UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN

INFORME CON FINES DIDÁCTICOS DEL DESARROLLO EN IMPLEMENTACIÓN EN SISTEMAS DE SOFTWARE LIBRE EN LINUX Y PROGRAMACIÓN EN VISUAL BASIC .NET

TRABAJO ESCRITO EN LA MODALIDAD DE SEMINARIOS Y CURSOS DE ACTUALIZACIÓN Y CAPACITACIÓN PROFESIONAL
QUE PARA OBTENER EL TÍITULO DE
INGENIERO EN COMPUTACIÓN
P R E S E N T A:

JAVIER VARGAS MARTÍNEZ

MÉXICO **2006**

AGRADECIMIENTOS

A MI FAMILIA:

Papá (Pedro): Que me enseñaste.
Mamá (Guadalupe): Simplemente estoy aquí por ti.
Concepción: Mi modelo a seguir en aprendizaje y dedicación.
Sandra: Tu presencia es señal de que vale la pena vivir.

A MIS AMIGOS:

Saben que ustedes son pilar importante en mi formación profesional y personal.

A MIS PROFESORES:

Enseñándome el significado de ser universitario. En especial al Mtro. Luis Ramírez Flores, por que me enseñó a comprender las cosas de una manera sencilla.

A LAURA OCAMPO GARCÍA:

Demostrando que se puede hacer todo lo que uno quiera de manera pragmática y ordenada. Sobre todo por aceptarme como soy y por amarme. Estando presente en esta etapa de mi vida. Te quiero.

ÍNDICE

CAPITULO III

ADMINISTRACIÓN DE SERVIDORES WWW CON LINUX 40

CAPITULO IV

INTRODUCCIÓN A LA SEGURIDAD EN CÓMPUTO 65

INTRODUCCIÓN

La Redes de Información se ha convertido en parte de toda empresa sea cual sea el ámbito en el que se desarrolla, en número de tareas que se ejecutan son cada vez más grandes y a medida que pasa el tiempo se hacen cada vez más críticas, al mismo tiempo estas tareas hacen que la empresa se expanda y con ello toma mayor relevancia el manejo de la información (a nivel computacional), por consiguiente se hace necesaria la implementación de Sistemas de Alta Disponibilidad en los sistemas de cómputo.

Es necesario contar con el conocimiento para realizar un trabajo de manera óptima, y dada la gran importancia del uso de los sistemas, mejorar la utilización de recursos y lograr con mayor eficiencia los objetivos de cualquier tipo de organización.

Dar a conocer los principios, técnicas y fundamentos básicos para la administración y creación, así como la familiarización con las herramientas administrativas y la utilización de las mismas con el fin de que se tome conciencia de la gran necesidad que los ingenieros cuenten con dichos conocimientos.

Este trabajo se desarrolla bajo el enfoque Didáctico, ya que hoy en día se requiere que las nuevas generaciones tengan presente que ya existe material necesario con el cual disponer, para que aprendan de una manera rápida y enfocada a la práctica, el uso de diferentes sistemas y software que existe en el mercado laboral, con el cual se pueden desarrollar y de este modo poder tener, al menos, las herramientas básicas para que puedan empezar a desenvolverse como ingenieros en computación. Así, con el presente trabajo se busca dar el conocimiento elemental e intuitivo para que el usuario o la persona que se interese por dicho material,

Aquí, se busca integrar el uso de sistemas y software para elaborar un sistema que permita automatizar las tareas y trabajo de manera que se acreciente la utilización y administración de medios, y reducción de tiempo. Con esto, por ejemplo, se ayudaría en gran medida que el registro de inventario de cada equipo ya no se haga manualmente, sino de forma semiautomática y, por consiguiente, almacenarlo en una base de datos; se puede lograr con la utilización del código de barras, ya que al capturar el código de barras, con algún dispositivo especial para ello, se puede interactuar directamente a la base de datos.

Hoy en día el uso de los sistemas de software libre se ha desarrollado, y es una muy buena opción a la hora de hablar de Implementación de Sistemas. Una de las principales ventajas es que se trata de software que es de fácil adquisición y además de forma gratuita; reduciendo de ésta manera costos. Además este tipo de tecnologías está en constante desarrollo y son relativamente fáciles de implementar.

En el primer capítulo se trata de uso del ***Sistema Operativo Linux***, ya que es necesario conocerlo y utilizarlo (al menos básicamente) antes de empezar a instalarlo. Es necesario comprender la forma en la que este Sistema Operativo trabaja, la manera en la que está estructurado: el ***Hardware***, el ***Kernel*** y el ***Shell***. También hay que saber que existen, al igual que Windows, diferentes versiones de Linux, como Slackware (que es la versión con la cual se trabaja en este trabajo), Mandrake, Debian, Suse, Fedora, Red Hat, y un largo etcétera.

Cuando se empieza a trabajar con un sistema es necesario comprender como es que se empieza a interactuar con él, y esto se hace por medio de ***Comandos***, y ***Archivos***. Hay que tener en cuenta que tanto los comandos como los archivos tienen muchos atributos, permitiendo al usuario obtener mayor eficiencia al momento de ejecutar alguna acción dentro del sistema.

En el segundo capítulo se habla de cómo ***Instalar y Administrar el Sistema Operativo Linux***, las Actividades y Tareas que debe desempeñar un Administrador. En necesario tener en

cuenta antes de la instalación qué paquetes van a ser necesarios para el Sistema. Cabe mencionar que existen al menos 3 tipos diferentes para la instalación de un Sistema. Para usuario de Oficina o Casa, como Estación de Trabajo (o *Workstation*) y como Servidor, siendo éste el sistema en el cual se instalan casi todos los paquetes, en especial aquellos referentes a la administración de recursos en red, gestión de usuarios, etc. Se ilustra con algunas imágenes el proceso de instalación tanto de paquetes como de recursos y servicios que tendrá instalado el Sistema.

El tercer capítulo trata de la ***Administración y Aplicación de Servidores Web***, en este libro se trabaja con el Servidor Apache. Se describen las diferentes aplicaciones que tienen los servidores Web. Las Directivas, que son las Opciones de Configuración de alguna Funcionalidad que ofrece algún Servicio dado, y con estás se brindan y/o restringen servicios, recursos y archivos a diferentes usuarios.

En el cuarto capítulo trata de la ***Introducción a la Seguridad en Cómputo***, la clasificación y diferentes tipos de ataques de información. Diferentes herramientas utilizadas en la administración y manejo de información a través de una red de computadoras. Encriptación y Cifrado de información. Mecanismos de Seguridad, Sistemas de Detección de Intrusos (Intrusion Detection System, IDS).

En el quinto capítulo se habla acerca de la ***Creación de Páginas Web***, por medio del leguaje HTML, de los diferentes tipos de Etiquetas y elementos que pueden estar dentro de las páginas Web. Los diferentes Editores Web que existen, desde un simple editor de Textos (***vi*** o ***pico***) hasta algunos más sofisticados como ***Quanta Plus*** que se utiliza a nivel más gráfico. La comprensión de URL y los elementos que la constituyen. Diferentes formas de dar formatos a textos, tablas y formularios. Como agrupar diferentes páginas Web en una sola.

En el sexto capítulo se trabaja con la ***Programación con PHP***, que es una alternativa viable y muy robusta para interactuar con información en la Web. La sintaxis con la que cuenta este lenguaje de programación es muy fácil de entender, al igual que los diversos tipos de datos y operadores que maneja. También se describe el manejo de los diferentes bloques y sentencias, Funciones y Arreglos. Así como el manejo de cadenas (que no son más que información de carácter no numérico -textos, por ejemplo-) para poder interactuar con archivos. Manejo de sesiones y cookies. Además de las funciones de envío y manejo de correo.

El séptimo capítulo trabaja con la ***Interacción de WWW con Bases de Datos***, utilizando para ello el manejo de MySQL, que es un gestor de Bases de Datos muy utilizado para administrar información, utilizado en plataformas Linux aunque puede usarse en otras plataformas. Con MySQL se pueden crear desde Bases de Datos, Tablas; Insertar, Cambiar, Actualizar y Borrar Información entre otras cosas. También se pueden vincular estas actividades con su uso en ambiente web, con lo cual se incremente de una manera significativa la administración de datos de manera remota y más confiable.

El octavo capítulo trata acerca de la ***Programación con Visual Basic .Net***, de cómo con este programa se pueden hacer aplicaciones utilizando herramientas con interfase gráfica y por medio de programación, haciendo que dé como resultado aplicaciones robustas, seguras, extensibles y personalizadas. En la programación se manejan elementos básicos, el uso de arreglos y correcciones de errores.

CAPITULO I

SISTEMA OPERATIVO LINUX

La historia de Linux se remonta con Unix. El Sistema Operativo UNIX fue desarrollado como un Sistema Multitarea con tiempo compartido para minicomputadoras y mainframes a mediados de los años 70 del siglo pasado, y desde entonces se ha convertido en uno de los sistemas más utilizados a pesar de su, ocasionalmente, confusa interfaz con el usuario y el problema de su estandarización.

Linux es una versión de UNIX de libre distribución, inicialmente desarrollada por Linus Torvalds en la Universidad de Helsinki, en Finlandia. Fue desarrollado con la ayuda de muchos programadores y expertos de Unix, Parte del software para Linux se desarrolla bajo las reglas del proyecto de GNU de la *Free Software Foundation*, en Cambridge, Massachusetts. El 5 de Octubre de 1991, Linus anunció la primera versión "oficial" de Linux, la 0.02. Ya podía ejecutar ***bash*** (el shell de GNU) y ***gcc*** (el compilador de C de GNU), pero no hacía mucho más. No había nada sobre soporte a usuarios, distribuciones, documentación ni nada parecido. Hoy, la comunidad de Linux aún trata estos asuntos de forma secundaria. Lo primero sigue siendo el desarrollo del kernel.

En su Diseño Linux cuenta con determinadas características, es un sistema operativo completo con multitarea y multiusuario (como cualquier otra versión de UNIX). Esto significa que pueden trabajar varios usuarios simultáneamente en él, y que cada uno de ellos puede tener varios programas en ejecución. Fue desarrollado buscando la portabilidad de las fuentes: se encontrará que casi todo el software gratuito desarrollado para UNIX se compila en Linux sin problemas. Y todo lo que se hace para Linux (código del núcleo, drivers, librerías y programas de usuario) es de libre distribución. En Linux también se implementa el control de trabajos POSIX (que se usa en los shells csh y bash), las pseudoterminales (dispositivos pty), y teclados nacionales mediante manejadores de teclado cargables dinámicamente. Además, soporta consolas virtuales, lo que permite tener más de una sesión abierta en la consola de texto y conmutar entre ellas fácilmente.

1.1 FILOSOFÍA

La Filosofía de Linux, al igual que su diseño, está basada en el Sistema UNIX, se dice que un sistema potente y complejo debe de ser **simple, general y extensible.** El principio fundamental de esta filosofía es *diseñar comandos que hagan solamente una cosa pero que la hagan de forma simple y correcta.* El sistema contempla los archivos de manera extremadamente simple y general dentro de un modelo único. Ve de la misma forma los directorios, archivos ordinarios, los dispositivos, tales como impresoras, discos, teclados y terminales de pantalla.

1.2 ESTRUCTURA

La Estructura de Linux, se considera por niveles, al igual que UNIX. En el fondo se encuentra el **hardware**, que consiste de discos, terminales, CPU, memoria y demás dispositivos. Corriendo sobre el hardware está el **sistema operativo**; su función es controlar el hardware y proporcionar una interfaz de llamada al sistema a todos los programas. Estas llamadas al sistema permiten a los programas del usuario crear y manejar procesos, archivos y recursos. Para entender un poco mejor esta estructura, es necesario describir los componentes de la misma, y mostrado en la figura 1.1:

El **Hardware**: Es la parte tangible y visible que compone toda la computadora y sus partes, como son el microprocesador, tarjeta madre, de red, modem, disco flexible, CD-ROM, teclado, mouse, bocinas, micrófono, impresora, etc. En su conjunto sirven para ingresar y obtener información a través del Kernel y el Shell.

El **Kernel**: Es el corazón (o núcleo) del sistema operativo y es iniciado cada vez que el sistema es arrancado. Maneja todos los recursos del sistema y los presenta al usuario de una

manera coherente. Controla el acceso a la computadora y sus archivos, asigna recursos a las distintas actividades que se llevan a cabo en la computadora, mantiene el sistema de archivos y administra la memoria de la computadora. Aunque los usuarios comunes rara vez tienen una interacción explícita con el Kernel, éste es un aspecto medular para comprender Linux. La versión con la que se trabajó Linux es 2.4.26, y la versión que está disponible, al momento de hacer este reporte es la 2.6.11.

El **Shell**: Cuando se ingresa a Linux, la interacción con el sistema es controlada por un programa llamado ***Shell*** o intérprete de Comandos, y por medio de éste es como el usuario se entiende con el sistema. Un shell es un programa que interpreta y ejecuta los comandos conforme se proporcionan de la terminal. No se requiere ningún privilegio especial para ejecutar un shell; para el kernel es un programa más.

Entre las características más comunes de un shell están la interpretación de comandos para formar interconexiones, la recuperación de comandos previos, etc. Un guión de shell es un archivo que contiene secuencias de comandos previos de shell, igual que se hubieran tecleado, los guiones de shell permiten adecuar el entorno de trabajo sin tener que realizar programación "real".

El shell se utiliza para tres propósitos principales:

1. Como intérprete de comandos: Cuando el shell se utiliza como intérprete de comandos, el sistema espera recibir comandos en el prompt.

2. Para personalizar el ambiente de trabajo dentro de una sesión: Un shell de Linux define las variables para controlar el comportamiento del sistema cuando se ingresa a él. El sistema define estas variables por sí mismo; el usuario puede definir otras variables en los archivos de inicialización que son leídos y ejecutados cada vez que se acceda al sistema.

3. Como lenguaje de programación: Todos los Shell de Linux contienen un conjunto especial de comandos que pueden usarse para crear programas de shell. De hecho, muchos de los comandos de shell pueden usarse como si fueran comandos de Linux, y viceversa. Los programas de shell son útiles para ejecutar comandos repetidamente o en forma condicional, como en un lenguaje de alto nivel; también pueden ejecutarse archivos de forma secuencial mediante archivos ejecutables.

Entre los shell más utilizados se tienen:

- ✓ El *Shell Bourne*, **sh** → Escrito por Steve Bourne en 1979, es el más viejo y es parte de la séptima edición de UNIX y el primero de los shell principales que utiliza Linux. Muchos de los usuarios lo prefieren para uso interactivo. Casi todos los guiones de shell siguen los convencionalismos del shell Bourne.

- ✓ El *Shell C,* **csh** → Disponible a través del comando csh, se desarrolló como parte de BSD UNIX; su nombre no lo hace más parecido al lenguaje C que el shell Bourne. Incorpora alias e historia de comandos.

- ✓ El *Shell Korn*, **ksh** → Ofrece una síntesis de las características de los shell Bourne y C además de las propias. Creado por David Korn de AT&T Bell Laboratorios en 1982, presentando versiones mejoradas en 1986 y 1988. Se incluye como características estándar de la versión 4 de System V y otros sistemas; sigue los convencionalismos del shell Bourne. Incluye operaciones aritméticas y manipulación de cadenas.

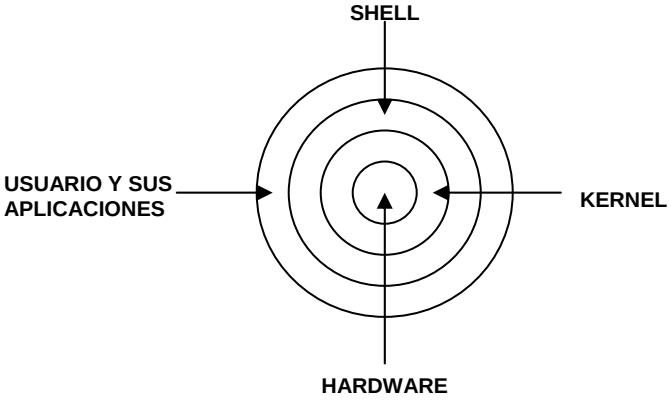


Figura 1.1.- Diagrama de la estructura del S.O. Llinux

1.3 PLATAFORMAS MÁS COMUNES

La desconcertante elección entre un número siempre creciente de distribuciones de Linux puede crear confusión entre aquellos nuevos en Linux. Se listan algunas distribuciones que generalmente se consideran las más extendidas entre los usuarios de Linux de todo el mundo. Hay otras muchas que probar.

Lycoris, Xandros y Lindows son consideradas las mejores para aquellos usuarios nuevos en Linux que quieren empezar a ser productivos con Linux lo antes posible sin tener que aprender todas sus complejidades. En el lado opuesto tenemos a Gentoo, Debian y Slackware que son distribuciones más avanzadas que requieren un completo aprendizaje antes de poder ser usadas eficientemente. Mandrake, Red Hat y SuSE se encuentran a medio camino entre ambas. Knoppix es un caso aparte, es genial para probar Linux sin tener que hacer nada, ya que funciona directamente del CD, si ninguna instalación.

Como Linux es un sistema de código abierto, muchos usuarios de todo el mundo han ido desarrollando diferentes versiones, como se muestra en la Tabla 1.1.

- Debian GNU/Linux
- LindowsOS
- Knoppix
- Red Hat Linux
- SuSE Linux
- Gentoo Linux
- Lycoris Dekstop/LX
- Mandrake Linux
- Slackware Linux
- Xandros OS

Tabla 1.1.- Versiones de Linux

En este curso se utiliza la versión de Slackware. Dado que esta version es extremadamente estable y segura, muy recomendada para servidores. Los administradores con experiencia en Linux encuentran que es una distribución con pocos fallos, ya que usa la mayoría de paquetes en su forma original, sin demasiadas modificaciones propias de la distribución, que son un riesgo potencial de añadir nuevos fallos. Slackware es una buena distribución para aquellos interesados en profundizar en el conocimiento de las entrañas de Linux. Posiblemente, la mejor característica de esta distribución es: que si se necesita ayuda con el sistema linux, encuentra un usuario de Slackware. Es más probable que resuelva el problema que otro usuario familiarizado con cualquier otra distribución.

Pros: Alta estabilidad y ausencia de fallos, sigue fielmente los principios de UNIX.
Contras: Toda la configuración se realiza mediante la edición de ficheros de texto, autodetección de hardware limitada.
Sistema de paquetes: TGZ
Descarga gratuita: Si
La página de Slackware Linux: **http://www.slackware.org**

1.4 COMANDOS Y UTILERIAS

Conceptos Básicos

Para comenzar a trabajar bajo el ambiente linux, es necesario que el usuario tenga una cuenta para accesar a la PC o se conecte de manera remota a un servidor Linux. Esta cuenta consta de un ***nombre de usuario*** o ***'login'*** que es el nombre que tendrá el usuario y, una ***contraseña*** o ***'password'***, que es una clave que sólo tendrá el usuario asignado a esa cuenta; esta clave podrá contener caracteres numéricos y alfanuméricos. Hecho lo anterior, el usuario ya podrá empezar a tabajar en su sesión, ya sea en un servidor, en la PC o terminal remota. Ya que se ha terminado de usar la sesión el usuario puede salir de la sesión con el comando ***logout, logoff*** o ***exit***. Éstas instrucciones sólo sirven para salir de una sesión, no para apagar la computadora, o cerrar la aplicación de la conexión remota (en éste caso algunos programas, si permiten cerrar su aplicación al teclear alguna de las instrucciones mencionadas anteriormente).

En Linux, se tiene opción de utilizar múltiples consolas o terminales, denominadas ***Consolas Virtuales***[1]. Si el usuario decide cambiar la contraseña que se le ha asignado, que por seguridad se pide que cada mes se renueve, se puede hace con el comando *passwd.* Esta contraseña se guarda en el archivo */etc/shadow* de manera encriptada para seguridad del usuario.

Como se ha dicho con anterioridad, Linux, heredó la estructura de datos de UNIX. De modo tal que el sistema se estructura de modo jerárquico en el cual no existen unidades de disco, en su lugar cada unidad de almacenamiento así como cada dispositivo de hardware son reconocidos como un archivo o directorio dentro del sistema. Existe sólo una raíz en el sistema la cual representa la parte más alta de la estructura de directorios y es conocida como root o raíz, es representada por el símbolo de / , a partir de este punto se desprenden diferentes ramas de directorios como las siguientes:

`/`	Raíz del sistema
`/ var`	Variables del sistema
`/var/spool/mail`	Archivos de correo electrónico
`/var/log`	Bitácoras del sistema
`/dev`	Dispositivos de hardware
`/etc`	Archivos de configuración del sistema

Estructura básica de los comandos.

