiores del protocolo. Si el reenvío se hace a nivel de las direcciones de nivel-MAC (bridge), dicho dispositivo será invisible a los protocolos. Si se hace a nivel IP, será visible.

Veamos un ejemplo en el que hay dos redes conectadas por un bridge:

Red 1
128.6.5
Red 2
128.6.4
128 .6 .5.2 128 .6 .4.3 128 .6 .4.4
ordenador A bridge ordenador B ordenador C

Hay que decir que un bridge no tiene una dirección IP. En lo que se refiere a las computadoras A, B y C, hay una sola Ethernet a la que están conectadas. Esto se traduce en que las tablas de enrutamiento deben configurarse de manera que las computadoras de ambas redes se traten como si fuesen locales. Cuando la computadora A abre una conexión con la computadora B, primero se envía una petición ARP preguntando por la dirección Ethernet de la computadora B. El bridge debe dejar pasar esta petición de la red 1 a la red 2 (en general, los bridges deben atender todas las peticiones). Una vez que ambas computadoras conocen las direcciones Ethernet del otro, las comunicaciones usarán las direcciones Ethernet en el destino. Llegados a este punto, el bridge puede empezar a ejecutar alguna selección, y dejará pasar aquellos datagramas cuya dirección Ethernet de destino se encuentren en una máquina de la otra red.

De esta manera un datagrama desde A hasta B pasará de la red 2 a la red 1, pero un datagrama desde B hasta C se ignorará.

Con objeto de hacer esta selección, el bridge necesita saber en qué red está cada máquina. La mayoría de los bridges modernos construyen una tabla para cada red a la que se conecta, listando las direcciones Ethernet de las máquinas de las que se sabe en qué red se encuentran, y para ello vigilan todos los datagramas de cada red. Cuando un datagrama aparece primero en la red 1 es razonable pensar que la dirección del remitente corresponde a una máquina de la red 1.

Un bridge debe examinar cada datagrama por dos razones: la primera, para usar la dirección de procedencia y aprender qué máquinas están en cada red, y, la segunda, para decidir si el datagrama ha de ser reenviado o no en base a la dirección de destino.

Como mencionamos anteriormente, por regla general los bridges dejan pasar las peticiones de una red a la otra. En varias ocasiones hay peticiones para localizar algún recurso. Una petición ARP es un típico ejemplo de lo anterior. Debido a que un bridge no tiene manera de saber si un host va a responder a dicha petición, deberá dejarla pasar a la otra red. Algunos bridges tienen filtros definidos por el usuario, que les posibilita dejar pasar algunos y bloquear a otros.

Podemos permitir peticiones ARP (que son esenciales para que el protocolo IP funcione) y restringir otras peticiones menos importantes a su propia red de origen. Por ejemplo, podemos elegir no dejar pasar las peticiones rwhod, que usan algunos sistemas para conocer los usuarios conectados en cada sistema, o podemos decidir que sólo pueda tener acceso a una parte de la red.

Ahora veamos un ejemplo de dos redes conectadas por un gateway:

Red 1
Red 2
128.6.5 128.6.4
128 .6 .5.1
128 .6 .4.1
128 .6 .5.2
ordenador A
gateway
128 .6 .4.3 128 .6 .4.4
ordenador B ordenador C

Los gateways tienen asignada una dirección IP por cada interfase. Las tablas de enrutamiento de las computadoras deberán configurarse para hacer los envíos a las direcciones adecuadas.

Así, por ejemplo, la computadora A tienen una entrada estableciendo que debe usarse el gateway 128.6.5.1 para alcanzar la red 128.6.4.

Debido a que las computadoras tienen conocimiento de la existencia del gateway, el gateway no necesita inspeccionar todos los paquetes de la Ethernet. Las computadoras le enviarán datagramas cuando sea apropiado. Por ejemplo, supongamos que la computadora A necesita enviar un mensaje a la computadora B. En la tabla de enrutamiento de A se indica que deberemos usar el gateway 128.6.5.1, y entonces se le enviará una petición ARP para esa dirección, respondiéndonos el gateway a la petición como si se tratase de un host cualquiera. A partir de entonces, los datagramas destinados a B serán enviados con la dirección Ethernet del gateway.

Más sobre bridges

Hay varias ventajas para usar direcciones del nivel MAC, como lo hace un bridge. La primera es que cada paquete en una Ethernet, o en una red IEEE, usa dichas direcciones, y la dirección se localiza en el mismo lugar en cada paquete, incluso si es IP, DECnet, o de cualquier otro protocolo. De tal manera que es relativamente rápido obtener la dirección de cada paquete. Por otro lado, un gateway debe decodificar toda la cabecera IP y, si soporta otros protocolos distintos a IP, debe tener un software distinto para cada protocolo. Esto significa que un bridge soporta automáticamente cualquier protocolo posible, mientras que un gateway debe prever qué protocolo debe soportar.

Sin embargo, también hay desventajas. La principal se refiere al diseño de un puente

* Un puente debe mirar cada paquete de la red, no solo aquéllos a los que se le destinan.

Esto hace posible que haya sobrecargas en el bridge si se coloca en una red muy concurrida, incluso si el tráfico que atraviesa el bridge es pequeño. No obstante, existe otra desventaja basada en la manera como los bridges están diseñados. Sería posible, en principio, diseñar bridges sin estas desventajas, pero no hay indicios de que se cambie. La desventaja se debe al hecho de que los bridges no tienen una tabla de enrutamiento completa con todos los sistemas de las redes, ya que sólo tienen una simple lista con las direcciones Ethernet que se encuentran en sus redes. Lo que significa que

* Las redes que usan bridges no pueden tener bucles en su diseño. Si hubiera un bucle, algunos bridges verían el tráfico procedente de una misma dirección Ethernet venir de ambas direcciones, por lo que le sería imposible decidir en qué tabla debe poner dicha dirección. Hay que aclarar que un camino paralelo

en la misma dirección constituye un bucle y, por tanto, no se podrán usar múltiples caminos con el fin de descargar el tráfico de la red.

Hay algunos métodos para afrontar el problema de los bucles. Muchos puentes permiten configuraciones con conexiones redundantes, pero desactivando enlaces de manera que no haya bucles. Si un enlace falla, uno de los desactivados entra en servicio. Así, los enlaces redundantes nos proporcionan una fiabilidad extra, pero nos proporcionan nuevas capacidades.