Sintaxis:

comando -opciones argumento1 argumento2 ...

[1] Se puede accesar a éstas con el juego de teclas ***Ctrl + Alt + F1 … F6*** (donde F1 … F6 son el número de cada una de las consolas) y un juego de teclas (***Ctrl + Alt +F7***) para accesar a modo gráfico, siempre y cuando este modo esté habilitado.

Siempre, al inicio se teclea el comando que se quiere ejecutar, a continuación las opciones si se necesita modificar la salida de la ejecución del comando, y al final los argumentos que indican la parte donde se deposita la salida del comando. Las opciones no son necesarias para la ejecución del comando, siempre son antecedidas de un guión y se puede utilizar más de una opción[2].

```
ls –la
less  /etc/passwd
ps –fea  | grep  httpd
```

Para ubicar el directorio de trabajo en el cual está trabajando el usuario, se teclea el comando **pwd** (Print Working Directory), este comando imprime en pantalla el directorio actual de trabajo, es decir, donde se encuentra posicionado dentro del sistema de directorios indicándonos la ruta absoluta.

Después, se empieza a hacer una exploración del sistema en general; para cambiarse de directorios, se emplea el comando **cd** (acronimo de *change directory*) que permite al usuario ir de un lugar a otro dentro del sistema, tecleando la ruta a la que se desea ir.

Para poder entender como cambiarse de directorio es necesario conocer, que hay ***Rutas Relativas*** y ***Rutas Absolutas***. La Ruta Absoluta comienza desde la raíz hasta el directorio o subdirectorio que se quiera accesar, por ejemplo:

```
root@inventario:~$  cd  /home/usuario
root@inventario:/home/usuario$
```

Se le indica al sistema que se quiere ir al directorio *usuario* ubicado dentro del directorio *home* del sistema raíz (/).

Ahora, cuando se habla de una ruta Relativa, ésta comienza desde la posición actual en la que está trabajando el usuario. Por ejemplo, si el usuario esta en */home/usuario*, para cambiar a */etc/rc.d* se tiene que subir hasta el nivel que sea común entre ambas rutas, la del usuario y la de destino, en este caso el directorio común es / (raíz). Entonces se describe, que para ir a un directorio de nivel superior o directorio padre se tienen que poner **. .** (punto punto) y para trabajar al mismo nivel del directorio es con **.** (punto). Así se tiene que, para hacer el cambio de directorio, sería de la siguiente manera:

```
cd ../../etc/rc.d
```

Para poder ver la información que contiene un directorio se tiene el comando **ls,** que con algunos atributos muestra la información de diferentes modos. Con **ls** muestra la información de ésta manera

```
javier@inventario:~$  ls
TA.txt    archivo1   mail/   passwd     public_html/
```

Mostrando sólo los directorios y archivos contenidos dentro del directorio. Si se quiere ver o listar la información adicional respecto a dichos archivo se puede hacer de la siguiente manera:

[2] Los argumentos son requisito para la ejecución del comando, si no se da el argumento apropiado para la ejecución del comando, éste no se ejecutará y mandará un mensaje de error el sistema.

ls –la

Donde:

-l → Muestra el contenido del directorio en forma detallada.
-a → Incluye los archivos y/o que están ocultos en el directorio.

Comodines *, ?

Existen dos caracteres llamados comodines los cuales pueden sustituir un caracter o conjunto de caracteres durante la ejecución de comandos.

1.5 ATRIBUTOS DE LOS ARCHIVOS

Los atributos sirven para entender mejor los datos que se presentan en pantalla, van a indicar el tipo de archivo, los permisos con los que cuenta, el usuario y grupo de trabajo al que pertenece, tamaño en bytes que tiene, fecha y hora de creación ó última modificación. A continuación se decriben mediante un ejemplo:

```
d rwx r-x r-x 6 javier users 4096 2005-04-21 12:56 public_html/
```

Donde:

d → Es el Tipo de Archivo, que puede ser:

d → Directorio
b → Bloque
o → Ordinario
l → Liga
- → Normal

rwx → Este primer bloque, corresponde a los permisos que tiene el ***usuario propietario (u)*** del archivo o directorio. En este bloque se tienen habilitados todos los permisos:

r (read) de lectura: Permite que el archivo pueda ser leído o copiado.

w (write) de escritura: Permite escribir en el archivo, hacer modificaciones o borrarlo si así lo desea.

x (execute) de ejecución: Los archivos ejecutables son aquellos que pueden realizar un proceso.

Los siguientes dos bloques se muestran con los permisos **r-x**. El primer bloque que aparece corresponde a los permisos del ***grupo de usuarios (g)*** que puede trabajar con el archivo o directorio y, el siguiente bloque pertenece a ***otros usuarios (o)***, que son usuarios ajenos al grupo de usuarios. Estos elementos se describirán más a detalle, en el siguiente capítulo. Para ambos bloques de permisos, se tienen solamente habilitados los permisos de **lectura** y de **ejecución**, de manera que sólo pueden ver los archivos y ejecutarlos si fuere ese el caso, pero no pueden modificar nada de éstos archivos y, a continuación, en informática, especialmente en sistemas operativos de la familia Unix (Solaris,HP-UX,AIX,GNU/LINUX, etc.).

A todos los archivos y directorios del sistema operativo se accede internamente mediante inodos[3], en sistemas operativos Unix antiguos, el número de inodos estaba limitado a 65535

[3] Un inodo o Index Node es un apuntador a sectores específicos de disco duro en los cuales se encuentra la información del archivo, un inodo también contiene información de permisos, propietarios y grupos a los cuales pertenece el archivo.

(inodos de 32 bits), pero en la actualidad, la mayoría de los sistemas de archivos modernos utilizan inodos de 64 bits.

javier → Es el nombre del usuario propietario al que pertenece el archivo.
users → Es el grupo de usuarios al que pertenece el archivo. Por predeterminación de Linux, se crea el grupo de trabajo ***users***.
4096 → Es el tamaño que tiene el archivo en Bytes.
2005-04-21 12:56 → Es la fecha y hora en que se creó o se modificó el archivo o directorio.
public_html/ → Es el nombre que tiene el Archivo o el Directorio. Se diferencia un archivo de un directorio, por que el directorio tiene una diagonal (***/***) al final del nombre, en cambio el archivo no lo tiene.

Para tener una mayor información, se puede consultar el manual de comandos, simplemente ejecutando el comando **man** seguido del comando que se quiere ver, por ejemplo se quiere ver para que sirve el comando **ls** junto con información adicional, hay que ejecutar el comando de la siguiente manera:

```
man ls
```

Lo que desplegará una pantalla como sigue:

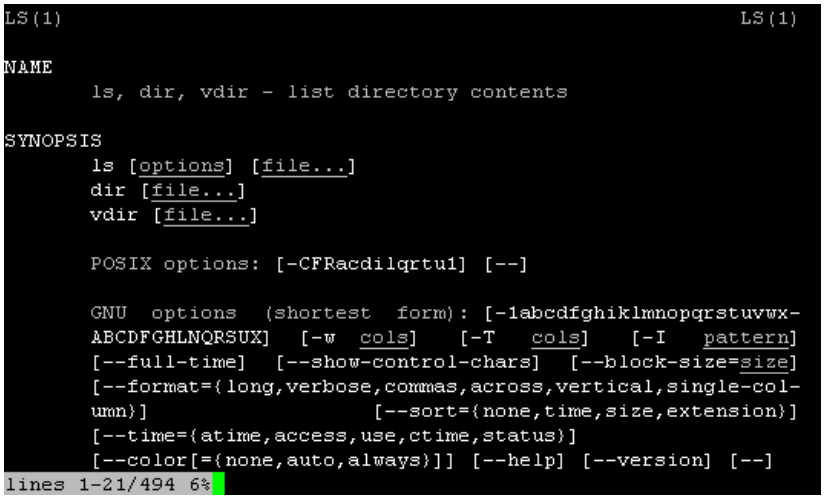


Fig. 1.2.- Fragmento de contenido del manual de referencia del comando **ls**

La ayuda se compone de 8 secciones, las cuales se describen a continuación:

1 Herramientas de usuario
2 Llamadas al sistema
3 Llamadas a bibliotecas C
4 Información de controladores de dispositivos
5 Archivos de configuración
6 Juegos
7 Paquetes
8 Herramientas de sistema

Para consultar directamente una sección del manual se solicita el manual seguido de la sección deseada y al final el comando que se desea consultar.

Con los comandos vistos anteriormente, se pueden listar los elementos contenidos en un Directorio. Ahora se verán otros comandos para trabajar con Directorios y Archivos.

Para ***crear*** un Directorio, se teclea el comando ***mkdir***, pero hay que tomar en cuenta que el sistema Linux es un sistema de tipo ***case sensitive***, lo que significa que distingue entre mayúsculas y minúsculas. De tal forma que:

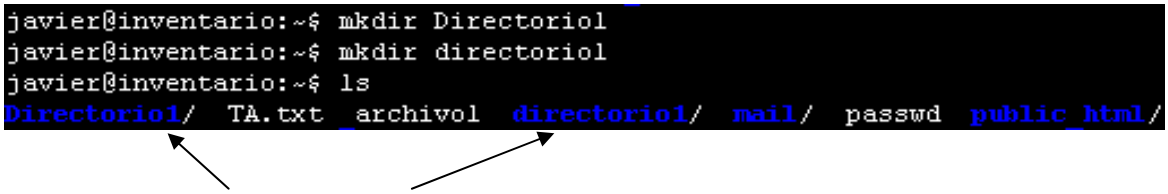


Fig. 1.3.- Creación de Directorios (directorio1 y Directorio1).

Para que se pueda ejecutar **mkdir** es necesario que se tengan los permisos de escritura en el directorio en el cual se quiera generar el otro directorio.

Para ***Borrar*** un Archivo o un Directorio vacío se utiliza el comando **rm**, que es abreviación de *remove*. Así que se tiene la siguiente sintáxis:

```
javier@inventario:~$ rm  archivo.cualquiera
```

Aquí se está indicando que se va a borrar o remover el **archivo.cualquiera,** así que será eliminado.

Ahora, para eliminar un directorio se hace de manera similar, como se muestra a continuación:

```
javier@inventario:~$ rm Directorio1/
javier@inventario:~$ rmdir Directorio1/
```

Así el **Directorio1/** se borrará o removerá. Hay que tomar en cuenta que para borrar un directorio de esta manera debe estar vacío. Por el contrario si el Directorio en cuestión contiene archivos o subdirectorios, hay que agregarle el atributo **–r** o **–R ,** que borra o remueve ***recursivamente*** el Directorio junto con todo su contenido (tanto Archivo como Subdirectorios). Quedando de la siguiente manera:

```
javier@inventario:~$ rm -R Directorio1/
```

Para poder ***copiar*** un archivo o directorio, se utiliza el comando ***cp***; es necesario ubicar el origen del archivo o directorio a copiar, y saber el lugar de destino que tendrá la copia. Por ejemplo, para copiar el archivo que corresponde a los passwords (passwd), ubicado en el directorio /etc/:

```
javier@inventario:~$ cp /etc/passwd  .
```

Donde se le está indicado con el punto (**.**) que copie el archivo al directorio en donde se está trabajando actualmente. Se puede copiar un mismo archivo y guardarlo en el mismo directorio:

```
javier@inventario:~$ cp  passwd  p1
javier@inventario:~$ cp  passwd  p2
```

En el ejemplo anterior, se hicieron dos copias del archivo passwd, y se guardaron con el nombre ***p1*** y ***p2,*** quedando estos dos como copias idénticas del archivo passwd.

Para ***cambiar de ubicación*** o ***mover*** un archivo se utiliza el comando ***mv***. Es necesario, como en el caso de copiar, saber el lugar en donde se encuentra el archivo a mover, y el lugar de destino para llevar a cabo la instrucción. Se muestra en el siguiente ejemplo:

```
javier@inventario:~$ mv p1 directorio1/
```

Se está moviendo el archivo ***p1*** al directorio ***directorio1/***, de tal manera que el archivo p1 ya no existe en la ubicación en la que estaba originalmente.

Pero también, se puede utilizar el comando ***mv*** para ***renombrar***. Esto es posible hacerlo, cuando no hay ni Archivo ni Directorio con ese nombre, se muestra esto a continuación, con el archivo ***p1*** renombrándolo a ***p11***:

```
javier@inventario:~/Directorio1$ ls
p1
javier@inventario:~/Directorio1$ mv  p1  p11
javier@inventario:~/Directorio1$ ls
p11
```

También se pueden crear archivos vacíos, para su posterior utilización, pueden ser archivos sin extensión o con alguna en específico, de modo tal, que Linux puede leer un archivo sin extensión. El archivo se deberá ejecutar con el programa adecuado para su visualización (de lo contrario, si se abre con un simple editor, por ejemplo una archivo de imagen, el archivo se verá simplemente como texto en código).

Se puede ***desplegar el contenido*** de un archivo con el comando ***cat***, pero el archivo deberá ser de texto, pero hay que tener en cuenta que no todos lo archivos se pueden desplegar, por que no tienen un formato adecuado, por ejemplo los archivos de tipo Binario.

Para desplegar el contenido de dos o más archivos, se puede unir (concatenar) esta visualización, como a continuación se muestra:

```
javier@inventario:~$ cat  p11  p2
```

Aquí, se observa que simplemente es necesario indicar el comando y a continuación de ésta escribir los archivos que se quieran ver.

Con ***cat*** se puede redireccionar una salida estándar (pantalla) a un archivo cualquiera con **>**. Esto es, que en vez de desplegar la información, se guarde en otro archivo.

El ***Redireccionamiento*** (**>**), consiste en enviar la salida estándar (despliegue de contenido de uno o varios archivos en la pantalla) a un archivo. Así, se tienen diferentes maneras de redireccionamiento. Se describen en el Anexo 1:

El comando ***cat*** sólo se puede desplegar el contenido de un archivo de forma continua, para poder verlo detenidamente se tiene el comando ***more,*** que permite hacer esta acción. A esto se le conoce como ***Desplegado Páginado***. Con la tecla de **Retroceso** (***Enter ↵ o Intro***) o la **B**arra **E**spaciadora se puede desplazar por el documento, pero sólo de principio a fin.

Para poder solventar este detalle, con el comando ***less*** se puede ***desplegar el documento*** de manera similar que con *more*, pero permite recorrer al archivo de principio a fin y viceversa. Conocido también como ***Desplegado Páginado (Con Regreso)***.

Para ***salir de la visualización*** del archivo, ya sea de ***less*** o ***more,*** se teclea la letra ***Q*** y se sale al prompt de la Terminal.

Cuando se desea saber la identidad que tiene el Sistema operativo, el nombre de la máquina, dominio, dirección ***IP***, etc., se tiene el comando ***hostname***. Los ejemplos se presentan el en Anexo 1.

Cuando se quiere ***terminar una sesión*** se tiene el comando ***logout***, ***log off, exit y <Ctrl> + <d>***.

Para ***imprimir información acerca del Sistema Operativo***, se tiene el comando ***uname***. Los ejemplos se presentan el en Anexo 1.

Para apagar la computadora se tiene el comando ***halt***. Éstos y otros comandos se verán en el siguiente capítulo.

Si se desea ***buscar archivos*** se tiene el comando ***find***. Esto lo hace buscando en Directorios de manera jerárquica. Continua en el Anexo 1.

1.6 PERMISOS DE ARCHIVOS

Cada archivo y directorio tiene diferentes permisos, dependiendo del tipo de usuario. ***Para cambiar los permisos de un archivo o Directorio*** se tiene el comando ***chmod,*** la asignación de permisos puede hacerse de dos formas, la primera mediante el método simbólico o la otra mediante el método octal.

Método Simbólico:

Asignación de permisos:

chmod indica al sistema el cambio de permisos.

u	Modifica los permisos del usuario dueño del elemento.
g	Modifica los permisos asignados al grupo.
o	Modificalos permisos de los otros usuarios que no pertenecen al grupo.
a	Afecta los permisos para todos los usuarios.

Continua en el Anexo 1.

Método Octal:

Con el método octal se asignan directamente los permisos en base a una tabla de validación que va numerada de 0 a 7 dependiendo de los permisos que se deseen asignar, cada permiso adquiere un valor numérico:

Asignación		Permisos
0	_ _ _	Ninguno
1	_ _ x	Ejecución
2	_ w _	Escritura
3	_ w x	Escritura y Ejecución
4	r _ _	Lectura
5	r _ x	Lectura y Ejecución
6	r w _	Lectura y Escritura
7	r w x	Lectura, escritura y ejecución

Para ***desplegar la información ordenada alfabéticamente*** se tiene el comando ***sort***. Para ordenar el contenido de un archivo en forma inversa alfabéticamente, con ***sort*** y el atributo ***-r***.

Cuando se necesita ***mostrar o ver el tipo de archivo*** se tiene el comando ***file***, y muestra si es de tipo ASCII, binario, ejecutable, etc. Continua en el Anexo 1.

Cuando en un archivo se repiten varias líneas, y se quiere ver solamente su contenido pero no las repeticiones, se tiene el comando ***uniq***. La condición es que previamente deberán estar ordenadas, o sea que la repeticiones de líneas deben ser de forma sucesiva y el comando ***sort*** ordena el archivoen orden ascendente.

1.7 EDITORES

Un Editor de texto es una pequeña aplicación que sirve para desplegar la información de un archivo, y poder ***modificar el contenido del mismo***. Teniendo para ello direntes editores, destacando dos de ellos, que son los que mas se utilizan.

PICO

pico es un programa que permite editar archivos de texto; es muy fácil de utilizar, ya que presenta un menú de comandos en la parte inferior de la pantalla y no requiere de muchos comandos, además permite el desplazamiento del cursor mediante las teclas de navegación.

```
pico nombres.txt
```

La descripción de los comandos de pico está en el Anexo 1.

VI

Un Editor que es clásico en el ambiente Unix y que se heredó a Linux es el Editor ***vi***. Con él se puede modificar el contenido de un archivo, sólo que la manera de trabajar con este archivo es un poco complicada. Para poder editar un archivo con esta opción, simplemente se teclea el ***vi*** seguido del nombre del archivo que se desea editar.

Por ejemplo:

```
vi  nombres.txt
```

En el Anexo 1 se muestra una lista con las formas y los comandos con los que se trabaja en ***vi***, cabe recordar, como se mencionó anteriormente, que Linux es de tipo *case sensitive*, o sea que, detecta cuando se esta escribiendo en minúsculas y mayúsculas.

1.8 EXPRESIONES REGULARES

Una ***Expresión Regular*** es un patrón de comportamiento de un conjunto de caracteres o metacaracteres.

Un ***Metacaracter*** es un carácter con un significado especial.

GREP

grep → es un programa usado para imprimir las líneas de un fichero que cumplan un patrón especificado. Programas instalados: ***egrep*** (enlace a grep), **fgrep** (enlace a grep) y grep.

Descripciones cortas

- ✓ **egrep** → Muestra las líneas que coincidan con una expresión regular extendida.
- ✓ **fgrep** → Muestra las líneas que coincidan con una lista de cadenas fijas.
- ✓ **grep** → Muestra las líneas que coincidan con una expresión regular

grep es muy útil si se quieren desplegar líneas o datos específicos de una archivo, por ejemplo las bitácoras. Cabe destacar que, la función de las opciones del comando **grep** varían dependiendo de la versión de Linux que se esté usando, en los ejemplos se ha utilizado la versión ***Slackware***. Se muestran algunos ejemplos de su uso en el Anexo 1.

1.9 COMANDOS BÁSICOS DE RED

Dentro del sistema operativo LINUX existen diferentes comandos que permiten mantenerse en constante comunicación con los usuarios, sin importar el lugar donde se encuentren ni el servidor donde se encuentre alojada su cuenta de correo. En el Anexo 1 se describen algunos de estos comandos:

1.10 PROCESOS

Un proceso es un programa que se encuentra corriendo dentro del servidor, cada proceso tiene un tiempo de vida que va desde el momento en que se ejecuta (al presionar la tecla de enter) hasta que termina la ejecución del proceso. Todos los procesos tienen un dueño, este dueño es el usuario que ha invocado el proceso desde el interprete de comandos, de tal forma que ningún otro usuario puede afectar las tareas que ese usuario esta realizando, debido a que no son dueños del proceso que se esta ejecutando.

Cuando se ha iniciado una sesión dentro del servidor en realidad se está ejecutando una copia del interprete de comandos que se tiene por default, a esta copia del interprete se le asigna un número el cual esta asociado al proceso en ejecución, por cada sesión que el usuario inicie se va a ejecutar una nueva copia del shell asignándole un nuevo número de identificación para este proceso.

- ✓ ***ps*** → Es un comando que permite ver los procesos en formato corto que se están ejecutando desde una sesión abierta, y que se encuentre en uso en ese momento.
- ✓ ***ps –u login*** → Muestra un listado de todos los procesos que esta ejecutando un usuario (*login*), aunque se encuentre trabajando en varias sesiones el sistema listará todos los procesos que le pertenecen a dicho usuario.

- ✓ ***ps –f*** → Entrega un listado de los procesos en formato largo que se están ejecutando en la sesión abierta por el usuario, donde se puede ver la dependencia de los procesos por el **UID** (***Identificador del Usuario dueño del proceso***), **PID** (***Identificador del Proceso en ejecución***) y el **PPID** (***Identificador del Proceso Padre del Proceso en ejecución***)
- ✓ ***ps –e*** → Muestra un listado en formato sencillo de todos los procesos ejecutándose en el servidor.
- ✓ **ps –fea** → Muestra todos los procesos ejecutándose en el servidor en formato largo. Los procesos pueden encontrarse en diferente estado dependiendo de la atención que estén teniendo en un momento en específico.
- ✓ ***ps –h*** → Se indica el estado en que se esta ejecutando los procesos.

Estados de los procesos:

Z → Zombie, cuando un proceso corre en forma independiente, incluso del shell o del proceso que lo invocó.
R → En ejecución, se encuentra en la CPU.
T → Stopped o detenido.
S → Sleeping o Durmiendo, en espera de recibir una solicitud para iniciar su trabajo.
D → Dormido sin interrupción posible, normalmente relacionado con entrada/salida.

Los **procesos** tienen dos formas de ejecutarse, en **Foreground** (***Procesos en primer plano***) y **Background** (***Procesos en segundo plano***)***.***

Una vez que un programa se está ejecutando, se le puede enviar a segundo plano presionando la combinación de teclas *ctrl+z.*

jobs → Éste comando permite ver los programas que se están ejecutando en segundo plano, nótese que a cada proceso en segundo plano se asigna un número que sirve para identificar el proceso.
jobs –l → Muestra los procesos en segundo plano con formato largo.

Es posible enviar un proceso a segundo plano en forma automática desde el momento en que éste se inicia, esto se logra colocando el caracter de *ampersand* (&) al final de la línea del comando.

Generalmente los procesos que escriben directamente en pantalla no pueden enviarse a segundo plano como el **ls**.

Para mandar un proceso a primer plano se debe de dar la instrucción ***fg #,*** donde el **#** representa el número que entrega el comando *jobs*. De esta forma se puede interactuar con los procesos mandándolos a primer o segundo plano según sean las necesidades de trabajo.

- ✓ ***kill*** → Este comando permite entre otras opciones matar procesos, los usuarios '*comunes*' únicamente pueden matar sus propios procesos.
- ✓ ***kill –9 PID*** → Mata el proceso indicado.
- ✓ ***kill –15 PID*** → Mata el proceso pero al mismo tiempo lo reinicia asignándole el mismo PID, generalmente se utiliza esta opción para procesos que necesitan estar siempre en ejecución.
- ✓ ***killall –9 comando*** → Mata el proceso que está ejecutando un comando determinado.

1.11 SISTEMAS DE RESPALDOS

Us sistema de respaldo de datos, es la conjuncion de varios archivos en uno sólo, y de manera compacta. Para ello existen algunas utlerías que permiten hacer dicho trabajo.

TAR

tar → Comando que permite aglutinar una estructura de directorios, su función principal es almacenar el contenido de un directorio en forma recursiva dentro de un archivo.

Sintaxis:

tar [opciones] archivo.tar archivo

En el Anexo 1 se decriben las principales opciones de tar.

GZIP

gzip → Permite realizar una compresión de archivos, manejando 9 niveles de compresión que van desde el –1 al –9 donde el nivel 1 es el nivel de compresión más bajo mientras el nivel -9 es el nivel de compresión más alto, por ende el más optimo de usar.

Al momento de ejecutar esta utilera el programa le añade la extensión ***.gz*** al archivo comprimido, por default maneja el nivel de compresión 6.

gzip [-n] archivo

La opción **–n** indica el nivel de compresión que se quiera realizar, como el nivel óptimo es el –9, permite realizar la compresión más alta que puede tener el archivo afectado.

Algunas opciones se describen en el Anexo 1.

1.12 ALGUNOS PROGRAMAS ÚTILES DENTRO DE LINUX

- ✓ **fortune** → Elige de una base de datos el mensaje del día como el que aparece en pantalla cada vez que se inicia una sesión.
- ✓ **date** → Muestra la hora del sistema.
- ✓ **df** → Muestra el estado de las particiones de los discos duros indicándonos el espacio utilizado y el estado libre en disco.
- ✓ **du** → Muestra el espacio utilizado entre archivos y directorios a partir del directorio desde donde se ejecutó el comando.
- ✓ **last** → Muestra las últimas conexiones de los usuarios al servidor.
- ✓ **wc** → Cuenta las líneas, palabras y caracteres contenidos en un archivo.
  wc –l → Cuenta únicamente las líneas de un archivo.
- ✓ **banner** → Este programa permite hacer carteles de tipo banner, por default tiene un ancho de 132 líneas lo que impide que el banner pueda ser leído directamente en pantalla; para ello se debe utilizar la opción **–w#** para cambiar el número de columnas que abarcará el mensaje.
- ✓ **cal** → El sistema cuenta con un calendario el cual comprende desde el año 1 hasta el año 9999, por default el comando **cal** entrega el calendario del mes en curso.

Algunos ejemplos se muestran en el ANEXO 1

1.13 SERVICIOS DE CORREO ELECTRÓNICO

El servicio de correo electrónico, se ofrece dentro de la sesión de cada usuario, siempre y cuando se tenga activado este servicio y se cuente con los permisos necesarios para enviar y recibir correo. En linux, se ofrecen algunas aplicaciones y programas que sirver como manejadores de correo electrónico.

MAIL

mail → Este comando permite enviar correo electrónico a cualquier usuario que tenga acceso a una cuenta de e-mail.

Todos los correos revisados con este programa son almacenados en un archivo llamado ***mbox*** que se encuentra en el directorio $HOME

Para poder empezar a trabajar con ***mail,***tecleando el comando sólo revisa la lista de correos de usuario y dará acceso a una lista de correo que se haya recivido, en caso de que no se tenga correo mandará un mensaje de que no hay correo en la cuenta. En el Anexo 1 se describen algunas opciones para la revisión de correo.

PINE

pine*mail*, ***pine*** permite trabajar con el servicio de correo electrónico, pero este programa tiene muchas ventajas en comparación con el comando *mail*, ***pine*** presenta un menú de opciones, configuración del servició de correo, envío de archivos adjuntos, etc.

Para ejecutar este programa basta con teclear ***pine*** desde el símbolo de sistema y presionar la tecla de enter.Todos los correos que son revisados con este programa son almacenados en un archivo que lleva el mismo nombre que el login y se encuentra en la ruta /var/spool/mail/. Esta ruta puede cambiar dependiendo de la distribución y la versión del sistema operativo que estemos usando.

1.14 Ligas

Las ligas son accesos a diferentes archivos o directorios. Existen dos tipos de ligas, una de tipo suave y la otra es de tipo dura.

La ***liga de tipo suave es un vínculo hacia un archivo pero la liga es de tamaño muy pequeño ya que sólo hace referencia al archivo que se esta ligando***, funciona como un acceso directo y su función generalmente es asociar a un programa de nombre complejo con una liga la cual tiene un nombre más fácil de recordar.

Sintaxis de la liga suave:

```
ln –s nombre_del_archivo  nombre_de_liga
```

ln → Permite generar las ligas, en el caso anterior, con la opción –s se indica al sistema que se esta generando una liga de tipo suave.

Liga dura, ***al igual que la liga suave mantiene una relación con un archivo el cual esta ligado***, la diferencia es que ***al comparar el tamaño de la liga con el archivo, ambos tienen el mismo tamaño***, ***y al editar el contenido de uno se actualiza automáticamente el otro, pero a comparación de un archivo común es que la liga esta asociada al mismo inodo del archivo con el que esta ligado***.

```
ln  nombre_del_archivo  nombre_de_liga
```

Al aplicar el comando *ln* sin opciones, el sistema interpreta el comando como la creación de una liga dura.

Hasta este punto, ya se puede empezar a trabajar de manera óptima con el Sistema Operativo Linux, y con algunas de sus tareas básicas, al comprender, analizar y utilizar las aplicaciones, comandos, archivos y directorios de este sistema. Se oberva que la usabilidad que se le puede dar a este sistema, al empezar a trabajar con el, se desempeña de forma rápida y eficaz. La administración de archivos, directorios y usuarios,se verán en el capítulo siguiente, que trata el tema de Instalación y Administración Linux.

CAPITULO II

INSTALACIÓN Y ADMINISTRACIÓN DE LINUX

2.1 ADMINISTRACIÓN DE LINUX

Para que el trabajo de un usuario con la información que se meneja en una computadora sea de manera eficiente, fácil y secilla de usar, se necesita que previo a ello exista una buena administración del sistema con el cual se este trabajando.

Es necesario que haya una o varias personas encagadas de administrar un sistema, fándole así rubutez, para ello se requiere que dicha persona personas, que sea(n) la(s) encagada(s) de dicho trabajo, que cuenten con un determinado perfil, caracteristicas, y que cumplan con una serie de requisitos para poder resolver la tarea de administrar un sistema.

Perfil y Actividades del Administrador

El ***administrador de sistemas*** es la persona responsable de *configurar, mantener y actualizar* el sistema o conjunto de sistemas que forman una red. Éste, cuida el funcionamiento del software, hardware y periféricos de forma que estén disponibles para ser utilizados por los usuarios.

La iportancia de la administración → Proporcionar un ambiente seguro, eficiente y confiable, brindar un funcionamiento confiable del sistema, se divide el trabajo entre varios administradores, dependiendo del tamaño del sistema.

Las Tareas Administrativas → Administrar usuarios, Configuración de dispositivos, Realizar respaldos, Capacitar usuarios, Asegurar el sistema, Registrar los cambios del sistema, Asesorar a los usuarios.

Actividades del administrador.

- ✓ Mantenimiento de claves de usuarios. Se les asignan '*contraseñas fuertes*', para evitar que haya alguna violación a la cuenta del usuario y/o al sistema.
- ✓ Instalación y mantenimiento de dispositivos.
- ✓ Instalación y actualización de software. Éstos pueden ser Applets para Java, PDFs, para Macromedia, para actualizar servicio de Web del usuario, de FTP, etc.
- ✓ Configuración de las interfaces de red.
- ✓ Administración de los recursos.
- ✓ Atención a usuarios. Revisar la Conexión de cada uno de los usuarios.
- ✓ Monitoreo del sistema.
- ✓ Detección de fallas. De Hardware y Software.
- ✓ Auditoria e implantación de la seguridad del sistema. Generalmente, esta actividad no la hace el Administrador, se le pide a un Auditor de Sistemas Especializado. El Sistema se puede Auditar de manera sencilla con ***nmap***.

Los Conocimientos que debe tener un Administrador

- ✓ Técnicas de programación.
- ✓ Dominio de al menos un lenguaje de programación.
- ✓ Funcionamiento del sistema operativo.
- ✓ Técnicas de administración del sistema operativo. Mediante un sistema Remoto, Local, y por medio de Scripts.
- ✓ Conocimientos básicos de hardware y mantenimiento de dispositivos.
- ✓ Comprensión profunda sobre redirección, tuberías, procesamiento en segundo plano, etc.
- ✓ Manejo de ***vi***, pues es el común denominador entre los sistemas UNIX y Linux.
- ✓ Programación shell.
- ✓ Planear las actividades.

- ✓ Guardar copias de seguridad. **Jamás modificar** sin respaldar previamente.

Un Administrador de Sistemas cuenta con Recursos y Documentación, con los cuales puede hacer una Administración Óptima, como se observa en la siguiente tabla.

Utilierías de un Sistema	Básicas: cut, sort, paste, diff, comm, tail, head, grep, egrep, compress, etc Control de tareas: at, crontab. Respaldos: dump, dd, restore, tar, cpio.
Herramientas de programación	C, shell, awk, perl
Documentación	En línea (man, apropos, info). Impresa (libros, manuales). Internet.
El hardware de la máquina. Organizar	Características, modelo, capacidad, etc. Ubicación física
Mantener canales de comunicación con los usuarios	/etc/motd, News, Wall Write, Mail, Web
También debe contar con los siguientes aspectos de tipo Administrativo	Establecer políticas, Apertura de cuentas Horas de mantenimiento Responsabilidad de los respaldos Borrado de archivos temporales Cuotas de disco, Seguridad del sistema

Tabla 2.1- Tabla deRecursos y Documentación

2.2 Instalación

Slackware Linux no requiere de un sistema extremadamente potente para ejecutarse. Se puede ejecutar desde equipos 386 o superiores.

Los requerimientos mínimos para instalar y ejecutar Slackware son:

- ✓ Procesador 386.
- ✓ 16MB en RAM.
- ✓ 50 megabytes de espacio libre en disco duro.
- ✓ Unidad de 3,5 ”.
- ✓ Hardware adicional tarjeta de vídeo para ejecutar X Windows y una tarjeta de red por si se desea tener acceso a algún tipo de red.

Nota:

Cabe mencionar, que son los requerimientos mínimos para que un sistema Linux pueda trabajar de manera austera. En el caso práctico se utilizaron PCs con procesadores Pentium IV o AMD K7, y la última versión de Slackware mínimamente necesita 3GB de espacio libre en Disco tan sólo para su instalación. La Versión de Linux Slackware es 10.0

PAQUETES

Antes de instalar Slackware se deben seleccionar la series de paquetes que se van a instalar. Linux Slackware es una de las distribuciones de Linux más antiguas, contiene un conjunto de paquetes que se reparten en series y comienzan desde la letra A hasta la Z. Esto formó un

estándar, de esta manera era más rápido conseguir las actualizaciones de Slackware ya que no era necesario descargar todo el software completo de una distribución con paquetes que no se iban a emplear, pero era mucho más rápido y fácil descargar series de paquetes individuales.

Hoy en día las distribuciones de Linux son distribuidas en CD-ROM e incluyen todos los paquetes completos. Se pueden omitir algunos paquetes en la instalación, por ejemplo, si no se va a trabajar con X Windows se puede omitir la serie de paquetes X. Se describen a continuación algunas de las Series con que cuenta Slackware:

A - Es la base, contiene suficiente software para levantar y ejecutar Slackware, contiene algunos editores de texto y programas de comunicaciones.
AP - Varias aplicaciones que no requieren del sistema X Windows.
F - Contiene FAQs, HOWTOs, y otro tipo de documentación.
GTK - Contiene el ambiente de escritorio de GNOME, biblioteca de widget de GTK, y el GIMP.
K - El código fuente para el núcleo de Linux.
KDE - Contiene el ambiente de trabajo de escritorio KDE.
N - Contiene programas para configuración de una red. Demonios, programas de correo, telnet, programas de lectura de noticias, etcétera.
T - Contiene el sistema de formato para documentos de tipo teTeX.
TCL - Contiene las herramientas del lenguaje de comandos, el Tk, el TclX, y el TkDesk.
U - Contiene paquetes de programas diseñados específicamente para trabajar solamente en sistemas de UltraSPARC.
X - Contiene la base del sistema X Window.
XAP - Contiene aplicaciones X que no son parte de un ambiente de escritorio importante. Por ejemplo: Ghostscript y Netscape.
XD - Contiene Bibliotecas, kit de la conexión del servidor, y ayuda de PEX, para el desarrollo de X11.
XV - Contiene Bibliotecas de XView, manejadores de ventanas aplicaciones de otras de XView.
Y - Contiene una colección de juegos de BSD.

Inicio de la instalación.

Después de haber elegido los paquetes se procede con la instalación de Linux Slackware, basta con insertar el CD de la distribución en la unidad de CD-ROM y reiniciar el equipo.

Cuando el sistema haya arrancado nuevamente se ejecutará la primera parte de la instalación de Linux mostrando una pantalla de bienvenida y en la parte inferior de esta pantalla el indicador **boot:**.

La pantalla que se muestra, contiene información sobre algunos parámetros extras que se pueden pasar durante el arranque del sistema, como no se necesita pasar ningún parámetro extra al núcleo, sólo se pulsa enter y de esta forma se iniciará la primera parte de la instalación.

En caso de que no se haya elegido ninguna opción en un determinado tiempo (1 minuto) automáticamente se lanzará el programa de instalación. Linux Slackware detectará automáticamente el hardware que esta instalado en el equipo. Una vez finalizada la detección del hardware aparecerá la pantalla de bienvenida a la instalación de Linux Slackware e iniciará la segunda parte del proceso de instalación.

Antes de continuar con la instalación se necesita agregar una partición de tipo nativa y una partición de tipo swap, para poder realizar modificaciones dentro del sistema se necesita tener permisos de superusuario (root), para poder obtener estos permisos se escribe root y se oprime enter. La partición de tipo nativa es la partición donde se va a instalar todo el Sistema Operativo, incluyendo software. La partición de tipo Swap, es un segmento de espacio en el disco duro reservado para uso junto con la memoria RAM, y por lo regular, casi siempre, el tamaño de memoria Swap será del doble del tamaño de la memoria Física (RAM), ayudando con esto a que el sistema tenga siempre memoria disponible para almacenar tantos datos como se requiera.

Linux cuenta con su propio fdisk, para poder crear particiones de tipo nativa y swap en el disco duro, esta herramienta es parecida al fdisk de Windows pero con muchas más y mejores características.

Linux tiene su propia forma de nombrar a los diferentes dispositivos que se encuentran instalados en la computadora, como son: discos duros, discos flexibles, unidades de CD-ROM, etcétera.

Los discos duros en especial son nombrados de acuerdo a su interfaz de conexión y al número de particiones, recordando que Linux los reconoce por directorios dentro del directorio ***dev/*** (devices). Al contrario de lo que sucede en Ms-Dos o Windows, en donde los dispositivos tales como discos flexibles, unidades de CD-ROM, discos duros y sus particiones se nombran con una letra del abecedario C:, D:, E:, ..., etc., sin importar el tipo de interfaz de conexión, esta es tal vez la forma en como los usuarios reconocen su disco duro.

Para que sea más fácil ubicar la forma en como Linux reconoce a los discos duros, enseguida se muestra una tabla comparativa entre Windows y Linux, y la forma en como reconoce cada sistema los discos duros:

Linux	Descripción
/dev/hda	Disco duro ide primario (maestro).
/dev/hda1	Disco duro primario, partición activa y partición de arranque.
/dev/sda	Disco duro scsi primario (maestro). Aquí tamien se pueden montar las memorias *USB*
Windows	Descripción.
C:	Disco duro ide primario (maestro).

Tabla 2.2- Tabla Comparativa de Reconocimiento de Discos Duros

Como el servidor sólo cuenta con un disco duro, la forma de ejecutar fdisk es la siguiente:

```
#fdisk /dev/hda
```

Una parte importante en la creación de las particiones es como será distribuido el espacio total en disco, es muy común encontrar discos duros con dos particiones, la partición ***nativa*** y la de ***swap***. **Esto no es recomendable para un equipo que funcionará como servidor ya que si por alguna razón alguno de los directorios se llegan a dañaran todo el sistema se vería comprometido y dejaría de funcionar, se recomienda para evitar este tipo de problemas crear más de una partición nativa y sólo tener una partición swap.**

Algunos directorios como /tmp, /home, /var, /usr, etc. pueden ubicarse en un lugar diferente de donde se instalará el sistema que se conoce como raíz "/", pero directorios como /bin, /etc, /dev, etc; siempre deben estar en la misma partición que el sistema "/".

En base a lo anterior, se puede decidir ubicar el directorio /home y el directorio /var en particiones diferentes a la de la raíz, por motivos de seguridad. Estos dos directorios tienden a crecer demasiado y podrían ocasionar que el sistema quede sin espacio y pueda ocurrir una denegación de servicio.