También es posible construir un bridge capaz de manejar líneas punto a punto paralelas, en un caso especial donde dichas líneas tienen en sus extremos un bridge. Los bridges tratarían las dos líneas como una única línea virtual y usarlas alternativamente, siguiendo algún algoritmo aleatorio.

El proceso de desactivar conexiones redundantes hasta que no queden bucles es conocido como un "algoritmo de expansión de árboles". Este nombre se debe a que un árbol se define como un patrón de conexiones sin bucles. Lo que se hace es ir desactivando conexiones, ya que las conexiones restantes en el árbol incluyen a todas las redes del sistema. Para llevarlo a cabo, todos los bridges del sistema de redes deben comunicarse entre ellos.

Hay una tendencia a que los árboles de expansión resultantes cargan demasiado a la red en alguna parte del sistema. Las redes cercanas a la "raíz del árbol" manejan todo el tráfico entre las distintas partes de la red. En una red que usa gateways, sería posible poner enlaces extras entre partes de la red que tengan un gran tráfico, pero dichos enlaces extras no pueden ser usados por un conjunto de bridges.

Más sobre gateways

Los gateways tienen sus propias ventajas y desventajas. En general, un gateway es más complejo de diseñar y administrar que un bridge. Un gateway debe participar en todos los protocolos para los que está diseñado para reenviar. Por ejemplo, un gateway IP debe responder a peticiones ARP. El estándar IP también necesita estudiar por completo las cabeceras IP, decrementando el tiempo para activar campos y obedecer cualquier opción IP.

Los gateways son diseñados para manejar topologías de redes más complejas que las que son capaces de manejar los bridges. Y, como ya hemos mencionado, tienen diferentes (y más complejas) decisiones que estudiar. En general, un bridge tiene decisiones más fáciles que tomar: si se debe reenviar un datagrama y, en caso de que deba hacerse, qué interfase hemos de elegir. Cuando un gateway reenvía un datagrama, debe decidirse a qué host o gateway hay que enviarlo a continuación. Si un gateway envía un datagrama de vuelta a la red de donde procede, también debe enviar una redirección al emisor del datagrama indicando que use una mejor ruta. Muchos gateways pueden también manejar caminos paralelos. Si hay varios caminos igualmente buenos para un destino, el gateway usará uno de ellos determinado por algún tipo de algoritmo aleatorio (esto se hace también en algunos bridges, pero no suele ser lo usual. En ambos casos, se elige uno de ellos mediante algún tipo de algoritmo aleatorio. Esto tiende a hacer que la llegada de los datagramas tenga un orden distinto al que fueron enviados. Lo que puede complicar la labor de procesamiento de los datagramas de los hosts de destino, e, incluso, hay viejas implementaciones TCP/IP que tienen errores a la hora de ordenar los datagramas).

Para poder analizar todas estas decisiones, un gateway tendrá una tabla de enrutamiento muy similar a la de los hosts. Al igual que las tablas de enrutamiento, las tablas de los gateways contienen una entrada por cada posible número de red. Para cada red hay, o bien una entrada indicando que la red está conectada

directamente al gateway, o hay una entrada indicando que el tráfico para esa red debe reenviarse hacia algún otro gateway o gateways. Describiremos posteriormente los "protocolos de enrutamiento" usados para elaborar esta información, en la discusión sobre cómo configurar un gateway.

COMPARANDO LAS TECNOLOGÍAS DE CONMUTACIÓN

Los repetidores, repetidores con buffer, bridges y gateways forman un espectro. Los dispositivos del principio de la lista son mejores para redes pequeñas, además son más baratos y fáciles de configurar aunque tienen menos servicios. Los del final de la lista son apropiados para construir redes más complejas. Muchas redes usan mezclas de dispositivos, con repetidores para conectar pequeños segmentos de red, bridges para algunas áreas grandes y gateways para enlaces de larga distancia.

Hasta ahora hemos asumido que sólo usan gateways. La sección de cómo configurar un host describe cómo configurar una tabla de enrutamiento, listando los gateways que se debían usar para alcanzar a distintas redes. Los repetidores y bridges son invisibles a IP, y, en lo que a las anteriores secciones se refiere, las redes conectadas mediante ellos se deben considerar como una única red. Más adelante, se describe cómo configurar un host en el caso en que varias subredes se traten como una única red física; la misma configuración debería usarse cuando varias subredes se conectan mediante repetidores o bridges.

Como ya mencionamos, las características a tener en cuenta en un dispositivo conmutador son: aislamiento, rendimiento, enrutamiento y las facilidades de mantenimiento de la red.

Aislamiento

Generalmente, los dispositivos conmutadores se usan para conectar redes. Así que, normalmente, pensamos en ganar conectividad, no en el aislamiento. Sin embargo, el aislamiento es algo digno de tener en cuenta. Si conectamos dos redes y no tenemos en cuenta el aislamiento para nada, entonces cualquier problema en otras redes aparecerá en la nuestra también. Asimismo, dos redes juntas pueden tener suficiente tráfico como para saturar la nuestra. Es por lo tanto conveniente elegir un nivel apropiado de protección.

El aislamiento puede llegar de dos maneras: aislamiento frente al mal funcionamiento y frente al tráfico. Con el objeto de discutir el aislamiento debido a errores de funcionamiento, vamos a señalar una clasificación de malfunciones:

* Fallos eléctricos, como por ejemplo una bajada de tensión o algún tipo de fallo que distorsiona la señal. Todos los tipos de dispositivos deberán confinarlo a un lado del dispositivo (repetidor, repetidor con buffer, bridge, gateway).
* Problemas con los transceiver y controladores que, en general, generan señales eléctricamente correctas, pero de contenido erróneo (por ejemplo, paquetes de tamaño infinito o demasiado grandes, falsas colisiones, portadora continua). Todos, excepto el repetidor, nos protegen de estos problemas, que no son muy comunes.
* Errores en el software que provocan un excesivo tráfico entre algunos hosts (no nos referimos a mensajes de tipo broadcoast). Los bridges y gateways pueden aislarnos de estos errores. (Este tipo de fallos son bastante raros. La mayor parte de los problemas del software y de protocolos generan broadcoasts).
* Errores en el software que provocan un excesivo tráfico de broadcast. Los gateways se aíslan de estos problemas. Generalmente, los bridges no lo hacen, porque deben dejar las peticiones ARP y otros broadcasts. Los bridges con filtros definidos por el usuario podrían protegernos contra algunos de

estos errores de sobrecarga de broadcast. Sin embargo, en general, los bridges deben dejar pasar ARP y la mayoría de estos errores se deben a ARP. Este problema no es tan grave en redes donde el software tiene un cuidadoso control, pero tendremos regularmente problemas de este tipo en redes complejas o con software experimental.