Es mucho más fácil utilizar herramientas especiales para recuperar estas particiones que tener que volver a reinstalar el sistema por completo en caso de un daño grave. El tamaño de las particiones dependerá del tamaño del disco duro y del uso que se le dará al sistema**,** y como se mencionó anteriormente, **la única partición que tendrá un tamaño predefinido será la partición**

de memoria swap que tendrá un tamaño por lo menos 2 veces a la cantidad de memoria RAM.

De acuerdo a las necesidades y características de los equipos, se propone la siguiente tabla de particiones, tomando en cuenta que la forma en como serán ubicadas aumentará el desempeño del servidor web, :La tabla de particiones para el servidor web por ejemplo uno de 20 GB, queda de la siguiente manera:

Tipo de Partición	Tamaño de Partición
Partición para raíz ***/***.	1era. partición 4000MB para el sistema /.
Partición para el directorio ***/home***.	2da. partición 4000MB para /home
Partición para el directorio ***/usr***.	3ra. partición 4000MB para /usr
Partición ***/var***.	4ta. partición 3800MB para /var
Partición *Swap*.	5ta. partición 1024 MB para swap

Tabla 2.3- Tabla de tipo y tamaño de Particiones.

La creación de particiones Linux no es algo trivial como en Windows, así es que se hará un pequeño paréntesis en esta parte del capitulo para mostrar como se crea una partición en Linux.

La herramienta fdisk, que viene con el CD de instalación, cuenta con un menú donde se puede observar cada una de las opciones con las que cuenta esta herramienta, las opciones más importantes son:

- **p** - despliega la tabla de partición actual.
- **m** - despliega el menú de las opciones.
- **d** - elimina una partición.
- **n** - crear una nueva partición.
- **t** - cambia el sistema de identificación de la partición.
- **q** - sale de fdisk sin guardar los cambios.
- **w** - escribe los cambios en el disco y sale de fdisk.

Para crear una nueva partición se oprime la tecla **n**, *fdisk* pregunta si se quiere crear una partición primaria (***p***) o extendida (***e***), como la primera partición será para el sistema "/" entonces se oprime la tecla ***p*** para una partición primaria.

Se tiene que llevar un orden en las particiones, así es que fdisk preguntará por el número de la partición (1-4), se oprime 1 después enviará un mensaje con el número del primer cilindro que se encuentra disponible sólo se da un enter para tomar el valor que se indica por omisión, enseguida preguntará por el último cilindro, el cual se puede especificar en megas de la siguiente manera **+tamañoM**. Donde tamaño es la cantidad en megas para el tamaño de la partición, por ejemplo:

+2000M - partición de 2Gb.
+3500M - partición de 3.5Gb.

Ya que se tiene la primera partición, se puede continuar con las otras siguiendo los mismos pasos.

A continuación se muestra un ejemplo para un disco de 80 GB creando la partición primaria (3 o 20 B) y dentro de ésta varias extendidas.

Espacio de Disco o partición / # de Partición Extendida	20GB de Espacio	3GB de Espacio
5 → / **(Partición raíz)**	4G	2.5GB
6 → /home	4G	50MB
7 → /home/users	2.6GB	50MB
8 → /var/spool/mail	4G	50MB
9 → Swap	256	128MB

Tabla 2.3- Tabla de asignación de espacio para instalar el Sistema Operativo

Cuando se haya terminado de crear las cuatro particiones estas aparecerán como Linux nativa, se debe cambiar el tipo de partición que será la partición Swap (**256MB**).

Se oprime la letra ***t*** para cambiar el identificador, se indica el número de la partición (5-9) que se va a modificar y se selecciona el tipo de identificador de una lista oprimiendo la tecla **t**, se elige el número de identificador, que seria **82** y se da un enter. Ahora ya se tienen las cinco particiones, se guardan los cambios con la tecla w y se continua con la instalación.

Para ingresar al programa de instalación se teclea la palabra **setup** y se pulsa enter. Enseguida aparecerá una pantalla con un menú con diferentes opciones, las cuales se pueden seleccionar con las teclas de cursor y la tecla enter para ingresar a ellas.

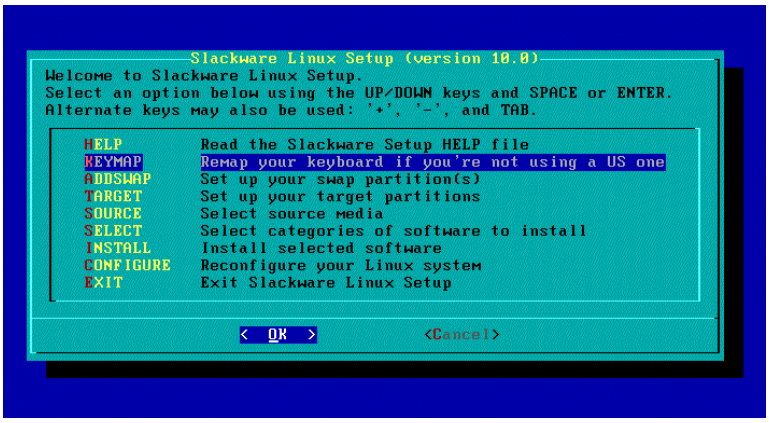


Fig 2.1- Menú de instalación

Se elige **KEYMAP** para seleccionar la configuración con la que va a trabajar el teclado. Se elige el mapa **es.map** de la lista para configurar el teclado en español y se oprime la tecla enter.

Se realiza una pequeña prueba al teclado para comprobar la configuración, después oprimir la tecla ***1*** en la pantalla y se pulsa enter, con esto se acepta la configuración.

En el siguiente paso el programa de instalación detectará las particiones swap que existan en el disco duro enviará un mensaje en pantalla con las características de esta, se elige yes y se pulsa enter para continuar, como se muestra en la figura 2.2.

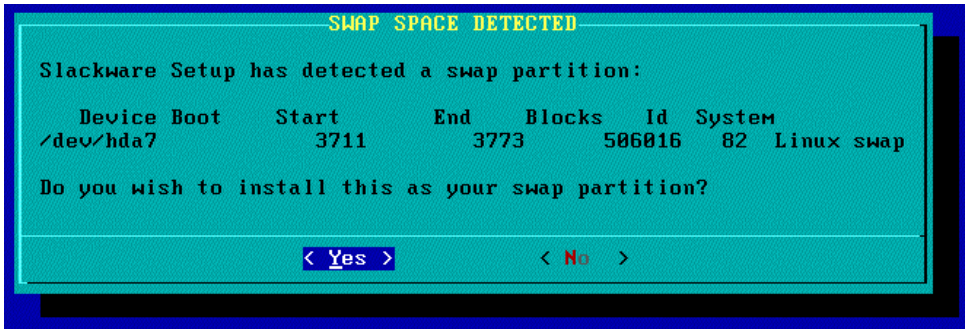


Fig. 2.2- Detección de la partición swap.

La partición **swap** se formatea y se activa para su uso inmediato, a diferencia de otras distribuciones. Linux Slackware habilita la partición swap primero antes que otras, en **caso de que el equipo donde se instala Slackware no tiene suficiente memoria RAM entonces hace uso inmediato de la memoria swap**.

Fig. 2.3- Formateando de la partición swap.

Una vez terminado el formato se agrega una línea nueva al archivo de configuración **fstab**, este archivo indica los dispositivos que serán montados al arranque del sistema, se pulsa enter para continuar.

Enseguida el programa de instalación detectará todas las particiones nativas que existan en el disco duro, aparecerá una lista con cada una de las particiones nativas y sus características, la primera partición será donde se instale el sistema "***/***", se selecciona y se pulsa enter.

En la siguiente pantalla se elige una de tres opciones para dar formato a la partición, se elege la primera opción, y se oprime enter para continuar.

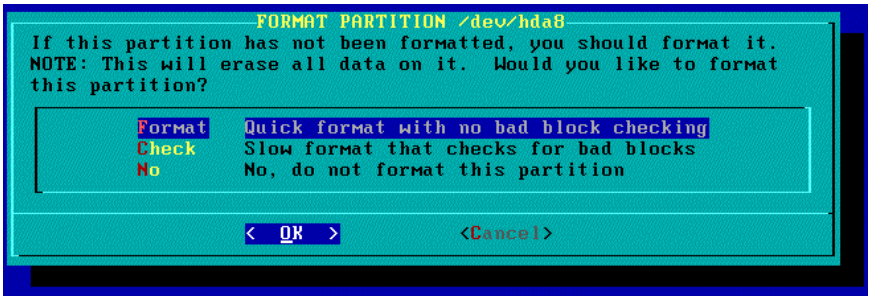


Fig 2.4- Preparación de la partición.

Al seleccionar la opción ***Check***, seguiodo de esto se necesita seleccionar la densidad del ***ínodo***, entoces se elige la primera opción, como lo indica el programa de instalación. **La partición es dividida en pequeños bloques llamados ínodos cada ínodo tiene un tamaño que puede variar, pero por omisión se utilizara el tamaño de 4096 bytes** que es el estándar en Linux, estos ínodos contienen las características de cada archivo y directorio como su tamaño, permisos,

dueño, etc. Se elige la primera opción y se oprime enter para continuar. Hecho la anterior se elige el formato de tipo **Reiserfs (Reiser File System)**, porque utiliza mejor los espacios de los inodos al trabajar con los archivos.

Después se habilitará la primera partición como partición de arranque, ahora sólo falta agregar las particiones restantes para cada una de ellas se siguen los mismos pasos que los anteriores, pero con la diferencia de que se les debe asignar un punto de montaje*. Como se vió en la tabla 2.3, la segunda partición se montará en /home y la tercera en /home/users y la cuarta /var/spool/mail.

Al final cada partición quedará lista para su uso y se agregaran líneas nuevas al archivo **fstab**, para que sean montadas al arranque del sistema.

Continuando con la instalación se debe seleccionar el lugar de donde se obtendrán los paquetes, solamente se toma la opción correspondiente en este caso, dado que se está instalando desde el CD y se pulsa enter.

Llegado el momento de seleccionar los paquetes necesarios, si se oprime la tecla espaciadora se puede agregar o quitar paquetes, por omisión la mayoría de los paquetes se encuentran seleccionados. De la lista sólo se agregaran los paquetes de la serie A, Ap, D, F, K y N. Para fines prácticos, en "**PACKAGE SERIES SELECTION**" se eligen todos los elementos que vienen para instalarse, para eso se definió anteriormente que se debía de tener un espacio de almacenamiento en disco duro de al menos 3GB para que quepan todos los paquetes de instalación. Por lo que se elige la opción ***FULL*** para que sea la instalación sin preguntas y en forma rápida, como se muestra en la figura 2.5.

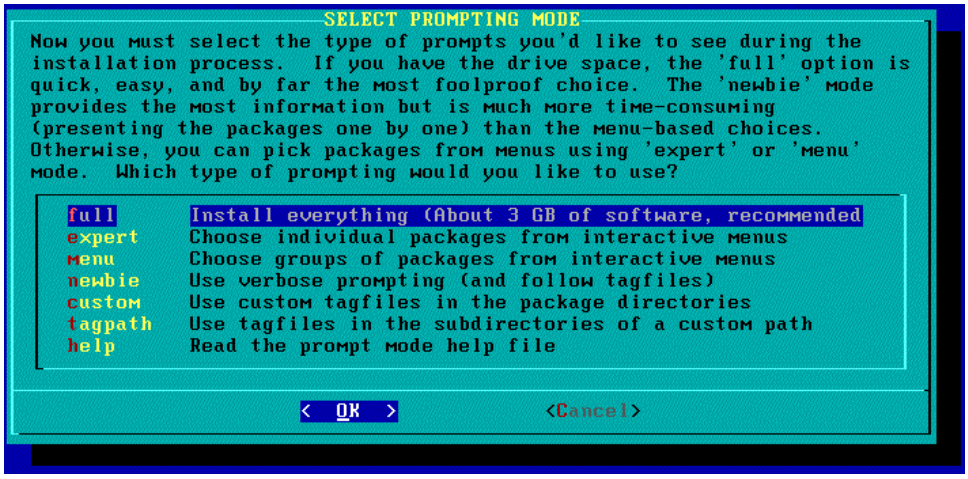


Fig. 2.5- Elección del modo de interacción.

Cuando todos los paquetes seleccionados hayan terminado de instalarse aparecerá una ventana donde se puede crear un disco de arranque, esto no es necesario y en pocas ocasiones se llega a emplear, ya que se puede utilizar el CD-ROM para arrancar el sistema y darle mantenimiento, se puede saltar esta parte eligiendo la opción de ***continue.*** Esto se observa en la figura 2.6

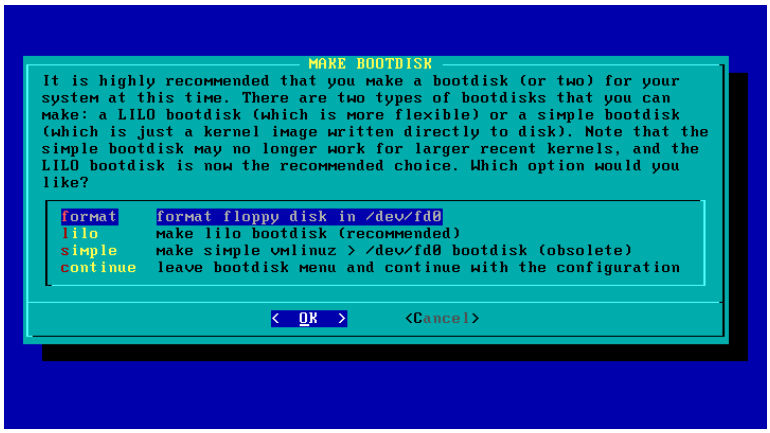


Fig. 2.6- Crear un disco de arranque

Si el equipo no contiene ningún tipo de módem ya que la conexión será a través de un enlace dedicado, se elige la opción no módem y se pulsa enter.

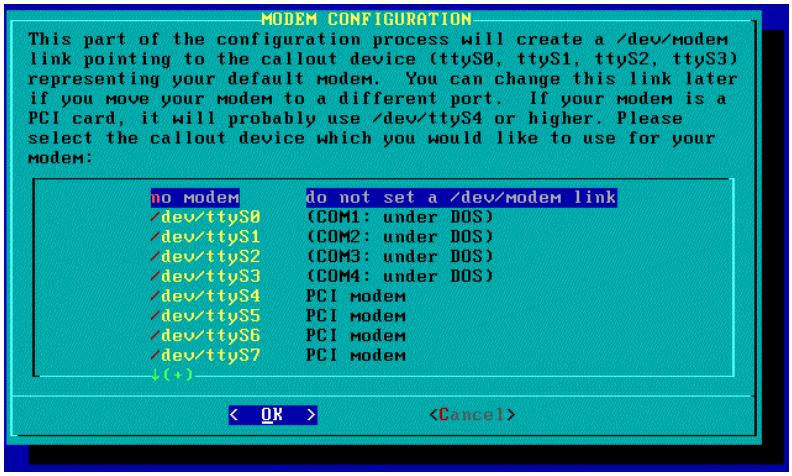


Fig. 2.7- Elección de Modem

En la pantalla de Font Configuration, figura 2.8, se puede seleccionar un tipo de letra diferente con el que se esta trabajando en la instalación, de esta forma si se selecciona un tipo diferente este tipo de letra aparecerá cuando finalice la instalación y el sistema arranque nuevamente el equipo, este paso se puede omitir solamente pulsando enter.

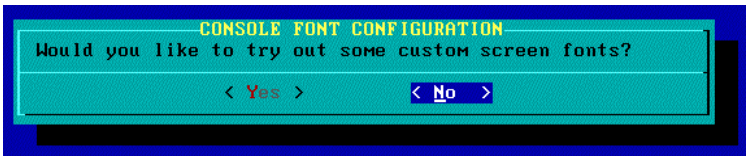


Fig. 2.8 - Selección del tipo de fuente.

La instalación de LiLo (acrónimo de **Li**nux **Lo**ader) es uno de los pasos de la instalación más importantes se debe elegir de entre dos opciones, simple y experto, se elige la segunda opción y se selecciona ok y se pulsa enter.

LiLo se alojará en el MBR (Master Boot Record) que es el primer sector donde lee la bios para saber qué S.O. debe arrancar, o en el inicio de la partición raíz, se pulsa enter para continuar.

Después, aparecerá el lugar en donde se instalará LiLo (fuigura 2.9), si se desea puede ser en **/dev/hda/**, para que desde ahí se cargue LiLo

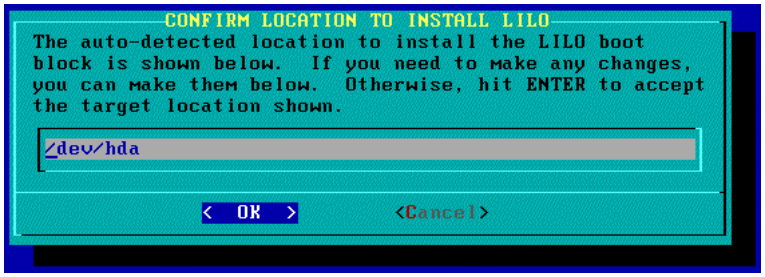


Fig. 2.9 - Ubicación del programa LILO

La siguiente ventana es la configuración de los dispositivos de red. Este paso se puede omitir para configurar la red más adelante, se selecciona No para continuar.

El siguiente paso es seleccionar el tipo de ratón instalado en el equipo que es un ratón serial, el programa de instalación lo detecta automáticamente se selecciona ok y se pulsa enter para continuar.

Para finalizar con la configuración del ratón se elige el puerto COM correcto en donde esta instalado físicamente (figura 2.10), se elige la primera opción y se teclea ok para continuar.

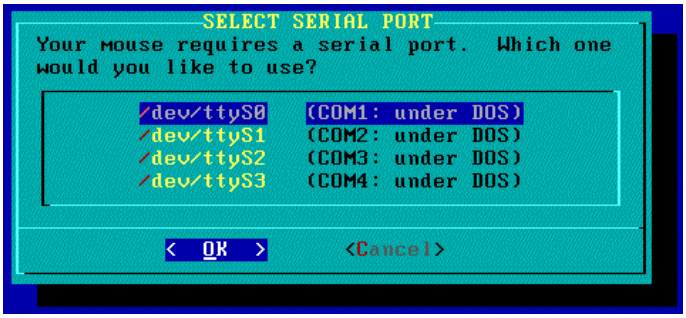


Fig. 2.10 - Elección del puerto del ratón

El GPM es un programa que permite poder trabajar en modo texto con el ratón, se puede cortar, copiar y pegar; es muy útil cuando se dictan archivos de configuración. Se selecciona Yes y se pulsa enter para instalarlo y se continúa con la instalación.

No se utilizará el servicio UTC (*Universal Time Clock –Reloj de Tiempo Universal*), así es que se elige la primera opción del menú para utilizar solamente el horario proporcionado por el sistema, se selecciona OK y se continua con la instalación.

Para configurar la zona de horario correcta para nuestro equipo se debe seleccionar uno de la lista que aparece en esta ventana, se elige America/Mexico_City, y se selecciona OK para continuar.

Para finalizar con la instalación se debe proporcionar un password al superusuario que es root, se debe elegir un password lo bastante complejo para que no pueda ser descifrado fácilmente por cualquier persona extraña que quiera ingresar al sistema, debe tener entre 6 y 8 caracteres, combinar letras mayúsculas, minúsculas, signos de puntuación caracteres especiales, etc. Una vez ingresado el password para root se selecciona YES para finalizar con la instalación.

El programa de instalación enviará un mensaje en pantalla para indicar que la instalación ha terminado (figura 2.11) y que se puede reiniciar el equipo; para poder ingresar al sistema se selecciona OK para continuar.

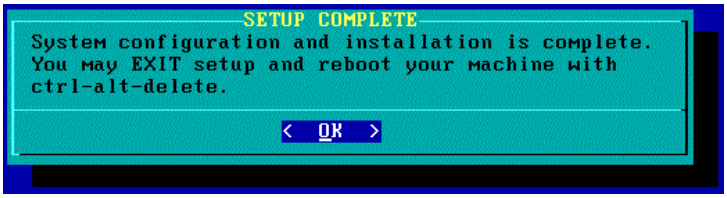


Figura2.11 - Configuración completa y listo para reiniciar.

Inmediatamente después aparecerá nuevamente la pantalla que aparece la principio de la instalación, se selecciona la opción EXIT para salir del programa y se selecciona OK.

Por último se debe reiniciar el sistema, para poder ingresar al sistema y saber si funciona correctamente, como se indica en la pantalla con las teclas de <ctrl> + <alt> + <supr>.

Con esto, el sistema arrancará nuevamente, se debe quitar el disco de instalación de Slackware de la unidad y con esto se da por finalizada la instalación.

Hecho lo anterior, la pc se reinicia y aparece el menú de selección de sistemas de LILO, y se escribe en la línea de comandos la palabra Slackware, que es el nombre que tiene el sistema (figura 2.12):

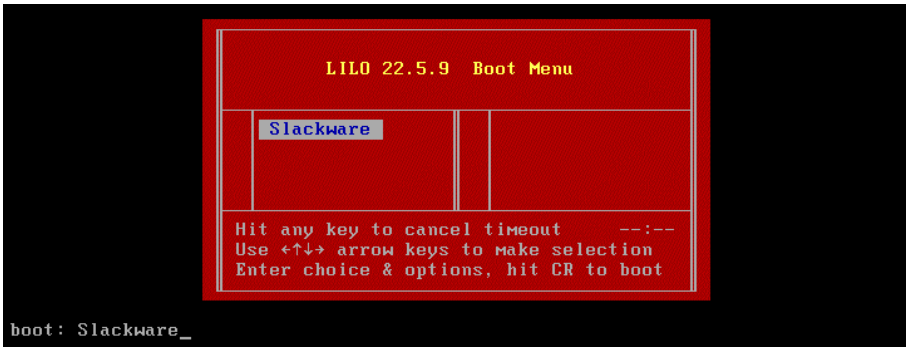


Figura 2.12 – Ventana de LILO

2.3 Configuración del sistema

Ahora que ya se tiene instalado Linux es necesario realizar algunas modificaciones al sistema antes de que este listo para funcionar como servidor de páginas web, estas tareas son:

1. **Creación de cuentas de usuario**
2. **Configuración de los dispositivos de red.**

Para realizar cualquier modificación al sistema es necesario contar con permisos de root.

Cuentas de usuario

Crear una cuenta de usuario en Linux es bastante fácil se puede hacer de tres formas distintas utilizando una herramienta gráfica, editando archivos de configuración o a través de la línea de comandos. Esta última es la forma más fácil de crear cuentas, se utiliza la herramienta ***adduser*** para este propósito.

Esta herramienta hace todo el trabajo necesario para crear una cuenta lo único que debe saber el administrador es el login y contraseña que se asignará a cada usuario, la forma de arrancar ***adduser*** es la siguiente: se ingresa al sistema con la cuenta root y en la línea de comandos se escribe adduser y se pulsa enter. Para trabajar con *adduser,* es bastante fácil y se explicará a detalle su funcionamiento posteriormente, aparte de que existen libros completos de administración que detallan el uso de esta herramienta.

En el servidor web existen varias cuentas:

1. **La cuenta personal del administrador**
2. **El encargado de la página web**
3. **La del jefe de departamento.**

Una vez creadas todas las cuentas que habrá en el sistema, es momento de continuar con el siguiente paso: configurar los dispositivos de red.

Configuración de los dispositivos de red

Para configurar los dispositivos de red que están instalados en el servidor es necesario utilizar la herramienta ***netconfig***, sólo se debe saber algunos datos como son:

- **Nombre del host**
- **Dirección IP**
- **Servidor de nombre de dominio**
- **Nombre de dominio**
- **Puerta de enlace**

Para ejecutar *netconfig* solamente se teclea el nombre de la herramienta en la línea de comandos y se pulsa enter (figura 2.13).

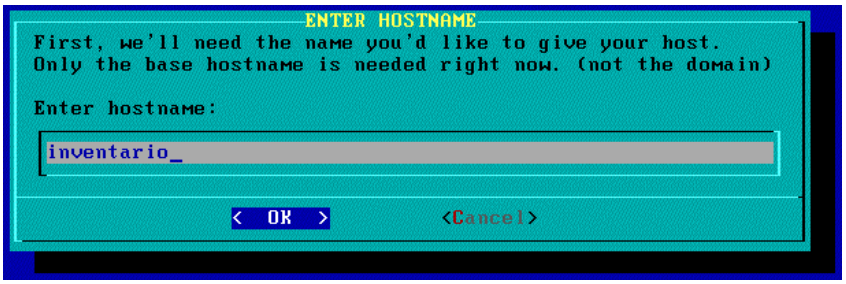


Figura 2.13 - Pantalla de inicio de netconfig.

La herramienta *netconfig* preguntará cada uno de los parámetros ya antes mencionados, sólo se debe escribir los datos correctos y se selecciona enter para continuar con la configuración.

Si se desea checar, ver o modificar la lista (etiquetas) del o los sistemas operativos que presenta LiLo, el usuario deberá de estar en **/etc/LiLo.conf**

Aparecerá una lista de la configuración de arranque de Linux y de los otros sistemas operativos (figura 2.14). Desde aquí se observa que se pueden cambiar, el Texto (Label) de las etiquetas de cada SO, que aparece al cargar LiLo, el lugar en donde esta el boot de cada sistema. Para agregar un **password** (contraseña), ya sea en cada Sistema Operativo o sólo en el que se desee, se le agregará el texto **password = "la contraseña"**, que va entre el **read-only** y **Linux…**

```
  UW PICO(tm) 4.9          File: /etc/lilo.conf

# LILO configuration file
# generated by 'liloconfig'
#
# Start LILO global section
boot = /dev/hda
message = /boot/boot_message.txt
prompt
timeout = 1200
# Linux bootable partition config begins
image = /boot/vmlinuz
  root = /dev/hda6
  label = Linux
  read-only
  password="contraseña"
# Linux bootable partition config ends
```

Figura 2.14 – Fragmento del archivo ***lilo.conf***

Hecho lo anterior, se guarda el archivo, se le dá una última vista para ver si se guardaron los cambios. A continuación, se teclea **lilo –t** para hacer un test de chequeo, que muestra que LiLo está trabajando correctamente. Aparecerá un mensaje de Alerta, indicando que LiLo esta correctamente configurado, y entonces se procederá a reiniciar la computadora para que se vuelva a cargar LiLo con las modificaciones hechas.

Reconfigurar una tarjeta de Red

Si se está trabajando en una red de tipo LAN, y se necesita saber si está funcionando la conexión o la tarjeta:

lsmod → Este comando entrega una lista de los Módulos (Drivers) que están instalados en el sistema.

lspci → Entrega una lista de los Dispositivos conectados a los sockets de tipo PCI. Muestra dispositivo, fabricante y modelo.

`/sbin/modprobe 3c509` → Para probar si el módulo de Red acepta el dispositivo 3c509, que es una tarjeta de red de la compañía *3Com*. Éstos están ubicados en ***/lib/modules//kernel/drivers/net/3c509***.

Así mismo se puede hacer con los demás dispositivos, por ejemplo:

```
/sbin/modprobe natsemi
/sbin/modprobe via-rhine
```

Son los módulos que corresponden a los dispositivos de audio y modem. Si no aparece ningún mensaje al ejectuar el ***modprobe***, significa que el dispositivo se configuró correctamente.

Para mostrar la información de los *drivers* y dependencias (que son aplicaciones necesarias para que los dispositivos trabajen apropiadamente), se encuentra en la siguiente ubicación:

```
/etc/rc.d/rc.modules
```

Dentro de este archivo es necesario habilitar la opción de **Actualización de la Lista de las Dependencias** de los módulos, con la siguiente instrucción:

```
/sbin/depmod -A
```

Se encuentra aproximadamente en la línea 38 del archivo mencionado (*rc.modules*).

Para **checar la configuración de la conexión de red** del sistema, con ***ifconfig***.

Para **configurar la Red** se tiene el comando ***netconfig***, que ya se vió con anterioridad, con esto se muestra en una pantalla de diálogo, donde se muestran el *Hostname, Domino* en la que se va a configurar la computadora, *Dirección IP (*de tipo estático*), la Máscara de Subred, la Puerta de Enlace, el Name Server (DNS).*

Para probar la conexión, se hace un *ping* a otra PC. Por ejemplo:

```
ping 132.248.75.120
```

```
root@inventario:/home/javier# ping 132.248.75.120
PING 132.248.75.120 (132.248.75.120) 56(84) bytes of data.
64 bytes from 132.248.75.120: icmp_seq=1 ttl=64 time=0.422 ms
64 bytes from 132.248.75.120: icmp_seq=2 ttl=64 time=0.166 ms
64 bytes from 132.248.75.120: icmp_seq=3 ttl=64 time=0.200 ms
64 bytes from 132.248.75.120: icmp_seq=4 ttl=64 time=0.185 ms
64 bytes from 132.248.75.120: icmp_seq=5 ttl=64 time=0.179 ms
64 bytes from 132.248.75.120: icmp_seq=6 ttl=64 time=0.175 ms

--- 132.248.75.120 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 4996ms
rtt min/avg/max/mdev = 0.166/0.221/0.422/0.090 ms
```

Para configurar la red de manera casi automática, es decir, sin necesidad de las pantallas que muestra *netconfig*, con ***ifconfig*** seguido del puerto de red para la conexión, la dirección IP que

se le asigna a la PC, la dirección de mascara de subred, la dirección de ***broadcast*** y por último la instrucción de levantamiento del servicio de conexión de red ***up***, como se muestra a continuación:

```
ifconfig eth0 132.248.75.249 netmask 255.255.255.0 broadcast 132.248.75.255 up
```

Para dar el servicio de baja: `ifconfig eth0 down`

Para dar activar o 'levantar' nuevamente el servicio: `ifconfig eth0 up`

Ahora, para que el servicio de Red tenga salida externa, por ejemplo a Internet: `route add default gw 132.248.75.254`

Si se quiere hacer la conexión primero vía MODEM, obligatoriamente, primero hay que dar de baja la tarjeta de Red, después se hace la conexión vía telefónica.

Si se quieren dar de alta dos tarjetas de red: `ifconfig eth0:1 192.168.0.10`

En la instrucción anterior, se está emulando otra tarjeta de red, dentro de la que ya se tiene instalada. Como máximo se pueden generar hasta 9 direcciones IP Virtuales.

También existen archivos o instrucciones relacionados con la configuración de red dentro del Directorio ***/etc***, ***hosts, resolv.conf, HOSTNAME***. Dentro del directorio ***/etc/rc.d/***, los archivos ***rc.inet1, rc.inet1.conf*** y ***rc.inet2***.

Para detener los servicios o apagar el Servidor, con los comandos ***halt, poweroff*** y ***reboot*** se cierran todos los servicios sin guardar ningún cambio, ***shutdown*** para apagar, además de que cuenta con otros atributos, que se pueden consultar en el manual de cada instrucción.

CLAVES DE USUARIO

En un sistema de tipo multiusuario, tiene que haber un **usuario** al que se denominará ***Administrador (root)***. Todo el poder administrativo se otorga a **root** o **súper usuario**, **root** controla a los usuarios individuales, a los grupos y los archivos (figura 2.15).

Para que un usuario *común* pueda accesar al uso de los recursos brindados por un Sistema o un Servidor, es necesario que cuente con una ***cuenta***. En un sentido más específico, una **cuenta** consta de los siguientes elementos:

- ✓ Un nombre de usuario y una contraseña válida.
- ✓ Un directorio inicial.
- ✓ Acceso al shell.

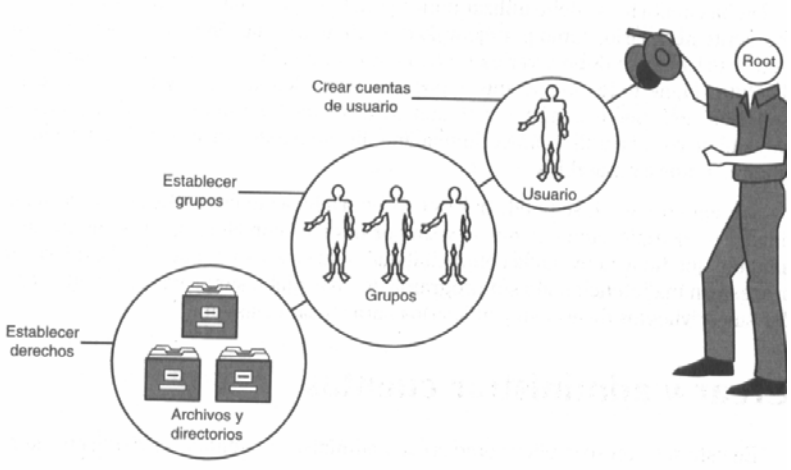


Figura 2.15 – Modelo de administración de usuarios

Cuando un usuario intenta iniciar una sesión, Linux comprueba si se cumplen estos requisitos, para lo que examinan el archivo ***passwd***. Dicho archivo

se encuentra en el directorio /etc. Un ejemplo de éstos se ve en figura 2.16 y su descipción en la tabla 2.1. Cada línea almacena el registro de una cuenta y cada registro consta de siete campos, los cuales están delimitados por dos puntos (:), username, password, userID, groupID, real name, user home y shell; los signos de dos puntos (**:**) son separadores de campos

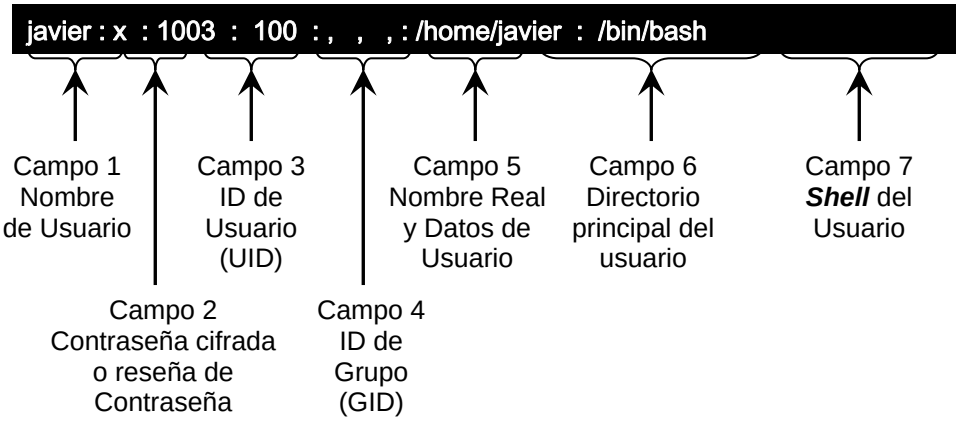


Figura 2.16 – Descipción de registro de una cuenta de usuario

Campo	**Significado**
Username	Almacena el nombre de usuario del usuario..
Password	Almacena la contraseña de acceso del usuario.
userID	Almacena el número de identificación de usuario.
groupID	Almacena el número de identificación de grupo del usuario.
Real name	Suele recibir el nombre de campo GECOS (General Electric Comprehensive Operating System) y almacena el nombre real del usuario.
User home	Almacena la ubicación del directorio de inicio del usuario.
Shell	Almacena el shell predeterminado del usuario.

Tabla 2.1

Existen varias formas de añadir usuarios:

- ✓ Utilizando herramientas gráficas.
- ✓ Utilizando herramientas de línea de comandos.
- ✓ Modificando el archivo /etc/passwd manualmente.

Si se va a editar /etc/passwd manualmente, se recomienda utilizar el editor vipw (abreviatura de vi passwd), vipw bloquea passwd mientras se realizan modificaciones, con lo que se garantiza que los cambios se realizan en forma segura. O en su defecto, si se carece de dicho editor, también se recomienda hacer una copia de respaldo, por ejemplo **/etc/passwd.respaldo**, para poder modificar el archivo ya sea con el editor ***vi*** o ***pico***.

Para ***crear un usuario***, lo tiene que hacer el usuario **root**, con el comando ***adduser***. Por default, el sistema va a crear el directorio de cada usuario dentro de directorio ***/home***.

Si el sistema carece del comando ***adduser***, se puede sustituir con el comando ***useradd***, sólo que éste trabaja de manera diferente:

```
useradd  -u 1002  -g 101  -d /home/prueba2  -s /bin/bash  prueba2
```

En el ejemplo anterior se teclea en primera instancia el comando ***useradd*** seguido de los atributos que definen el tipo de usuario que es:

- ✓ **-u #** → Define el UID correspondiente al usuario dentro del sistema, es necesario haber visto los ID's de los usuarios previamente creados, para cerciorarse que no se repitan.
- ✓ **-g #** → Define el GID que le corresponde al usuario.
- ✓ **-d /home/prueba2** → Corresponde al Directorio que se le asigna al usuario.
- ✓ **-s /bin/bash** → Es el tipo de Shell con el que trabajará el usuario a crearse.
- ✓ **prueba2** → Es el nombre, que en el ejemplo, se le asigna al usuario. Se puede observar que el nombre de usuario (***login***) se asigna al final, a diferencia de como se trabaja con *adduser*.

A continuación se deberá indicar al sistema un password para dicho usuario, ya que al ejecutar éste comando no se le agregó ninguno

Para eliminar usuarios, se puede hacer de tres formas:

- ✓ Utilizando herramientas gráficas.
- ✓ Utilizando herramientas de línea de comandos.
- ✓ Modificando /etc/passwd manualmente.

Para eliminar usuarios manualmente es necesario realizar los siguientes pasos:

1. Eliminar la entrada del usuario en el archivo de **/etc/passwd.**
2. Eliminar el **directorio** de inicio **del usuario**.
3. Cuando se elimine la entrada de /etc/passwd, se puede utilizar vipw; o también, se puede eliminar el directorio de un usuario de la siguiente forma:

```
rm –r /home/nuevousuario
```

CREAR GRUPOS

Crear grupos es una labor sencilla, basta con el comando ***groupadd*** seguido del nombre que se le va a dar al grupo a crear. Por ejemplo:

```
root@inventario:/home# groupadd prueba
root@inventario:/home# groupadd servicios
```

Los grupos se almacenarán en el archivo ***/etc/group***. La estructura de ***/etc/group*** es parecida a la de ***/etc/passwd***, El archivo /etc/group se compone de registros de grupos. Cada línea

almacena un registro y cada registro se **almacena** en cuatro campos delimitados por dos puntos (**:**).

- ✓ Group name
- ✓ Group password
- ✓ Group ID (GID)
- ✓ Group users

Todos los usuarios normales se agregan en el último campo, separados por una coma. Cuando se vaya asignar el **GID**, se debe utilizar el esquema de numeración que se ha establecido en Linux, arriba de 100 para los usuarios normales y debajo de 100 para los usuarios el sistema y ***para asignar un directorio y sus archivos a un grupo determinado*** se utiliza el comando ***chgrp***.

Para asignar un directorio y sus archivos a un dueño y grupo determinado se utiliza el comando ***chown,*** chown –r nuevousuario:grupo /home/nuevousuario.

Para eliminar un grupo, se suprime la entrada desde el archivo en ***/etc/group*** o se puede utilizar la herramienta ***groupdel***.

Así, ya se tienen los conocimientos necesarios para trabajar de una manera más dedicada con el Sistema Operativo Linux, con los comandos y utilerías. Con lo visto en este capítulo, ya es posible instalar Linux de una manera óptima, segura y rápida. Crear y administrar cuentas de usuarios y grupos. Así, como se verá en los siguentes capítulos, ya es posible empezar a explotar los recursos que nos brinda este Sistema Operativo.

CAPITULO III

ADMINISTRACIÓN DE SERVIDORES WWW CON LINUX

3.1 Administración de Servidores WWW con Linux

Un servidor ofrece servicios dentro del Word Wide Web en Internet. Un Servidor WWW: es un programa encargado de ofrecer comunicación mediante el protocolo HTTP (Hypertext Transfer Protocol). HTTP es el protocolo de red para el WWW.

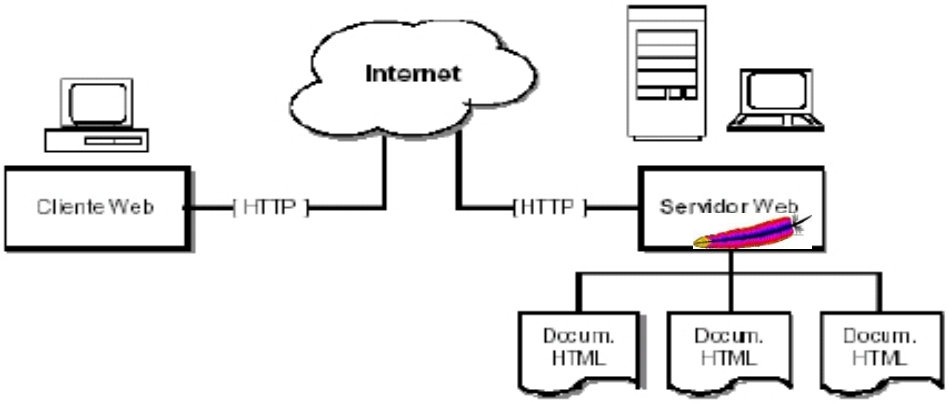


Figura 3.1 Arquitectura Cliente – Servidor (a)

Basa su operación en la arquitectura "Cliente- Servidor".