El aislamiento al tráfico es proporcionado por bridges y gateways. La decisión más importante al respecto es conocer el número de computadoras que podemos poner en una red sin sobrecargarla. Esto requiere el conocimiento de la capacidad de la red, y el uso al que se destinarán los hosts. Por ejemplo, una Ethernet puede soportar cientos de sistemas si se van a destinar para logins remotos y, ocasionalmente, para transferencia de archivos. Sin embargo, si las computadoras carecen de disco, y usamos la red para swapping, una Ethernet podría soportar entre 10 y 40, dependiendo de su velocidad y sus características de E/S.

Cuando ponemos más computadoras en una red de los que es capaz de manejar, deberemos dividirla en varias redes y poner algún dispositivo conmutador entre ellos. Si esto se hace correctamente, la mayoría del tráfico deberá realizarse entre máquinas de la misma parte de la división, lo que significa poner los clientes en la misma red que su servidor, poner los servidores de terminales en la misma red que los hosts a los que se accede más frecuentemente.

Bridges y gateways, generalmente, suministran el mismo grado de aislamiento al tráfico. En ambos casos, sólo el tráfico destinado a los hosts del lado de la unidad conmutadora se pasará.

Veremos esto más detalladamente en la sección del enrutamiento.

Prestaciones

Los límites de las prestaciones empiezan a ser menos claros, puesto que las tecnologías de conmutación están mejorando continuamente. Generalmente, los repetidores pueden manejar todo el ancho de banda de la red (por su propia naturaleza, un repetidor básico ha de ser capaz de hacer esto). Los bridges y gateways frecuentemente tienen limitaciones en sus prestaciones de varios tipos. Los bridges tienen dos estadísticos de interés: la tasa de paquetes analizados y el rendimiento.

Como explicamos anteriormente, los bridges deben analizar cada paquete que se encuentra en la red, incluso aquellos que no van a ser reenviados. El número de paquetes analizados por segundo es la unidad usada para medir la tasa de paquetes analizados. El rendimiento se puede aplicar tanto a bridges como a gateways, y refleja la parte del tráfico que ha sido reenviada; generalmente, depende del tamaño del datagrama. Así, el número de datagramas por segundo que una unidad puede manejar será mayor cuanto haya más datagramas pequeños que grandes.

Normalmente, un bridge puede manejar desde algunos cientos de datagramas hasta unos 7.000. Se puede obtener mayor capacidad de procesamiento con equipos que usan una hardware de propósito específico para acelerar la labor de análisis de paquetes. La primera generación de gateways podían procesar entre algunos cientos de datagramas por segundo hasta unos 1.000 ó más; sin embargo, los gateways de segunda generación, ampliamente extendidos, usan un hardware de propósito específico igual de sofisticado que el usado en los bridges y con ellos se pueden manejar alrededor de 10.000 datagramas por segundo.

Debido a que en este momento los bridges y gateways de altas prestaciones pueden manejar casi todo el ancho de banda de una Ethernet, las prestaciones no son una razón para elegir entre un tipo u otro de dispositivo. Sin embargo, para un

tipo dado de dispositivo, hay todavía grandes diferencias entre los distintos modelos, sobre todo en la relación precio/prestaciones.

Esto es especialmente cierto en los modelos de la gama baja. Los bridges más baratos cuestan menos de la mitad que los gateways más baratos.

Desgraciadamente, no hay un único estadístico para poder estimar las prestaciones de un dispositivo. No obstante, el que más se usa es el de paquetes por segundo. Hay que tener en cuenta que la mayoría de las empresas cuentan los datagramas una sola vez, cuando pase por el gateway; hay una compañía importante que cuenta los datagramas 2 veces, y, por tanto, deben dividirse por 2 para poder comparar. También hay que asegurarse, para hacer una comparación correcta, que los datagramas son del mismo tamaño. Un modelo para poder comparar prestaciones es tiempo_de_procesamiento = tiempo_conmutación + tamaño_datagrama * tiempo_por_byte.

Aquí, el tiempo de conmutación suele ser una constante; representa la interrupción latente, el procesamiento de las cabeceras, buscar en la tabla de enrutamiento, etc., más un componente proporcional al tamaño del datagrama, representando el tiempo necesario para hacer cualquier copia de datagrama. Un enfoque razonable para estudiar las prestaciones es dar los datagramas por segundo por los tamaños mínimos y máximos de los datagramas. Otra forma de conocer los límites de un dispositivo es conociendo la velocidad de los datagramas por segundo y el rendimiento en bytes por segundo, y aplicando la fórmula anterior.

Enrutamiento

Vamos a estudiar las tecnologías usadas para decidir hacia dónde debe enviarse un datagrama.

Por supuesto, no haremos esto para los repetidores, ya que éstos reenvían todos los paquetes.

La estrategia de enrutamiento de un bridge conlleva tomar dos decisiones:

1.- Activar o desactivar los enlaces de manera que se mantenga el árbol de expansión y;
2.- Decidir si debemos reenviar un paquete en particular y a través de cuál interfase (si el puente es capaz de manejar más de dos interfases).

La segunda decisión se toma en base a una tabla de direcciones del nivel-MAC. Como ya hemos descrito anteriormente, esta tabla se construye analizando el tráfico que pasa por cada interfase. El objetivo es reenviar aquellos paquetes cuyo destino se encuentre a otro lado del bridge. Este algoritmo requiere tener una configuración de red que no contenga bucles o líneas redundantes. Los bridges menos sofisticados dejan esta tarea al diseñador de la red, y debemos diseñar y configurar una red sin bucles. Los bridges más sofisticados permiten una topología cualquiera, pero irá desactivando enlaces hasta que no haya bucles; además, nos proporciona una fiabilidad extra, ya que, en caso de fallo de un enlace, se activará automáticamente un enlace alternativo. Los bridges que funcionan de este modo tienen un protocolo que les permite detectar cuándo una unidad debe desactivarse o activarse, de manera que el conjunto activo de enlaces abarquen el árbol de expansión. Si necesitamos la fiabilidad proporcionada por los enlaces redundantes, debemos asegurarnos que nuestros bridges sean capaces de trabajar de esta manera. Actualmente no hay un protocolo estándar para este tipo de bridges, pero está en camino. En caso de comprar bridges de más de una marca, debemos asegurarnos que sus protocolos para trabajar con los árboles de expansión pueden entenderse.