- ✓ El servidor http es el encargado de publicar "recursos" electrónicos.
- ✓ El cliente http consulta los recursos que el servidor ofrece.

La forma en la que funciona HTTP es la siguiente:

1. El cliente HTTP abre una conexión.
2. El server manda un "acknowledge" (*reconocimiento*) notificando que se ha abierto una sesión.
3. El cliente envia su "request message" (*mensaje de interrupción*) solicitando un recurso.
4. El servidor responde con "response message" (*mensaje de respuesta*) que contiene el recurso solicitado y cierra la conexión.

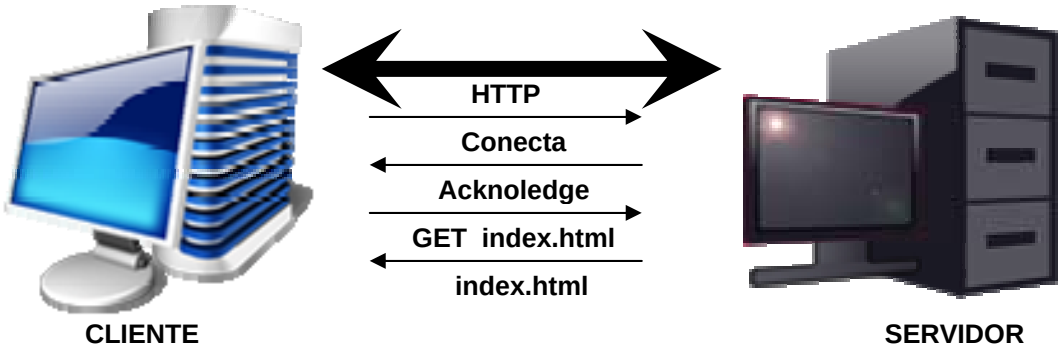


Figura 3.2 – Función HTTP

Los Servidores de WWW que más uso tienen, son:

- ✓ Internet Information Server - Microsoft
- ✓ Sun Java Web Server - Sun MicroSystems
- ✓ Roxen Web Server – Open Source
- ✓ Public Domain HTTP Daemon - NCSA
- ✓ Zeus Web Server - Zeus
- ✓ Apache Web Server – Open Source

3.2 APLICACIONES SERVIDOR WEB

Para entender bien el concepto de "servidor de aplicaciones" es imprescindible tener una visión histórica de la evolución de los servicios de Internet.

Tal y como se muestra en la figura 3.3, hace unos años, el hacer una página web estática y colocarla en Internet resultaba suficiente para todo el mundo, lo que reflejado en el entorno Linux significa que bastaba con instalar un Linux en Internet y un Apache como servidor de páginas HTML.

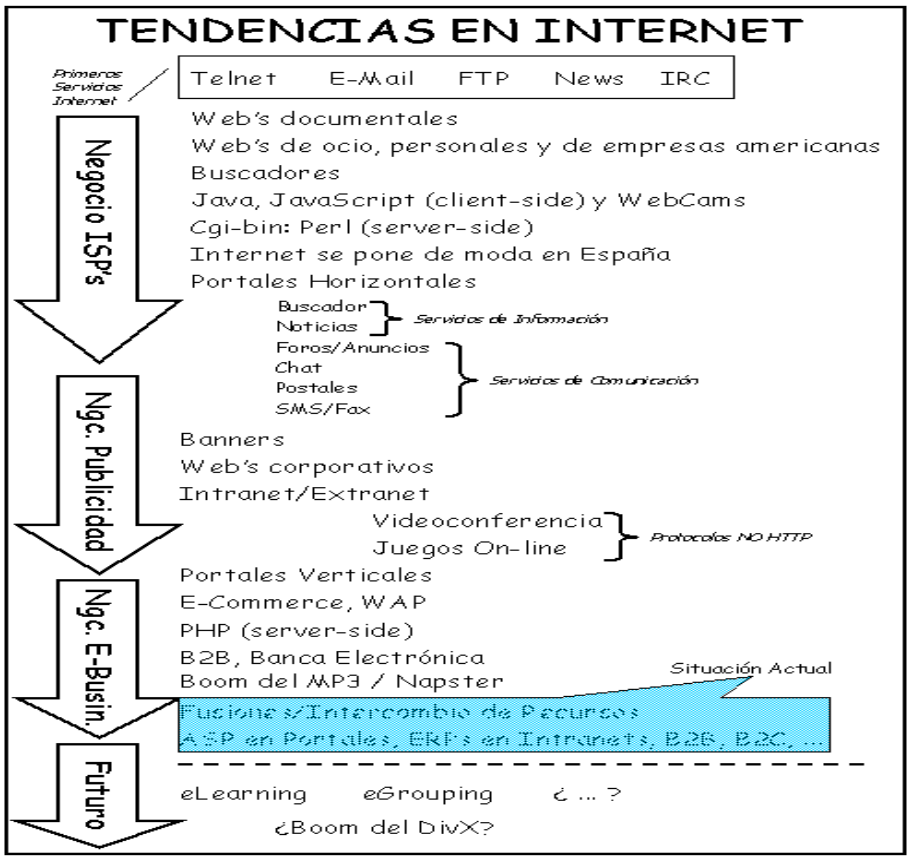


Figura 3.3 – Tendencias de Inernet

Sin embargo, nuevas funcionalidades y usos han ido apareciendo en Internet a lo largo de los últimos años, de forma que hoy en día es difícil encontrar websites en Internet completamente desarrollados con páginas estáticas. El que no tiene un *applet* de *Java*, tiene un *código JavaScript* o un *script en Perl* para realizar una determinada función. Últimamente y cada vez más, la extensión .html ha dejado de verse, para encontrar .phtml o .php, indicando que todo el website trabaja sobre un lenguaje dinámico, PHP. Así se consiguen algunas ventajas importantes como la actualización de contenidos por Web, la extracción de datos y resultados de búsquedas desde bases de datos, la personalización de portales, la actualización de diseños de forma modular, etc. De nuevo, si se refleja esta idea en el mundo de Linux, lo más probable es que se acabe hablando de ***LAMP = Linux + Apache + MySQL + PHP***, plataforma ideal para desarrollar entornos dinámicos en Internet.

Y es que la situación no se queda ahí. O quizá sí, para los que se conforman con una sencilla página web, pero no para las empresas que desean convertir Internet en una herramienta de trabajo. Cuando se avanza en el concepto de "web dinámica" se aprecian las muchas posibilidades que se van a presentan. Ya hay aplicaciones trabajando en Internet que hasta hace bien poco sólo se veían corriendo en el propio entorno gráfico preferido (como, por ejemplo, una agenda/organizador): sin ir más lejos *Yahoo* tiene este excelente servicio en su portal, y no requiere que se instale absolutamente ningún software en la computadora, la que por cierto, puede ser una PC bajo Linux como una Sun o un Windows XP sobre un 686. Conclusión: al hacer dinámica la web, se pueden crear aplicaciones multiplataforma, con soporte garantizado por el proveedor de Internet que aloja el servicio, y sin problemas de instalación o cambio de equipo.

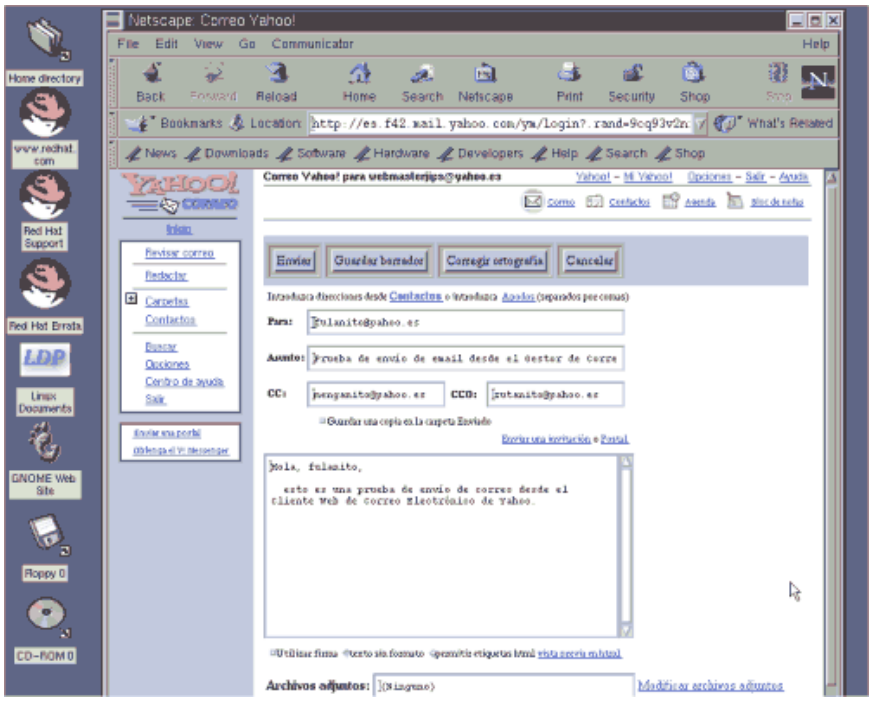


Figura 3.4 – Modelo de página Web dinámica

Cuando se habla de una agenda electrónica (http://es.yahoo.com/r/ha), de un cliente de correo electrónico (http://es.yahoo.com/r/fz), de un lector de *news* (http://groups.google.com/), una enciclopedia on-line (http://encarta.msn.es/), un traductor (http://world.altavista.com/), videojuegos (http://es.games.yahoo.com/), etc.; son programas por Internet residen en un "***servidor de aplicaciones***" (figura 3.4).

Nota: Al igual que hay servidores de aplicaciones en Internet, los hay también en las Intranets de algunas empresas.

Dicho esto, el concepto de "servidor de aplicaciones" queda un poco desmitificado, puesto que enseguida se precibe que simplemente se trata de unas páginas web dinámicas que se comportan como una aplicación que corre en un navegador. O sea, que un servidor de aplicaciones es un servidor web.

EJEMPLOS DE SERVIDORES DE APLICACIONES WEB CON ALGUNAS CARACTERISTICAS

APACHE:

Apache es uno de los Servidores de páginas más utilizados, posiblemente porque ofrece instalaciones sencillas para sitios pequeños y si se requiere es posible expandirlo hasta el nivel de los mejores productos comerciales. Si se utiliza para un sitio pequeño que sólo contenga archivos en HTML, esto es, no requiera de aplicaciones de servidor su funcionalidad es excelente.

¿Pero qué sucede cuando se requiere una aplicación de Servidor? La aplicación de este tipo implica lo siguiente:

Cuando el servidor de páginas (Apache) recibe la requisición para "x" página, éste reconoce cuando debe enviar un documento estático (HTML) o ejecutar algún tipo de aplicación, por ejemplo, supóngase que hay una solicitud, que "x" página invoca (llama) un programa en Perl y éste a su vez solicita información a una base de datos, por lo tanto para llevar acabo esta operación debieron iniciarse 2 procesos nuevos, quizás esto no sea de gran importancia para un sitio de 100 visitas diarias. ¿Pero qué sucedería con uno de 2 visitas por segundo?

Si no se tienen los suficientes recursos en cuanto a memoria y procesadores se refiere, seguramente caerá el servidor de páginas o bien se queme el "Host" (computadora física) por la demanda excesiva.

Apache tiene tanto tiempo de desarrollo que han sido realizadas diferentes soluciones para evitar estas ineficiencias, algunas de ellas:

- ✓ Es capaz de utilizar otros interpretadores y lenguajes como "Tcl", "PhP" y "Python"
- ✓ Puede conectarse directamente a una Base de datos.
- ✓ Entre otras, posee diversos módulos que le permiten utilizar una gran gama de lenguajes y desarrollar funcionalidades avanzadas.

Cabe mencionar que muchos sitios de alto tráfico aún permanecen bajo este tipo de arquitectura, en ocasiones si se tienen los recursos suficientes continua siendo costeable esta metodología a migrar a otro tipo de desarrollo, sin embargo, siempre es conveniente conocer otras alternativas.

IIS*:*

Es el servidor de páginas desarrollado por Microsoft para Windows NT/2000, a diferencia de los dos servidores de páginas mencionados anteriormente, IIS sólo puede operar en plataformas Windows. El punto más favorable de este servidor son ASP's que facilitan el desarrollo de aplicaciones y la "sencillez" de instalación, sin embargo, existen alternativas como ADP's de Aolserver y JSP's para Java. Desafortunadamente debido a la presencia de Microsoft en el mercado seguirá siendo necesario interactuar con este producto a pesar de todas sus desventajas:

Plataforma: Sólo esta disponible para Windows. Historia de Sistemas Operativos para Red y Porque es mas fácil y Económico configurar Unix que Windows en Red

Costo: Porque pagar licencia si existen productos flexibles y open-source mejores.

Confiabilidad: Menos confiable que otros productos, tan confiable que ni sus mejores técnicos podían utilizarlo cuando se encontraba bajo uno de los tantos ataques que sufren sitios de Internet: 26 de Enero del 2001 http://www.nytimes.com/2001/01/26/technology/26SOFT.html

Seguridad: Aún plagado de fallas en versiones de producción: 19 de Junio del 2001 http://news.cnet.com/news/0-1003-200-6312870.html

Seguridad: Más fallas en versiones de producción: 30 de Octubre del 2002 http://www.eweek.com/article2/0,3959,661891,00.asp

Los Criterios de Selección que hay que tener en cuenta para un servidor son:

- ✓ La Función del Servidor de web.
- ✓ *Expertise* (destreza) de los administradores.
- ✓ Plataforma disponible.
- ✓ Número de conexiones concurrentes.
- ✓ Número transacciones por Segundo.
- ✓ Costo computacional por transacción.
- ✓ Proyección del crecimiento esperado.
- ✓ Soporte para la tecnología utilizada para el desarrollo.
- ✓ Análisis del retorno de Inversión.

¿Por qué elegir un Servidor Apache? Esta pregunta es fácil de responder, debido a la gran cantidad de documentación y sobre todo ***usuarios*** que alrededor del mundo han utilizado este tipo de servidor, A continuación se mencionan algunas de sus características:

- ✓ Es Robusto, da soporte de un gran número de transacciones.
- ✓ Proporciona flexibilidad y facilidad para la Gestión de Unix.
- ✓ Es Configurable para diferentes entornos de trabajo. Cuesta poco configurarlo y ponerlo a funcionar.
- ✓ Con un alto nivel de seguridad.
- ✓ Disponible para una gran variedad de plataformas.
- ✓ Soporte para servicio de proxy.
- ✓ Soporte para granjas de servidores.
- ✓ Soporte para Scripting Languages integrados como módulos (por ejemplo PHP, mod_perl). Proporciona una gran cantidad de herramientas para crear sitios Web medianamente sofisticados.
- ✓ Incluye el código fuente del servidor
- ✓ Soporte para accesos restringuidos
- ✓ Soporte para SSL (Secure Socket Layer)
- ✓ Y además… **Es Gratuito**.

Otras de sus cualidades son que es invulnerable a Virus (Red Code, Nimda), Brinda un buen nivel del sistema de seguridad, contra Hackers, Buffer Overflows, que no es basado en Windows (Inestable, Inseguro, Tecnología propietaria, Imposible portar).

Actualmente, dadas las características ya mencionadas, Apache se manifiesta como un sólido servidor para Web, ya que en los últimos 10 años ha tenido un despunte verdaderamente significativo en comparación de los démas programas de servidores existentes en el mercado.

3.3 INSTALALACIÓN DEL SERVIDOR APACHE

Antes de comenzar con la instalación de éste servicio, hay que asegurarse de que no hay ningún servicio de Apache en ejecución (‘corriendo’), porque posiblemente, puede ser que ya se tenga un Servidor Apache instalado. En caso de ser afirmativo se procede como se indica en los pasos a y b, en caso contrario, se pasa al **Proceso de Instalación**.

a) Con el siguiente comando, despliega una lista con los servicios que se esten manejando de Apache: **ps -fea | grep httpd**

b) Si se dá el caso, de que haya un servidor activo se procede a detenerlo con el comando ***apachectl***, que es con el que el servicio de Web está trabajando, seguido de la opción ***stop***: **apachectl stop**

Con esto se detendrá el servicio que ofrece Apache, y se puede proceder con la instalación.

PROCESO DE INSTALACIÓN

Para obtener el programa de Servidor Apache, se consigue en la siguiente dirección http://www.apache.org, o desde un CD de instalación del Servidor de Apache.

Al descargar el programa de Servidor Apache, el archivo ***httpd-2.0.52.tar.gz***, que es la versión que se utilizó para este caso, actualmente esta la versión. Ya descargado el programa, se tiene también que descargar el archivo ***httpd-2.0.52.tar.gz.md5.*** Este archivo es de tipo ***md5 (Message Digest #5)***, que es un Programa para comprobar Certificado de Archivos.

Ya descargado este último, se tiene que hacer una comparación entre el certificado del ****.md5*** con el certificado que trae el programa del servidor. Se muestra el ejemplo como sigue:

```
cat httpd-2.0.52.tar.gz.md5
```

A continuación se da la instrucción para que el certificado del programa ***httpd-2.0.52.tar.gz*** se muestre con el comando ***md5sum*** (md5sum es un programa para comprobar y crear archivos MD5), éste es un algoritmo que se suele utilizar para realizar la comprobación de la integridad de ficheros binarios, siendo muy utilizado, por ejemplo, para la posterior verificación de imagenes ISO o programas descargados de Internet:

```
mdsum httpd-2.0.52.tar.gz
```

Cuando se hace la comparación de Certificados, simpre tiene que coincidir, garantizando que el programa a instalar estará libre de fallas y/o errores, de lo contrario el archivo se descargó erróneamente.

HTTP Daemon. Es un programa que corre de fondo en el servidor Web y espera peticiones externas al servidor. El ***daemon*** responde las peticiones y sirve los hipertextos y multimedios a través de Internet usando HTTP.

Hay que checar, el archivo de instalación que viene incluido dentro del archivo ****.tar.gz***, y analizar como se instala el servidor, esto va a depender del Sistema Operativo y el kernell (para las versiones de Unix y Linux) que se esté utilizando.

Para descomprimir el archivo ***httpd-2.0.52.tar.gz***, se utliza el comando tar como sigue:

```
tar -zxvf httpd-2.0.52.tar.gz
```

Hecho lo anterior, se le está dando la instrucción de que se descomprima, se desagrupe tar, se imprima el proceso de la información en pantalla y se indique que es un archivo. La descripción del comando ***tar*** y sus opciones o atributos se describieron en el capítulo I.

Seguido de esto se genera un directorio con el nombre que tiene el archivo compreso, esto es que el directorio se llamará ***httpd-2.0.52/***. A continuación ***se mueve el usuario dentro del directorio que se acaba de crear***, y se busca el archivo INSTALL.txt o el archivo README.txt, en donde se encuentra la forma de instalar el paquete del Servidor. Para los siguientes pasos el usuario deberá permanecer dentro del directorio creado, ya que ahí es donde se encuentran los archivos y comando que tiene que ejecutar para la instalación del servidor Apache.

Con ellos, el usuario verificará que se este procediendo a realizar una instalación confiable, además de que muestran diferentes opciones para hacer dicha instalación.

A continuación, como se indica en *la forma más confiable de instalación*, se ejecutará la ***compilación del paquete*** mediante el comando ***./configure***, de la siguiente manera:

```
./configure  --prefix = /usr/apache2
```

Con ello se le indica al programa que compile todos los archivos, y con el ***prefix*** se le está diciendo que tiene que copilarlos en una ruta determinada, en el ejemplo es ***/usr/apache2***.

Después, es necesario ***generar los archivos de tipo binario***, que serán los archivos previos que se ejecutarán para empezar la instalación. Simplemente tecleando el comando ***make*** y ↵'*enter'*. A continuación, ya que se han generado los archivos binarios, se procede la instalación del paquete con el comando ***make install***. Se comenzarán a instalar todos los archivos referentes a el Servidor de Apache, para que éste comience a funcionar.