Por otro lado, los gateways permiten cualquier tipo de topología, incluyendo bucles y enlaces redundantes. Debido a que tienen algoritmos más generales de

enrutamiento, los gateways deben mantener un modelo de toda la red. Diferentes técnicas de enrutamiento mantienen modelos de redes con más o menos complejidad, y usan esta información con distinto tipo de sofisticación. Los gateways que pueden manejar IP, normalmente soportan los dos protocolos estándares de Internet: RIP (Routing Information Protocol) y EGP (External Gateway Protocol). El EGP es un protocolo de propósito específico usado en redes donde hay una red principal, y permite intercambiar información de "cómo llegar" con la red principal. Por regla general, es bastante recomendable que nuestros gateways soporten EGP.

RIP es un protocolo diseñado para manejar rutas en redes pequeñas o medianas, donde la velocidad de las líneas no difieren demasiado. Sus principales limitaciones son:

* No puede usarse con redes donde los caminos pasan por más de 15 gateways. Se puede, incluso, reducir este número en el caso de que usemos una opción de dar un paso mayor de una a otra línea lenta.
* No puede compartir el tráfico entre líneas paralelas (algunas implementaciones permiten hacer esto si dichas líneas se encuentran entre el mismo par de gateways).
* No puede adaptarse a la sobrecarga de redes.
* No es adecuada para situaciones en las que hay rutas alternativas a través de líneas con muy distinta velocidad.
* No es estable en redes donde las líneas o los gateways cambian con frecuencia.

Algunas compañías venden modificaciones de RIP que mejoran su funcionamiento con EGP, o que incrementan la longitud del camino máximo más allá de 15, pero no incluyen otro tipo de modificaciones. En caso de que nuestra red disponga de gateways de más de una marca, en general necesitaremos que soporten RIP, puesto que suele ser el único protocolo de enrutamiento disponible. Si vamos a

trabajar, además, con otro tipo de protocolo, pueden sernos útiles gateways que traduzcan su propio protocolo y RIP. Sin embargo, para redes muy grandes o complejas no nos queda otro remedio que usar otros protocolos.

También existen otros protocolos más sofisticados. Los principales son IGRP y los basados en los algoritmos SPF (el camino más corto primero - short path fist). Usualmente, estos protocolos han sido diseñados para redes más grandes o complejas y, en general, son estables bajo una gran variedad de condiciones, pudiendo manejar líneas de cualquier velocidad.

Algunos de ellos permiten tener en cuenta la sobrecarga de algunos caminos, pero hasta el momento no conozco un gateway que sea capaz de hacer esto (hay serios problemas para mantener un enrutamiento estable para realizarlo). Hay numerosas variantes de tecnologías de enrutamiento, y éstas se están modificando rápidamente, así que deberemos tener en cuenta la topología de nuestra red para elegir un producto en concreto; tenemos que asegurarnos que puede manejar nuestra topología y que puede soportar otros requerimientos especiales, como compartir el tráfico entre líneas paralelas, o ajustar la topología ante fallos. A largo plazo, se espera que aparezcan nuevos protocolos que estandaricen estos trabajos. Pero, por el momento, no se usa otra tecnología de enrutamiento que la RIP.

Otro asunto concerniente al enrutamiento es la política en la que se basa el enrutamiento. En general, los protocolos de enrutamiento pretenden encontrar el camino más corto o más rápido posible para cada datagrama. En algunos casos, esto no es lo deseable; a veces, por razones de seguridad, razones económicas, etc, puede que deseemos reservar algunos caminos para algún uso específico. La mayoría de los gateways tienen la capacidad de controlar la propagación de la información de enrutamiento, lo que nos da algunas facilidades de administración sobre la forma en que estas rutas se usan, y el grado de control que soportan varía de un gateway a otro.

Administración de Redes

La administración de redes abarca un amplio número de asuntos. En general, se suelen tratar con muchos datos estadísticos e información sobre el estado de distintas partes de la red, y se realizan las acciones necesarias para ocuparse de fallos y otros cambios. La técnica más primitiva para la monitorización de una red es hacer "pinging" a los hosts críticos; el "pinging" se basa en un datagrama de "echo" (eco), que es un tipo de datagrama que produce una réplica inmediata cuando llega al destino. La mayoría de las implementaciones TCP/IP incluyen un programa (generalmente, llamado "ping") que envía un echo a un host en concreto. Si recibimos réplica, sabremos que host se encuentra activo, y que la red que los conecta funciona; en caso contrario, sabremos que hay algún error. Mediante "pinging" a un razonable número de ciertos hosts, podremos normalmente conocer qué ocurre en la red. Si los ping a todos los hosts de una red no dan respuesta, es lógico concluir que la conexión a dicha red, o la propia red, no funciona. Si sólo uno de los hosts no da respuesta, pero los demás de la misma red responden, es razonable concluir que dicho host no funciona.

Técnicas más sofisticadas de monitorización necesitan conocer información estadística y el estado de varios dispositivos de la red. Para ello necesitará llevar la cuenta de varias clases de datagramas, así como de errores de varios tipos. Este tipo de información será más detallada en los gateways, puesto que el gateway clasifica los datagramas según protocolos e, incluso, él mismo responde a ciertos tipos de datagramas. Sin embargo, los bridges e incluso los repetidores con buffer contabilizan los datagramas reenviados, errores de interfase. Es posible recopilar toda esta información en un punto de monitorización central.

También hay un enfoque oficial TCP/IP para llevar a cabo la monitorización. En la primera fase, usamos un conjunto de protocolos SGMP y SNMP, ambos

diseñados para permitirnos recoger información y cambiar los parámetros de la configuración y otras entidades de la red.

Podemos ejecutar los correspondientes programas en cualquier host de nuestra red. SGMP está disponible para varios gateways comerciales, así como para sistemas Unix que actúan como gateway. Cualquier implementación SGMP necesita que se proporciones un conjunto de datos para que pueda empezar a funcionar, y tienen mecanismos para ir añadiendo informaciones que varían de un dispositivo a otro. A finales de 1988 apareció una segunda generación de este protocolo, SNMP, que es ligeramente más sofisticado y necesita más información para trabajar y, para ello, usa el llamado MIB (Management Information Base). En lugar de usar una colección de variable SNMP, el MIB es el resultado de numerosas reuniones de Comités formados por vendedores y usuarios. También se espera la elaboración de un equivalente de TCP/IP de CMIS, el servicio ISO de monitorización de redes. Sin embargo, CMIS y sus protocolos, CMIP, todavía no son estándares oficiales ISO, pero están en fase experimental.

En términos generales, todos estos protocolos persiguen el mismo objetivo: permitirnos recoger información crítica de una forma estandarizada. Se ordena la emisión de datagramas UDP desde un programa de administración de redes que se encuentra ejecutando en alguno de los hosts de red. Generalmente, la interacción es bastante simple, con el intercambio de un par de datagramas: una orden y una respuesta. El mecanismo de seguridad también es bastante simple, siendo posible que se incluyan passwords en las órdenes (en SGMP nos referiremos a éstos como una "session name", en lugar de password). También existen mecanismos de seguridad más elaborados, basados en la criptografía.

Probablemente querremos configurar la administración de la red con las herramientas que tenemos a nuestra disposición para controlar diversas actividades. Para redes con pocas terminales, queremos controlar cuándo nuestros dispositivos de conmutación fallan, están fuera de servicio por

mantenimiento, y cuando haya fallos en las líneas de comunicación u otro hardware. Es posible configurar SGMP y SNMP para que usen "traps" (mensajes no solicitados) para un host en particular o para una lista de hosts cuando ocurre un evento crítico (por ejemplo, líneas activas o desactivas). No obstante, no es realista esperar que un dispositivo de conmutación nos notifique cuando falla. También es posible que los mensajes "traps" se pierdan por un fallo en la red, o por sobrecarga, así que no podemos depender completamente de los traps. No obstante, es conveniente que nuestros dispositivos de conmutación reúnan regularmente este tipo de información. Hay varias herramientas que visualizan un mapa de la red, donde los objetos cambian de color cuando cambian de estado, y hay cuadros que muestran estadísticas sobre los datagramas y otros objetos.

Otro tipo de monitorización deseable es recolectar información para hacer informes periódicos del porcentaje de uso de la red y prestaciones. Para ello, necesitamos analizar cada dispositivo de conmutación y quedarnos con los datos de interés. Hay informes mensuales en los que se refleja el tráfico que soporta cada gateway y algunas estadísticas de errores, elegidas para ver si hay un gateway que está sobrecargado (datagramas perdidos).

Sería posible que cualquier tipo de conmutador pudiese usar cualquier tipo de técnica de monitorización. Sin embargo, generalmente los repetidores no proporcionan ningún tipo de estadística, debido a que normalmente no tienen ningún procesador para abaratar su precio.

Por otro lado, es posible usar un software de administración de redes con repetidores con buffer, bridges y gateways. Los gateways, en la mayoría de los casos, incluyen un avanzado software de administración de redes. La mayoría de los gateways pueden manejar IP y los protocolos de monitorización anteriormente mencionados. Y la mayoría de los bridges tienen medios para poder recoger algunos datos de prestaciones. Puesto que los bridges no están dirigidos a ningún protocolo en particular, la mayoría de ellos no tienen el software necesario para

implementar los protocolos TCP/IP de administración de redes. En algunas ocasiones, la monitorización puede hacerse tecleando algunos comandos a una consola a la que esté directamente conectada (hemos visto un caso donde era necesario dejar el puente fuera de servicio para recoger estos datos). En los restantes casos, es posible recoger datos a través de la red, pero el protocolo requerido no suele ser ningún estándar.

Excepto para algunas pequeñas redes, debemos insistir en que cualquier dispositivo conmutador más complejo que un simple repetidor es capaz de recolectar estadísticas y algún mecanismo para hacernos con ellas de forma remota. Aquellas partes de la red que no soporten dichas operaciones pueden monitorizarse mediante pinging (aunque el ping sólo detecta errores graves, y no nos permite examinar el nivel de ruido de una línea serie y otros datos necesarios para llevar a cabo un mantenimiento de alta calidad). Se espera que la mayoría del software disponible cumpla los protocolos SGMP/SNMP y CMIS. También un software de monitorización no estándar, siempre y cuando sea soportado por los equipos que tenemos.

Una evaluación final

Vamos a reunir todo lo anterior indicando dónde se usa cada tipo de conmutador, normalmente:

* Los repetidores, normalmente, se restringen a un solo edificio. Puesto que no nos proveen de un aislamiento al tráfico, debemos asegurarnos en todas las redes conectadas por los repetidores que pueden hacer llegar a todas sus computadoras.

Puesto que no suelen tener herramientas de monitorización, no será deseable su uso para aquellos enlaces que fallan a menudo.

* Los bridges y gateways deben situarse de manera que se divida la red en partes cuyo volumen de tráfico sea manejable. Incluso se podrían emplazar bridges o gateways incluso en el caso de que no sean necesarios por razones de monitorización.
* Debido a que los bridges deben dejar pasar datagramas de broadcast, hay un límite en el tamaño de las redes que pueden conectar. Por lo general, basta limitar estas redes con un ciento de máquinas, aproximadamente. Este número puede ser mayor, si el bridge incluye algunas facilidades de filtrado de datagramas.
* Debido a que algunos tipos de redes son proclives al mal funcionamiento, deberemos usar los bridges sólo entre partes de la red donde un solo grupo es responsable de diagnosticar los problemas. Debemos estar locos para usar un bridge para conectar redes que pertenecen a distintas organizaciones. Las partes de la red "de tipo experimental" deberán aislarse del resto de la red por gateways.
* En muchas aplicaciones es más importante elegir un producto con la adecuada combinación de prestaciones, herramientas de administración de la red y otras características, para tomar la decisión de elegir entre bridges y gateways.

CONFIGURANDO GATEWAYS

Vamos a ver algunos aspectos específicos de la configuración de gateways. Aquellos gateways que entienden el protocolo IP son, al mismo tiempo, hosts de Internet y, por tanto, podemos poner en práctica lo visto para configurar las direcciones y el enrutamiento en los hosts. No obstante, la forma exacta de cómo debemos configurar un gateway depende del modelo en concreto. En algunos casos, deberemos editar algunos archivos incluidos en un disco del propio gateway. Sin embargo, por razones de fiabilidad, la mayoría de los gateways no

tienen discos propios; en su lugar, esta información se almacena en una memoria no volátil o en archivos que se cargan desde uno o varios hosts de la red.