Al terminar de instalarse, aparecerá un mensaje de que la instalación se realizó satisfactoriamente. Entonces se tiene que ubicar el usuario dentro del directorio en donde se instaló el Servidor, que en el caso de la práctica fue en ***/usr/apache2/***, donde:

- ✓ **/usr/apache2/bin/** → Es donde se encuentran los archivos binarios (ejecutables) para que comience a funcionar el servidor Apache. El usuario deberá ser el propietario del directorio para que pueda ejecutar dichos archivos, normalmente suele ser el usuario Administrador de Sistema (***root***) quien ejecuta dichos archivos y tiene permisos especiales.
- ✓ **/usr/apache2/conf/** → Archivos de configuración, referentes al Servidor Apache
- ✓ **/usr/apache2/htdos/** → Es el lugar en donde se almacenarán los archivos que sean de tipo público, que serán visualizados a través de la red, como páginas Web.

Existen otros directorios referentes a librerías, manuales, etc.

Para dar el servicio de '*alta'* del Servidor Apache, el usuario deberá ir a donde se encuentran los archivos de ejecución ***/usr/apache2/bin/***, ya estando dentro de dicho directorio, se dará la instrucción para comenzar con el Servicio de Apache, como se muestra a continuación:

```
cd /usr/apache2/bin/
./apachectl  start
```

Para checar que el servicio está habilitado, se muestran los servicios de httpd que se están ejecutando en ese momento con:

```
ps  -fea  |  grep  httpd
```

Si el servicio está ejecutándose correctamente, se mostrarán los servicios que ofrece el demonio httpd:

Para imprimir las conexiones de Red, Tablas de Ruteo, Estadísticas, Conexiones en máscara y multicast memberships se utiliza el comando ***netstat***.

Para ***mostrar que puertos de salida*** están habilitados con el comando ***nmap.***

Para revisar como están configurados los parámetros del Servicio de Apache, se encuentran dentro del archivo ***/usr/apache2/conf/httpd.conf***. Previamente se hará una copia de respaldo de dicho archivo, para evitar posibles daños al servicio que brinda Apache.

3.4 DIRECTIVAS

Una Directiva, es una Opción de Configuración de alguna Funcionalidad que ofrece algún Servicio dado.

Apache es administrado por más de 200 directivas las cuales permiten que determinada funcionalidad pueda ser incluida. En Linux el administrador controla que directivas estarán disponibles de acuerdo a los módulos con los que se compila Apache. Las Directivas se Administran dentro del archivo ***httpd.conf***.

En los Grupos de Directivas, la configuración de apache mediante directivas es clasificada en tres grupos:

- ✓ ***Global Environment,*** administra las directivas generales de operación para Apache.
- ✓ ***Main Server,*** administra las directivas del servidor principal o estándar de Apache.
- ✓ ***Virtual Host,*** administra las directivas donde los mismos procesos de apache soportan diversas Ips o nombres de dominio.

Para la versión de Apache 2.0, *ServerType* es standalone, para que no se genere tanto gasto de recursos.

Para comenzar a trabajar con las Directivas, es necesario deterner el Servico de Apache, e ingresar al archivo **httpd.conf,** para la práctica se generará un usuario llamado ***webmaster*** y un grupo llamado ***web***.

```
ps -fea | grep httpd
adduser webmaster
groupadd web
cd /usr/apache2/bin/
./apachectl stop
cd ../conf/
pico httpd.conf
```

GLOBAL ENVIRONMENT

Dentro de la Sección ***Global Environment***, se definen las siguientes Directivas:

User

Esta directiva define el ***user ID*** mediante el cual apache operará.

Valor por Default: User #-1
Se puede usar en: Server config, virtual host

Group

Esta directiva define el group ID mediante el cual apache operara.
Valor por Default: User #-1

Dentro del archivo *httpd.conf*, se localiza el lugar donde se menciona el usuario y el grupo con se va a afectar con las Directivas *User* y *Group* (aproximadamente línea 267), se modificará como sigue:

```
User webmaster
Group web
```

Se guardan los cambios hechos al archivo. Se ubica ahora dentro del directorio **usr/apache2/bin/**, y se realiza un test para verificar que este correctamente configurado el servicio de apache. Si está correcto, se mostrará un mensaje de confirmación.

```
cd /usr/apache2/bin/
./apachectl configtest
Syntax OK
```

A continuación, se debe de habilitar el servicio de Apache, se tendrá que verificar quien es ahora el nuevo Usuario al que esta dado dicho servicio.

```
./apachectl start
ps -fea | grep httpd
```

Listen en 2.0 (Reemplazando a Port de las versiones anteriores)

Esta directiva define cual es el puerto en que operará el servidor de Web.
El Valor por Default: 80

```
#Listen 12.34.56.78:80
Listen 8080
listen 8081
Listen 8082
```

Para esta Directiva, es necesario Reiniciar primero el Servidor. Se ejecuta el comando **netstat –antp**. Se modifica el puerto en el archivo *httpd.conf*, en la opción de ***Listen*** (aproximadamente línea 206). Se guardan los cambios hechos a dicho archivo. Se ejecuta un test al servicio y se reinicia el servicio.

```
./ apachectl configtest
./apachectl start
```

```
netstat -antp
```

ServerAdmin

Esta directiva define el correo electrónico del administrador del servidor web e indica la dirección a incluirse en los mensajes de error que serán enviados al cliente.

Dentro de *httd.conf* hay que escribir el **e-mail** que tiene el Administrador del Sistema (aproximadamente línea 277):

```
ServerAdmin javiervm@nombre.servidor.dominio
```

DocumentRoot

Esta directiva define la ruta absoluta donde se almacenaran los archivos html que se desean publicar.

Valor por Default: /usr/local/apache/htdocs
/var/www/html
Se puede usar en: Server config, virtual host

El Valor que aparecerá, según la configuración que se está manejando para la práctica es **/usr/apache2/htdocs/,** y se muestra como sigue:

```
DocumentRoot "/usr/apache2/htdocs"
```

ServerRoot

Esta directiva define la ruta absoluta donde los directorios conf y logs podrán ser encontrados.

Valor por Default: /usr/local/apache
Se puede usar en: Server config

```
ServerRoot "/usr/apache2"
```

MinSpareServers

Esta directiva define cual es el número mínimo de procesos servidores que son mantenidos en *spare*.

Se monitorea el número de conexiones en espera si ésta es menor a "MinSpareServers" se levantan los *deamons* necesarios.

Valor por Default: 5

MaxSpareServers

Esta directiva define cual es el número máximo de procesos servidores que son mantenidos en spare.

Se monitorea el numero de conexiones en espera si ésta es mayor a "MaxSpareServers" se eliminan los deamons necesarios, siempre y cuando no esten realizando alguna tarea.

StartServers

Esta directiva define cual es el número máximo de servidores que se iniciaran cuando se arranca apache.

Valor por Default: 5

MaxClients

Esta directiva define cual es el número máximo de procesos servidores que como máximo podran ser "levantados" o iniciados.

Valor por Default: 20

ErrorDocument

En caso de que un error o problema ocurra con Apache cuando se solicite un recurso, este puede ser configurado para que haga una de las siguientes acciones:

1. Enviar un mensaje de error (default).
2. Enviar un mensaje personalizado.
3. Redirigir a un URL local para manejar el problema o error.
4. Redirigir a un URL externo quien manejara el problema o error.

CÓDIGOS DE ERROR

El código de estado es un número de 3 dígitos que indica si la petición ha sido atendida satisfactoriamente o no, y en caso de no haber sido atendida, indica la causa. Los códigos se dividen en cinco clases definidas por el primer dígito del código de estado. Así tenemos:

Cláse de Código	Números de Código
1xx: Informativo. La petición se recibe y sigue el proceso. Esta familia de respuestas indican una respuesta provisional, está formada por la línea de estado y las cabeceras. Un servidor envía este tipo de respuesta en casos experimentales.	➢ 100, continuar. ➢ 101, cambio de protocolo.
2xx: Éxito. La acción requerida por la petición ha sido recibida, entendida y aceptada.	➢ 200, éxito. ➢ 201, creado. ➢ 202, aceptado. ➢ 203, información no autoritativa. ➢ 204, sin contenido. ➢ 205, contenido reestablecido. ➢ 206, contenido parcial.
3xx: Redirección. Para completar la petición se han de tomar más acciones.	➢ 300, múltiples elecciones. ➢ 301, movido permanentemente. ➢ 302, movido temporalmente. ➢ 303, ver otros. ➢ 304, no modificado. ➢ 305, usar *proxy*.

4xx: Error del cliente. La petición no es sintácticamente correcta y no se puede llevar a cabo.	➢ 400, petición errónea. ➢ 401, no autorizado. ➢ 402, pago requerido. ➢ 403, prohibido. ➢ 404, no encontrado. ➢ 405, método no permitido. ➢ 406, no se puede aceptar. ➢ 407, se requiere autentificación *proxy*. ➢ 408, límite de tiempo de la petición. ➢ 409, conflicto. ➢ 410, *gone*. ➢ 411, tamaño requerido. ➢ 412, falla una precondición. ➢ 413, contenido de la petición muy largo. ➢ 414, URI de la petición muy largo. ➢ 415, campo *media type* requerido.
5xx: Error del servidor. El servidor falla al atender la petición que aparentemente es correcta.	➢ 500, error interno del servidor. ➢ 501, no implementado. ➢ 502, puerta de enlace errónea. ➢ 503, servicio no disponible. ➢ 504, tiempo límite de la puerta de enlace. ➢ 505, versión de protocolo HTTP no soportada.

PidFile

La directiva PidFile define el lugar donde se almacenará el archivo donde se guarda el Process Id para el *deamon* "**padre**".

PidFile *file*
Default *file*: logs/httpd.pid

Nota: Si este archivo se pierde es común tener problemas cuando arranca o se detiene apache vía el script en ***/etc***.

Analizadores de Bitácoras

[1]Sin duda, las bitácoras forman un papel importante cuando de Administración de Servidores se trata ya que, mediante ellas es posible observar el comportamiento de un programa determinado o el estado del servidor, otorgándose con ello algunas facilidades, entre las principales:

1. Capacidad de depuración
2. Estadísticas
3. Monitoreo

[1] http://www.esemanal.com.mx/articulos.php?id_sec=5&id_art=689&id_ejemplar=50

Para explicar cada una de ellas se utilizarán como ejemplo las bitácoras generadas por el servidor Web Apache que normalmente tendrán dos archivos: ***access_log*** y ***error_log***. Como sus nombres lo indican, el primero contendrá toda la información de los accesos realizados al sitio y el segundo los errores generados por el servidor.

Capacidad de depuración. Ya sea que se estén realizando los primeros pasos en la configuración de un servidor Web o que se levanten nuevos servicios, e incluso para determinar ligas muertas en el sitio. Por otra parte, si alguien desea consultar una página que no existe en el sitio Web, en el archivo ***access_log*** se generará una línea como la que se muestra a continuación:

```
visitante.com - - [28/Oct/2003:09:52:27 -0600] "GET /noesta.html HTTP/1.1" 404 2294
visitante.com - - [28/Oct/2003:09:52:27 -0600] "GET /noesta.html HTTP/1.1" 404 2294 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
```

En esta liga es posible apreciar que aparece HTTP/1.1 404, en el que el 404 es el código de error para una página no encontrada, de la misma forma en el archivo error_log se generará la siguiente entrada:

```
[Tue Oct 28 09:52:27 2003] [error] [client 123.123.123.123] File does not exist: /var/www/misitioweb/noesta.htm l
```

Cabe señalar que lo anterior servirá para corregir las ligas que apunten a una página llamada noesta.html inexistente en el servidor.

Estadísticas → Igual que en los medios de comunicación, en los sitios Web, no hay razón de existir si no tienen audiencia, de allí la importancia de contabilizar a los visitantes que recibe nuestro sitio, qué secciones visita, de dónde provienen, etc. Estos datos se obtienen también de las bitácoras generadas por el Apache, para ejemplo se observan las siguientes líneas del access_log:

```
seisamex-servnet.serv.net.mx - - [28/Oct/2003:16:47:49 -0600] "GET /redes_neuronales/ HTTP/1.1" 200 2294
"http://www.aulafacil.net/Enlaces/Enlaces.asp?tipo=MANUALES" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
```

En las que se puede observar que el cliente seisamex-servnet.serv.net.mx el día 28 de Octubre del 2003 a las 16:47 horas, solicitó el directorio del manual de redes_neuronales siendo exitosa la petición que provino del URL www.aulafacil.net , además de exponer que el navegador que utilizó el visitante es Explorer 6.0 corriendo en plataforma MS Windows. Éste es un claro ejemplo de cómo son cada una de las líneas de la bitácora, mediante un estudio estadístico podrá determinarse si conviene desarrollar el sitio enfocado a una plataforma en particular, así como ver las horas pico de visitas, entre otras cosas.

Para este estudio estadístico existen herramientas libres que hacen el trabajo por el Administrador, una de ellas es la denominada *Webalizer*, que entregará resultados gráficos facilitando así su lectura. Gracias al monitoreo de la bitácora, es más fácil detectar un intento de ataque y en términos del servidor Web Apache es posible observar intentos de ataque del virus CodeRed por ejemplo, si en los logs aparece algo similar a:

```
132.248.34.235 - - [12/Mar/2003:11:33:59 -0600] "GET
```

```
/default.ida?XXXXXXXXXXXXXXXXXX0078%u0000%u00=a HTTP/1.0
```

Si se utiliza Linux, como en el esquema del ejemplo, se está inmune, sin embargo, sirve para detectar éste tipo de situaciones. Como se puede observar, el provecho que se puede sacar al estudio de las bitácoras es enorme, y dado que existen infinidad de bitácoras (para correo, para web, para transferencias, etcétera), existen herramientas disponibles que nos facilitan su análisis.
Para saber más:
http://www.mrunix.net/webalizer
Generador de estadísticas.
http://www.logwatch.org
Analizador de Logs.

A continuación se describen algunos ejemplos de bitácoras que se encuentran en el mercado:

Webalizer

Está escrito en lenguaje C para su portabilidad y rapidez y éstas son sus características especiales:

- ✓ Página Principal: En la página principal de las estadísticas se observa un reporte resumido de los últimos doce (12) meses. Este resumen contiene una fila por cada uno de los meses, las columnas están agrupadas en dos secciones: una contiene los promedios diarios de accesos, archivos, páginas y visitas. La otra sección contiene promedios mensuales de clientes, KBytes transferidos, visitas, páginas, archivos y accesos. Cada mes contiene un enlace a las estadísticas detalladas de ese período.
- ✓ Página de detalles: La primera información que se observa describe el período que comprende este reporte y la fecha en la que fue generado.
- ✓ Estadísticas mensuales: Es una tabla que contiene un total a la fecha de generación del documento de Accesos, Archivos, Páginas, Visitas, KBytes, Clientes y URLs.
- ✓ También contiene el promedio y el máximo de Accesos agrupados por hora y por día, archivos, páginas, visitas y KBytes agrupados por Día.
- ✓ Gráfica de uso diario: Esta gráfica contiene 3 histogramas, con los días del mes en la línea horizontal:
  - ➢ El primer histograma agrupa la cuenta de las páginas, los archivos y los accesos.
  - ➢ El segundo histograma agrupa la cuenta de los clientes y las visitas.
  - ➢ El tercer histograma despliega la cuenta del tráfico del sitio en KBytes.
- ✓ Estadísticas diarias del mes: Es una tabla con una fila por cada uno de los días del mes en los que se han llevado estadísticas y una columna por los accesos, archivos, páginas visitas, clientes y KBytes. Cada uno de estos tiene el número absoluto y el porcentaje relativo total de los días de la estadística.
- ✓ Gráfica de uso por horas: Es un histograma con las horas del día en la línea horizontal y en la vertical un promedio mensual por hora de páginas, archivos y accesos.
- ✓ Estadísticas por horas: Es una tabla que contiene una fila por cada hora y una columna por la cuenta de accesos, archivos, páginas y KBytes. Cada uno de estos tiene un promedio por hora, un número absoluto y su equivalente en promedio respecto al día completo.
- ✓ Primeros URL: Contiene una fila de los 30 primeros URL visitados ordenados por accesos, la siguiente tabla contiene los 10 primeros URL ordenados por tráfico en KBytes.
- ✓ Entrada y salida del sitio: Estas tablas contienen los 10 URL más usados como páginas de entrada y de salida, respectivamente.

- ✓ Clientes: Contiene información sobre los accesos, archivos, KBytes y visitas producidas por los clientes. Esta tabla contiene únicamente los primeros 30 clientes ordenados por accesos.

BlogPatrol

- ✓ Extensa estadística del sitio actualizada en tiempo real.
- ✓ Hace un seguimiento de las peticiones totales de su sitio, a los visitantes únicos por día, y a visitantes únicos diarios proyectados basados en actividad 60 minutos anteriores.
- ✓ Diario gráfico de actividad de los últimos 10 días.
- ✓ Informe total de visitantes únicos provenientes de 20 referencias. Esto le dice exactamente de donde surge el tráfico (incluyendo qué motores de búsqueda).
- ✓ 20 palabras claves del motor de búsqueda usadas para encontrar su sitio.
- ✓ Últimas 10 palabras clave usadas para encontrar su sitio.
- ✓ Muestra los browsers de la web, los sistemas operativos, y la resolución de la pantalla usada por sus visitantes.
- ✓ También proporciona el análisis detallado de los 25 visitantes más recientes.

iftop

Httpd iftop supervisa tráfico de la red en un interfaz de la red. Una herramienta para los administradores que tienen que mantener y localizar averías los servidores. Básicamente, ***iftop*** sigue los paquetes que pasan a través de un interfaz dado también como donde se dirigen esos paquetes.

Bitacoras de Apache – logging

Apache cuenta con dos archivos donde residen las bitacoras:

access.log → Registra todos los accesos al sitio.

error.log → Registra los errores que genere un acceso al sitio o que los procesos de Apache reporten.

ErrorLog

ErrorLog filename → Esta directiva define el nombre del archivo en el cual el servidor registrará los errores que se presenten.

Si filename no comienza con un "/" se considera que la ruta depende del ServerRoot.

Loglevel

Loglevel error → Esta directiva define el nivel de mensajes de error que serán registrados en la bitacora de "ErrorLog":

✓ emerg	warn
✓ alert	notice
✓ crit	info
✓ error	debug

CustomLog

CustomLog filename → Esta directiva define el nombre del archivo en el cual el servidor registrará los accesos que se presenten.

Si filename no comienza con un "/" se considera que la ruta depende del ServerRoot.

LogFormat format string nickname

LogFormat permite definir el formato que se utilizará para almacenar los registros. A cada formato se le puede asignar un nombre, utilizándolo luego para crear distintos tipos de ficheros de registro. Pueden existir varios logFormat distintos.

Por *default* viene configurado con la siguiente sintaxis:

"%h %l %u %t \"%r\" %>s %b"

Donde:

- ✓ %h = Registra la IP del cliente, hostname.
- ✓ %l = Si el deamon identd corre en el cliente, reporta la información que el identd devuelva.
- ✓ %u = Si se requiere de un username y password para acceder, en este campo se registra.
- ✓ %t = Fecha y hora del request en el formato [dia/mes/aňo:hora:minuto:segundo tzofffset].
- ✓ \"%r\" = Recurso solicitado por el cliente, entre comillas, request.
- ✓ %>s = Un código de tres digitos donde se muestra el valor devuelto al cliente, status.
- ✓ %b = El número de bytes devueltos exceptuando encabezados.

Exiten una serie de LogFormat preestablecidos:

Common → Detallado anteriormente (Default: "%h %l %u %t \"%r\" %>s %b").
Refered → Orientado a llevar un registro de las rutas de los usuarios dentro del sitio.
Agent → Orientado a registrar el nombre de los navegadores de los clientes.
Combined → Combina las caracteristicas de los anteriores.

HostNameLookup

Esta directiva habilita la resolución de nombres en el DNS cuando se establece una conexión.

Sintásis:

HostNameLookup on|off
Default: off

Se pueda habilitar esta Directiva en el archivo *httpd.conf*:

```
HostnameLookups On
```

, y desde línea de comandos ejecutar:

```
tail -f access_log
```

Nota: Por razones de rendimiento se sugiere no utilizarla. Para la resolución de nombres en las bitácoras existe el programa ***logresolve***, también de apache

Logresolve

Resuelve Hostnames para Direcciones IP en los archivos de 'logueo' (registro de acceso) de Apache.

Sintáxis:

logresolve [-s filename] [-c] < access.log > access.log.new

Descripción:

Logresolve es un ***programa de post-proceso para resolver direcciones IP en el servidor de Apache***. Para minimizar el impacto en el Servidor de Nombres, *logresolve* tiene su muy particular o propio caché de tabla de arreglos. Esto significa que cada número de IP sólo será visto la primera vez que se encuentre en el archivo de logueo.