Como mínimo, para configurar el gateway hay que especificar la dirección IP y la máscara de cada interfase, y activar un protocolo de enrutamiento adecuado. Normalmente será deseable configurar otros parámetros.

Un parámetro importante a tener en cuenta es la dirección de broadcast. Como explicamos con anterioridad, hay cierto software antiguo que no funciona bien cuando se envían broadcasts usando los nuevos protocolos de direcciones de broadcast. Por esta razón, algunos modelos nos permiten elegir una dirección broadcast para cada interfase, y se debe configurar teniendo en cuenta el número de computadoras que hay en la red. En general, si las computadoras soportan los actuales estándares, podrá usarse una dirección del tipo 255.255.255.255. No obstante, antiguas implementaciones deben comportarse mejor con otro tipo de direcciones, especialmente con aquellas direcciones que usan ceros para los números del host (para la red 128.6 éste tendría que ser 128.6.0.0. Para mantener la compatibilidad con redes que no soportan subredes podríamos usar 128.6.0.0 como dirección de broadcast, incluso para una subred como 128.6.4). Podemos observar la red con un monitor de red y ver los resultados de las distintas elecciones de direcciones de broadcast; en caso de que hagamos una mala elección, cada vez que hagamos un broadcast para ponerse al día del enrutamiento, muchas máquinas de nuestra red nos responderán con errores ARP o ICMP. Hay que hacer notar que cuando cambiamos las direcciones de broadcast en el gateway, necesitaremos cambiarla en las computadoras individuales también. Lo que se suele hacer es cambiar la dirección de aquellos sistemas que podemos configurar, para hacerlos compatibles con los otros sistemas que no pueden configurarse.

Hay otros parámetros de la interfase que pueden que sea necesario configurar para trabajar con las peculiaridades de la red a la que se conectan. Por ejemplo,

muchos gateways comprueban sus interfases a Ethernet para asegurarse de que el cable al que se conectan y el transceiver funcionan correctamente. Algunos de estos tests no funcionan correctamente con la antigua versión 1 de transceiver Ethernet. En caso de que usemos un transceiver de este tipo, deberemos desactivar este tipo de test. De forma similar, los gateways conectados a líneas en serie normalmente hacen este tipo de test para verificar su buen funcionamiento, y también hay situaciones en las que necesitaremos de activar el test.

Es bastante usual que tengamos que activar las opciones necesarias para el software que tengamos que usar. Por ejemplo, muchas veces es necesario activar explícitamente el protocolo de administración de red, y dar el nombre o la dirección del host donde se ejecuta el software que acepta traps (mensajes de error).

La mayoría de los gateways tienen opciones relacionadas con la seguridad. Como mínimo, hay que indicar un password para poder hacer cambios de forma remota (y una "sesión name" para SGMP). Si queremos controlar el acceso a ciertas partes de la red, también deberemos definir una lista de control de accesos, o cualquier otro mecanismo que use el gateway en cuestión.

Los gateways cargan la información de la configuración a través de la red. Cuando un gateway arranca, envía una petición broadcast de varias clases, intentando conocer su dirección Internet para luego cargar su configuración. Así, hay que asegurarse que haya algunas computadoras capaces de responder a dichas peticiones. En algunos casos, hay algún micro dedicado ejecutando un software especial. Otras veces, hay un software genérico que podemos ejecutar en varias máquinas. Por razones de fiabilidad, debemos comprobar que hay más de un host con la información y los programas que necesita. En algunos casos tendremos que mantener varios archivos distintos. Por ejemplo, existen gateways que utilizan un programa llamado "bootp" para que le proporcione su dirección Internet, y luego cargan el código y la información de la configuración usando TFTP. Esto significa que tenemos que mantener un archivo para "bootp" que contiene las

direcciones Ethernet e Internet para cada gateway, y un conjunto de archivos para la restante información de cada uno de ellos. Si una red es muy grande, podemos tener problemas para asegurarnos de que esta información permanece consistente. Podemos mantener copias nuestras de todas las configuraciones en una única computadora y que se distribuya a otros sistemas cuando haya algún cambio, usando las facilidades make y rdist de Unix. Si nuestro gateway tiene la opción de almacenar la información de la configuración en una memoria no volátil, podremos eliminar todos estos problemas logísticos, pero presenta sus propios problemas. El contenido de esta memoria debería almacenarse en alguna localización central, porque de todas maneras es difícil para el personal de administración de la red revisar la configuración si se encuentra distribuida entre los distintos gateways.

Arrancar un gateway que carga la información de su configuración desde una localización distante es especialmente arriesgado. Los gateways que necesitan cargar su información de configuración a través de la red, generalmente emiten una petición broadcast a todas las redes que conectan. Si alguna computadora de una de esas redes es capaz de responder, no habrá ningún problema. Sin embargo, algunos gateways que se encuentren a gran distancia donde las computadoras de su alrededor no soportan los protocolos necesarios, en cuyo caso es necesario que las respuestas le lleguen a través de una red donde haya unas computadoras apropiadas.

Desde un punto de vista estricto, esto va en contra de la filosofía de trabajo de los gateways, ya que normalmente un gateway no permite que un broadcast procedente de una red pase a través de una red adyacente. Para permitir que un gateway obtenga información de una computadora en una red distinta, al menos uno de los gateways que está entre ellos tendrá que configurarse para que pase una clase especial de broadcast usado para recuperar este tipo de información. Si tenemos este tipo de configuración, tendremos que comprobar este proceso periódicamente, ya que no es raro que nos encontremos con que no podamos

arrancar un gateway tras un fallo de energía, debido a un cambio en la configuración en otro gateway que hace imposible cargar esta información.

CONFIGURANDO EL ENRUTAMIENTO DE LOS GATEWAYS

Por último, vamos a tratar cómo configurar el enrutamiento. Este tipo de configuración es más difícil para un gateway que para un host. La mayoría de los expertos TCP/IP recomiendan dejar las cuestiones de enrutamiento a los gateways. Así, los hosts simplemente tienen una ruta por defecto que apunta al gateway más cercano (por supuesto, los gateways no pueden configurarse de esta manera. Ellos necesitan tablas completas de enrutamiento).

Para entender cómo configurar un gateway, vamos a examinar con un poco más de detalle cómo los gateways se comunican las rutas.

Cuando encendemos un gateway, la única red de la que tiene información es aquélla a la que esté directamente conectado (lo que se especifica en la configuración). Para llegar a saber cómo se llega a partes más distantes de la red, marca algún tipo de "protocolo de enrutamiento", que simplemente es un protocolo que permite a cada gateway anunciar a qué redes tiene acceso, y extender esa información de un gateway a otro. Eventualmente, cada gateway debería saber cómo llegar a cada red. Hay distintos tipos de protocolos de enrutamiento; en el más común, los gateways se comunican exclusivamente con los más cercanos; en otra clase de protocolos, cada gateway construye una base de datos describiendo cada gateway del sistema. No obstante, todos estos protocolos encuentran cómo llegar a cualquier destino.

Una métrica es un número, o conjunto de números, usado para comparar rutas. La tabla de enrutamiento se construye recogiendo información de otros gateways. Si dos gateways son capaces de llegar a un mismo destino, debe de haber algún

método para decidir cuál usar. La métrica se usa para tomar esta decisión. Todas las métricas indican de alguna forma lo "costoso" de una ruta. Podría ser cuántos dólares costaría enviar un datagrama por una ruta, el retraso en milisegundos, o cualquier otra medida. La métrica más simple es el número de gateways que hay hasta el destino ("cuenta de saltos"), y es la que generalmente se encuentra en los archivos de configuración.

Como mínimo, una configuración de enrutamiento consistiría en un comando para activar el protocolo de enrutamiento que vayamos a usar. La mayoría de los gateways están orientados para usar un protocolo; a no ser que tengamos razones para usar otro, es recomendable usar dicho protocolo. Una razón habitual para elegir otro protocolo es para hacerlo compatible con otros gateways. Por ejemplo, si nuestra red está conectada a una red nacional que nos exige usar EGP ("exterior gateway protocol") para que se pueda intercambiar rutas con ella, EGP sólo es apropiado para este caso específico. No deberemos usar EGP dentro de nuestra propia red, sino sólo para comunicarnos con la red nacional. Si tenemos varias clases de gateways, necesitaremos usar un protocolo entendible por todos ellos. En muchas ocasiones este protocolo será RIP (Routing Information Protocol). A veces podremos usar protocolos más complejos entre los gateways que los soporten, y usar RIP sólo cuando nos comuniquemos con gateways que no entiendan estos protocolos.

Si ya hemos elegido un protocolo de enrutamiento y lo hemos puesto en marcha, todavía nos quedan por tomar algunas decisiones. Una de las tareas mas básicas de configuración que tenemos que completar es suministrar la información de la métrica. Los protocolos más simples, como RIP, normalmente usan "cuenta de saltos", de manera que una ruta que pasa a través de dos gateways es mejor que una que pasa por tres. Por supuesto, si la última ruta usa líneas de 1'5 Mbps y las primeras líneas de 9.600 bps, sería una mala elección. La mayoría de los protocolos de enrutamiento tienen medios para tomar esto en cuenta. Con RIP, podríamos tratar las líneas de 9.600 bps como si fueran "saltos" adicionales, de

manera que la mejor línea (la más rápida) tenga una métrica menor. Otros protocolos más sofisticados tendrán en cuenta la velocidad de la línea de forma automática. Generalmente, estos parámetros deberán asociarse a una interfase en particular. Por ejemplo, con RIP deberemos establecer explícitamente el valor de la métrica, si se está conectado con una línea de 9.600 bps. Con aquellos protocolos que tienen en cuenta la velocidad de las líneas, deberemos de especificar la velocidad de las líneas (si el gateway no los puede configurar automáticamente).

La mayor parte de los protocolos de enrutamiento han sido diseñados para que cada gateway se aprenda la tipología de toda la red, y elegir la mejor ruta posible para cada datagrama. En algunos casos no estaremos interesados en la mejor ruta; por ejemplo, puede que estemos interesados en que el datagrama se desplace por una parte de la red por razones de seguridad o económicas. Una manera de tener este control es especificando opciones de enrutamiento.

Dichas opciones varían mucho de un gateway a otro, pero la estrategia básica es que si el resto de la red no conoce dicha ruta, no será utilizada. Estos controles limitan la forma en la que se van a usar las rutas.

Hay métodos para que el usuario ignore las decisiones de enrutamiento hechas por los gateways. Si realmente necesitamos controlar el acceso a ciertas redes, podemos hacer dos cosas:

- Los controles de enrutamiento nos aseguran que los gateways usan sólo las rutas que queremos;
- Usar listas de control de acceso en los gateways adyacentes a las redes controladas.

Estas dos opciones trabajan a distinto nivel. Los controles de enrutamiento afectan a lo que ocurre a la mayoría de los datagramas: aquéllos en los que el usuario no

ha especificado manualmente una ruta. Nuestro mecanismo de enrutamiento ha de ser capaz de elegir una ruta aceptable para ellos. Una lista de control de acceso añade una limitación adicional, preservándonos de usuarios que incluyesen su propio enrutamiento y pasasen nuestros controles.

Por razones de fiabilidad y seguridad, puede que también haya controles con listas de gateways de las que podemos aceptar información. También es posible hacer una clasificación de prioridad. Por ejemplo, podemos decidir hacer antes los enrutamientos de nuestra propia organización antes que los de otras organizaciones, u otras partes de la organización. Esto tendrá el efecto de dar preferencia al tráfico interno frente al externo, incluso si el externo parece ser mejor.

Si usamos varios protocolos distintos de enrutamiento, probablemente tendremos que afrontar algunas decisiones respecto a la información que se pasan entre ellos. Puesto que el uso de varios protocolos de enrutamiento está frecuentemente asociado a la existencia de varias organizaciones, deberemos de tomar la precaución de hacer estas decisiones consultando con los administradores de las redes de dichas organizaciones. Este tipo de decisiones puede tener consecuencias en las otras redes bastante difíciles de detectar. Podríamos pensar que la mejor forma de configurar un gateway es que fuese capaz de entender todos los protocolos, pero hay algunas razones por las que esto no es recomendable:

* Las métricas usadas por los distintos protocolos no son compatibles en muchas ocasiones. Si estamos conectados a dos redes externas distintas, podemos especificar que una siempre debe usarse preferentemente a la otra, o que la más cercana es la que debe usarse, en lugar de comparar la métrica recibida de las dos redes para ver cuál tiene la mejor ruta.
* EGP es especialmente delicado, ya que no admite bucles. Por esto hay unas reglas estrictas para regular la información que hay que intercambiar para

comunicarse con una red principal usando EGP. En aquellos casos en que se use EGP, el administrador de la red principal debería ayudarnos a configurar el enrutamiento.

* Si tenemos líneas lentas en nuestra red (9.600 bps o menos), puede que no queramos enviar la tabla de enrutamiento completa a través de la red. Si nos conectamos a una red exterior, tenemos la posibilidad de tratarla como una ruta por defecto, en lugar de introducir toda su información en nuestro protocolo de enrutamiento.

BLUETOOTH

Es una tecnología desarrollada por Ericsson en 1994, que hace factible la conectividad inalámbrica entre dispositivos a corta distancia, éstos pueden llegar a formar redes con diversos equipos de comunicación: computadoras móviles, radiolocalizadores, teléfonos celulares, PDAs, e, inclusive, electrodomésticos.

El estándar Bluetooth se compone de dos capítulos, uno de ellos describe las especificaciones técnicas principales, mientras que el otro define perfiles específicos para aplicaciones, estos últimos aseguran la interoperabilidad de dispositivos Bluetooth entre fabricantes. Algunos de estos perfiles son el de acceso genérico, identificación de servicio, puerto serial, acceso a LAN sincronización y el de dispositivo de información móvil (MIDP).

La IEEE ha desarrollado un protocolo equivalente denominado Wireless Personal Area Network (WPAN), 802.15, con el objetivo de lograr la interoperabilidad con otros dispositivos inalámbricos.

Características

* Tecnología inalámbrica. Reemplaza la conexión alámbrica en distancias que no exceden los 10 metros, alcanzando velocidades del rango de 1Mbps.

- Comunicación automática. La estructura de los protocolos que lo forman favorece la comunicación automática sin necesidad de que el usuario la inicie.
- Bajo consumo de potencia. Lo pequeño de los dispositivos y su portabilidad requieren de un uso adecuado de la energía, el cual provee esta tecnología.
- Bajo costo. Los dispositivos de comunicación que soporta pueden experimentar un incremento en su costo no mayor a 20 dólares con tendencia a bajar. Asimismo, su operación se efectúa bajo una banda de frecuencias no licenciada (2.4GHZ), lo que ayuda a su bajo costo.
- Integración de servicios. Puede soportar transmisiones de voz y datos de manera simultánea.
- Transmisión omnidireccional. Debido a que basa su comunicación en radiofrecuencia, no requiere línea de vista y permite configuraciones Punto multipunto.
- Seguridad.Utiliza Spread Spectrum Frequency Hopping como técnica de multiplexaje, lo que disminuye el riesgo de que las comunicaciones sean interceptadas o presenten interferencia con otras aplicaciones. Provee también especificaciones para autenticar dispositivos que intenten conectarse a la red Bluetooth, así como cifrado en el manejo de llaves para proteger la información.
- Establecimiento de redes. Tiene la característica de formar redes en una topología donde un dispositivo hace las veces de maestro y hasta siete más operando como esclavos. Esta configuración se conoce como piconet. Un grupo de piconets, no más de diez, es referido como Scatternet.

BIBLIOGRAFÍA

Burguess, Mark. Principles of Network and System Administration. John Wiley & Sons. USA. 2002.

Bob Toxen. Real World Linux Security. Prentice Hall, second edition, 2003.

C. F. Hemphill and J. Hemphill. Security Procedures for Computer Systems. Dow Jones- Irvin, 1973.

Donald E. Knuth. The Art of Computer Programming. Seminumerical Algorithms., volume
2. Addison-Wesley, 2nd edition, 1981.

Eckstein, Robert. Using Samba. Adison Wesley. USA. 1999.

FRISCh, Aeling. Essential System Administration. O'Reilly & Associates. USA. 1995.

Gómez, Pedro. Introducción a Windows 2000 Server. Ed. Eidos. España

Hilley, Valda. Los secretos de Windows NT 4.0 Ed. Anaya Multimedia. España

Hidalgo, José Félix. Introducción a las redes locales Ed. Anaya Multimedia. España.

Hung, Craig. TCP/IP Network Administration. O'Reilly & Associates. USA. 2002.

I. Vinogradov. Fundamentos de la Teoria de los Numeros. MIR, 1977.

J. van Leeuwen, editor. Handbook of Theoretical Computer Science. Elsevier Science Publishers, 1990.

Ken Thompson. Reflections on trusting trust. Communications of the ACM, 27(8):761-763, 1984. Malhotra, Ravi. IP Routing. O'Reilly & Associates. USA. 2002.

Michael R. Garey and David S. Johnson. Computers and Intractability. A guide to The Theory of NP-Completeness. W.H.Freeman and Company, 1979.

Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. Concrete Mathematics.

Addison-Wesley, second edition, 1994.

Ralph P. Grimaldi. Discrete and Combinatorial Mathematics. Addison-Wesley, 4 edition, 1999.

Sloan, Josepd D. Networks Troubleshooting Tools. O'Reilly & Associates. USA. 2001.

Stevens, W. Richard. The Protocols. (TCP/IP Ilustaded). Addison Wesley. USA.1993.

Tanembaun, A. S. Computer Networks. Prentice may. USA. 1996.

W.Richard Stevens. UNIX Networking Programing. Prentice Hall, 1990.

W.Richard Stevens. TCP/IP Illustrated. The Protocols, volume 1 of Professional Computing Series. Addison-Wesley, 1994.

W.Richard Stevens. TCP/IP Illustrated. The Implementation, volume 2 of Professional Computing Series. Addison-Wesley, 1995.

W.Richard Stevens. TCP/IP Illustrated. TCP for Transactions, HTTP, NNTP, and the UNIX Domain Protocols, volume 3 of Professional Computing Series. Addison- Wesley, 1996.

W. Richard Stevens. UNIX Network Programming, volume 1. Prentice Hall, second edition, 1998.

Lourdes Velázquez Pastrana.

http://www.enterate.unam.mx/Articulos/2004/octubre/bluetooth.htm
http://www.suse.org
http://www.linux.org
http://www.apache.org
http://www.microsoft.com