Opciones:

-s filename → Especifica el nombre del archivo para grabar estadísticas.

-c → Ocasiona que el logresolve se aplique sobre algunos DNS's; encontrando el hostname de la dirección IP, esto hace que se vean las direcciones IP por medio del hostname y checa que se parezca a alguna de las direcciones originales.

Entonces, lo que hace ***Logresolve***, después de esta breve descripción, es que ***está dedicado a estar resolviendo Servidores de Nombres para diferentes Direcciones IP***.

Logrotate

Logrotate, como su nombre lo indica, es un ***sistema de rotación de logs*** (ficheros de información de estado que generan algunos programas, sobretodo en entornos profesionales, como Linux).

La utilidad *logrotate* está pensada para simplificar la administración de los ficheros de log (también llamados ficheros bitácora) en aquellos sistemas en los que se generan en grandes cantidades, como por ejemplo servidores de Internet. *logrotate* efectúa una rotación automática con posible compresión de los datos y otras opciones, como su borrado programado o envío por correo electrónico al administrador.

logrotate se puede programar para que rote los ficheros una vez al día, a la semana o mensualmente, o incluso en función de un parámetro de tamaño límite. Normalmente, logrotate se ejecuta una vez al día, según lo establece el fichero cron.daily. Si se necesita trabajar con gran cantidad de ficheros de log y de gran tamaño se puede instalar esta utilidad. Se ahorrará espacio en disco y se ganará claridad en la administración de su sistema.

Si se piensa que puede tener algún problema para trabajar con los *logs* antiguos comprimidos, piense que ya existen programas, como *webalizer* que extraen estadísticas de estos, aunque estén comprimidos

<Directory> y <DirectoryMatch>

<Directory dir>
...
</Directory>

La directiva Directory permite aplicar otras directivas de forma específica a todos los recursos dentro de la ruta definida por "dir".

Nota: Se deben considerar SIEMPRE rutas absolutas.

Directiva options

La directiva options controla que caracteristicas del servidor estan disponibles para un directorio en particular. Se utiliza para habilitar o deshabilitar cierta funcionalidad, la activación es de modo jerárquico.

Options [+|-] *option* [[+|-]*option*]

Puede ser colocado a *option none* en caso de que no se desee ninguna funcionalidad adicional.

All → Todas las opciones se activan excepto MultiViews, (Esta es la opción por default).
ExecCGI → La ejecución de ***scripts CGIs*** es permitida.
FollowSymLinks → Las ligas simbólicas son válidas dentro de este directorio.

> **Nota**: Aún cuando el servidor seguira las ligas simbólicas éste no cambiara de la ruta definida por la directiva <Directory>.

Includes → Permite el uso de ***Server-side includes***.
IncludesNOEXEC → Server-side includes son permitidos, pero ***#execCGI*** son desactivados.
Indexes → Si un URL mapea a un directorio solicitado y no hay DirectoryIndex (index.html) en este directorio, entonces el servidor devolverá un listado de directorio.

```
<Directory /web/docs>
    Options Indexes FollowSymLinks
</Directory>
<Directory /web/docs/spec>
    Options Includes
</Directory>
   <Directory /web/docs/spec>
    Options +Includes –Indexes
</Directory>
```

3.5 SITIOS WEB DINÁMICOS

Un Servidor Web, brinda, como se ha mencionado muchas aplicaciones e información para que múltiples usuarios revisen y/o manejen información, si éste la proporciona. Uno de los muchos beneficios que brinda un Servidor, es la de generar Sitios Web Dinámicos, algunos de ellos son:

- ✓ PHP
- ✓ Servlets
- ✓ JSP
- ✓ ASP

Para Sitios Web Dinámicos puedan brindar sus servicios de forma adecuada, deberán contar, al menos, cuentan con lo que se conoce como ***CGI.***

CGIs

CGI "Common Gateway Interface" → Define un modelo de programación que puede ser implementado con múltiples lenguajes que ofrece dinamismo en la creación de sitios Web

Los CGIs son programas que corren en el servidor, reciben parámetros desde el cliente y su salida es enviada al navegador. Permiten generar páginas dinámicas y fueron la primera alternativa para generar dinamismo a un sitio web.

ScriptAlias

Convierte las solicitudes vìa URL al path absoluto donde residen los cgis.

ScriptAlias /cgi-bin /usr/www/cgi-bin
Se puede usar en: Server config, virtual host

```
# ScriptAlias: This controls which directories contain server scripts.
...
ScriptAlias  /cgi-bin/  "/usr/apache2/cgi-bin/ "
```

AddHandler

AddHandler permite que determinada extensión sea relacionada a un evento en particular, en el caso de los CGIs lo que se indica es que la extensión ***.cgi*** queda identificada como un Script.

AddHandler cgi-scripts .cgi
Default : No activado
Se puede usar en: Server config, virtual host

Server-Side Includes

La opción SSI activa un filtro que permiten incluir etiquetas dentro de un archivo HTML, las cuales son procesadas por Apache, antes de ser enviadas como HTML al cliente.

Es necesario habilitar la opción **+Includes** en el directorio donde se encuentran los archivos que incluyen etiquetas SSI.

Directivas requerida para SSI

AddType text/html .shtml → Relaciona los archivos con extensión ***shtml*** como aquellos que contienen etiquetas SSI.

AddOutputFilter INCLUDES .shtml → Habilita el filtro para los archivos .shtml.

Nota: SSI representa un riesgo y debe ser usado con cautela, para evitar un incidente de seguridad.

<!--#elemento atributo="valor" -->

Los elementos disponibles como tags (etiquetas) SSI son:

- config
  - ***errmsg***
  - ***sizefmt***
  - ***timefmt***
- exec
  - ***cgi***
  - ***cmd***
- fsize
  - file
  - virtual
- flastmode
- Include
  - file
  - virtual

3.6 SEVIDORES VIRTUALES

Virtual Host

Los servidores virtuales permiten que un mismo servidor Apache pueda responder a diferentes solicitudes, con lo cual es posible mantener múltiples sitios web con diferentes nombres y/o direcciones IPs. Para poder configurarlos, dentro del archivo *httpd.*copnf, hay que situarse en la seccion #3, en la parte final del archivo, en donde se hace referencia a ellos.

Sintáxis:

<VirtualHost servername>
...
</VirtualHost>

Cabe recordar que, previamente los puertos se habilitaron, al principio del ***httpd.conf***, con la opción ***Listen***.

3.7 CONTROL DE ACCESO

Apache ofrece la funcionalidad al estilo "ACL" (Access Control List), con lo cual es factible determinar las direcciones IPs y los usuarios que tendrán acceso sobre recursos específicos del servidor Web.

La configuración de acceso vía nombre de dominio o vía IP puede realizarse con ayuda de las siguientes directivas:

- ***Order*** → Define el orden en que allow o deny serán implementadas
- ***Allow*** → Define los hosts que tendrán acceso al recurso.
- ***Deny*** → Define los hosts que no tendrán acceso al recurso.

- **Allow Deny**

Permite definir un conjunto de clientes a los que se les pueden permitir o negar el acceso al servicio de Web.

Allow from all|host|env=env-variable [host|env=env-variable]
Deny from all|host|env=env-variable [host|env=env-variable]

La definición de los nombres se puede realizar mediante alguna de las siguientes formas:

El nombre parcial de un dominio parcial. Ejemplo:
.mascarones.unam.mx
Una dirección IP. Ejemplo: 132.248.75.249

La pareja Red y Máscara: 192.168.0.0/255.255.255.0
Un dirección de red definida por Classless Inter-Domain Routing (CIDR):
10.2.2.110/24

- **order allow, deny**

Permite explícitamente a los clientes definidos en ***allow***, niega a todos los demás. Los clientes que se encuentren en ambas directivas (allow y deny) serán denegados. **"*Todo lo que no está explícitamente permitido está prohibido"***

- **order deny,allow**

Niega explícitamente a los clientes definidos en ***deny***, permite a todos los demás. Los clientes que se encuentren en ambas directivas (allow y deny) serán permitidos. **"*Todo lo que no está explícitamente prohibido está permitido"***

Al ofrecer Apache la funcionalidad al estilo "ACL", también es factible determinar asignar un ***login*** y ***password*** por usuario quienes tendrán acceso sobre recursos especificos del servidor Web.

AuthUserFile

Indica cual es el archivo que contiene la lista de ***usuarios*** y ***passwords*** para realizar la autenticación.

AuthUserFile /usr/local/apache/conf/.htpasswd

Nota: Es importante que este archivo no sea accesible por los clientes web, pues tendrían acceso a la lista de passwords y podrían atacarla por métodos de diccionario o incluso fuerza bruta para conseguir nuevas claves del sistema.

AuthGroupFile

Define cual es el archivo que contiene la lista de grupos y los usuarios que la conforman para realizar la autenticación.

AuthGroupFile /dev/null

AuthName

Define el nombre de autorización para el recurso protegido, esté aparecerá como un mensaje en la caja de diálogo que solicita el password para accesar al recurso protegido.

AuthName "Nombre del recurso protegido"

require valid-user

Esta directiva indica que se requiere de un usuario y una clave de acceso para acceso para un recurso determinado.

3.8 Módulos

En Apache los módulos son los encargados de agregar funcionalidad adicional incrementando el número de directivas disponibles.

Módulos estáticos → Se agregan al momento de compilar apache.

Modulos dinámicos → Se agregan durante el ambiente productivo, permiten adicionar funcionalidad sin necesidad de recompilar apache.

Hasta este punto, ya es possible instalar un servidor Web, que en este caso es Apache, y dentro de éste, trabajar con el manejo de directivas para brindar una mayor funcionalidad con ***Global Environment***, administrar puertos de conexión para los usuarios, grupos de operación para el servico de Apache.

También se pueden ya administrar las directivas para saber si se trabajará como servidor principal o estándar, y el soporte de multiples IPs o nombres de dominio, así como sus respectivas limitantes para determinar las direcciones IP y los usuarios que tendràn acceso sobre los recursos específicos de servidor Web.

CAPITULO IV

INTRODUCCIÓN A LA SEGURIDAD EN CÓMPUTO

4.1 Introducción a la Seguridad en Cómputo

Antecedentes de Seguridad

Las redes fueron diseñadas para el intercambio de información y compartir recursos. La seguridad no era un factor tomado en cuenta en el diseño de las redes. De esto se hace una remembranza de lo que ha sido el desarrollo de la seguridad informática:

- ✓ Computación electrónica 50 años.
- ✓ Redes sólo tienen 30 años de vida.
- ✓ Seguridad 23 años.
- ✓ Internet 15 años.
- ✓ Web 8 años.
- ✓ Intranets 5 años.
- ✓ VPNs 4 años

Desafortunadamente, surge un problema con este gran crecimiento, que incluirá (o incluye) individuos deshonestos que se introducen a los sistemas informáticos. Ninguna institución está libre del acecho de estos individuos. Todo el mundo alguna vez ha sido afectado por algún problema de seguridad.

Para empezar, es necesario ir definiendo algunos conceptos en cuanto a Seguridad en Informática se Refiere:

- ✓ **Amenaza** → Es ***Circunstancia*** o evento que puede causar daño violando la confidencialidad, integridad o disponibilidad. El término se refiere a un evento (por ejemplo un tornado, robo, infección por virus de cómputo, etc.) Frecuentemente se aprovecha una vulnerabilidad
- ✓ **Vulnerabilidad** → Es la ***Ausencia*** de una contramedida, o debilidad de la misma, de un sistema informático que permite que sus propiedades de sistema seguro sean violadas. Es una ***Condición*** que tiene potencial para permitir que ocurra una amenaza, con mayor frecuencia e impacto. La debilidad puede originarse en el diseño, la implementación o en los procedimientos para operar y administrar el sistema. En el argot de la seguridad computacional una vulnerabilidad también es conocida como un *hoyo*.

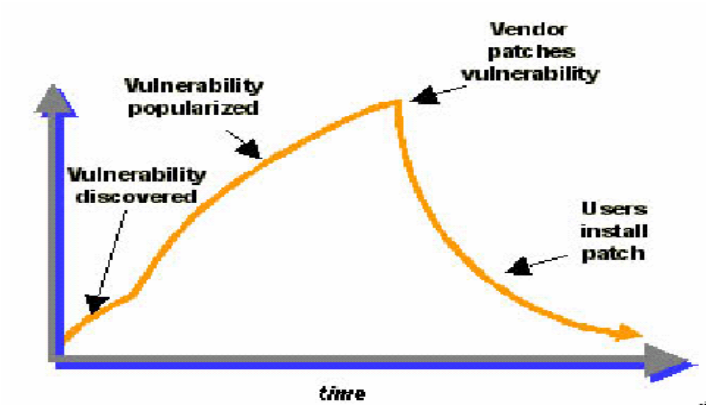


Figura 4.1.- Tiempo de vida de una vulnerabilidad

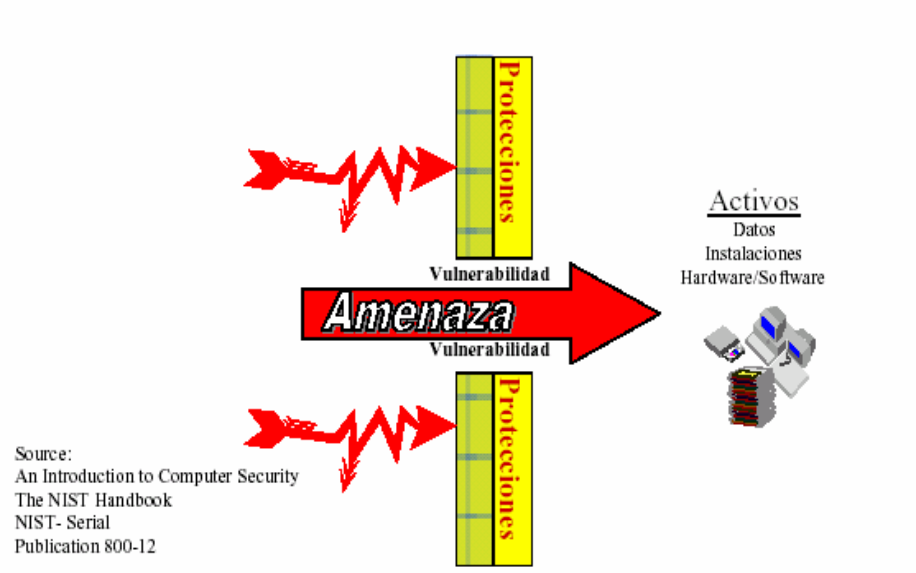


Figura 4.2 - Amenaza, Vulnerabilidad, Activos

Riesgo → Es el ***Potencial*** para perdida o falla, como respuesta a las siguientes preguntas ¿Qué podría pasar (o cual es la amenaza)?, ¿Qué tan malo puede ser (impacto o consecuencia)?, ¿Qué tan frecuente puede ocurrir (frecuencia)?, ¿Qué tanta certidumbre se tiene en las primeras 3 respuestas (grado de confianza)?, etc. Nótese que si no hay incertidumbre, no hay un riesgo que perseguir.

Exploit → Se refiere a la forma de explotar una Vulnerabilidad, término muy enfocado a herramientas de ataque, sobre equipos de cómputo). Aprovechamiento automático de una vulnerabilidad, generalmente en forma de un programa/software que realiza de forma automática un ataque aprovechándose de una vulnerabilidad

Ataques → Acción o acciones que tienen por objetivo el que cualquier parte de un sistema de información automatizado deje de funcionar de acuerdo con su propósito definido. Esto incluye cualquier acción que causa la destrucción, modificación o retraso del servicio no autorizado. Este tipo de ataque no es de tipo físico (aunque puede ser). Un ataque no se realiza en un sólo paso, depende de los objetivos del atacante y Puede consistir de varios pasos antes de llegar a su objetivo.

Se dice entonces que, se entiende por amenaza una condición del entorno del sistema de información (persona, máquina, suceso o idea) que, dada una oportunidad, podría dar lugar a que se produjese una violación de la seguridad (confidencialidad, integridad, disponibilidad o uso legítimo). [1]

Clasificación y tipos de ataques contra Sistemas de Información[2]

La política de seguridad y el análisis de riesgos habrán identificado las amenazas que han de ser contrarrestadas, dependiendo del diseñador del sistema de seguridad especificar los

[1] http://www.canal-ayuda.org/Seguridad/tipos_ataques.htm
[2] Idem

servicios y mecanismos de seguridad necesarios. Las amenazas a la seguridad en una red pueden caracterizarse modelando el sistema como un flujo de información desde una fuente, como por ejemplo un fichero o una región de la memoria principal, a un destino, como por ejemplo otro fichero o un usuario.

Un ataque no es más que la realización de una amenaza. Las cuatro categorías generales de amenazas o ataques son las siguientes:

- ✓ **Interrupción**: un recurso del sistema es destruido o se vuelve no disponible. Este es un ataque contra la disponibilidad.
- ✓ **Intercepción**: una entidad no autorizada consigue acceso a un recurso. Este es un ataque contra la confidencialidad. La entidad no autorizada podría ser una persona, un programa o un ordenador.
- ✓ **Modificación**: una entidad no autorizada no sólo consigue acceder a un recurso, sino que es capaz de manipularlo. Este es un ataque contra la integridad.
- ✓ **Fabricación**: una entidad no autorizada inserta objetos falsificados en el sistema. Este es un ataque contra la autenticidad.

Tipos de Ataques

Los ataque informáticos son de dos tipos, Ataque Pasivos y Activos.

En los ***Ataques Pasivos*** el atacante no altera la comunicación, sino que únicamente la escucha o monitoriza, para obtener información que está siendo transmitida. Sus objetivos son la intercepción de datos y el análisis de tráfico, una técnica más sutil para obtener información de la comunicación, que puede consistir en:

- ✓ Obtención del origen y destinatario de la comunicación, leyendo las cabeceras de los paquetes monitorizados.
- ✓ Control del volumen de tráfico intercambiado entre las entidades monitorizadas, obteniendo así información acerca de actividad o inactividad inusuales.
- ✓ Control de las horas habituales de intercambio de datos entre las entidades de la comunicación, para extraer información acerca de los períodos de actividad.

Los ataques pasivos son muy difíciles de detectar, ya que no provocan ninguna alteración de los datos. Sin embargo, es posible evitar su éxito mediante el cifrado de la información y otros mecanismos que se verán más adelante.

Los ***Ataques Pasivos*** sirven para:

- ✓ Recopilar información en Bases de Datos.
- ✓ Whois (Saber la identidad)
  - ➢ ARIN, LACNIC
- ✓ DNS - NIC
  - ➢ Listados de Ips y Nombres de Dominio, Trasferencias de Zonas
- ✓ Trazado de Ruta
  - ➢ Traceroute. www.visualroute.com
- ✓ Sniffers
  - ➢ Snort, dsniff
- ✓ Analizadores de Trafico
  - ➢ Ethereal

Los ***Ataques Activos*** implican algún tipo de modificación del flujo de datos transmitido o la creación de un falso flujo de datos, pudiendo subdividirse en cuatro categorías:

- ✓ **Suplantación de identidad:** Donde, el intruso se hace pasar por una entidad diferente. ormalmente incluye alguna de las otras formas de ataque activo. Por ejemplo, secuencias de autenticación pueden ser capturadas y repetidas, permitiendo a una entidad no autorizada accesar a una serie de recursos privilegiados suplantando a la entidad que posee esos privilegios, como al robar la contraseña de acceso a una cuenta.
- ✓ **Reactuación:** Uno o varios mensajes legítimos son capturados y repetidos para producir un efecto no deseado, como por ejemplo ingresar dinero repetidas veces en una cuenta dada.
- ✓ **Modificación de mensajes:** Una porción del mensaje legítimo es alterada, o los mensajes son retardados o reordenados, para producir un efecto no autorizado. Por ejemplo, el mensaje "Ingresa un millón de pesos en la cuenta A" podría ser modificado para decir "Ingresa un millón de pesos en la cuenta B".
- ✓ **Degradación fraudulenta del servicio:** Impide o inhibe el uso normal o la gestión de recursos informáticos y de comunicaciones. Por ejemplo, el intruso podría suprimir todos los mensajes dirigidos a una determinada entidad o se podría interrumpir el servicio de una red inundándola con mensajes espurios. Entre estos ataques se encuentran los de denegación de servicio, consistentes en paralizar temporalmente el servicio de un servidor de correo, Web, FTP, etc.

Principales Ataques

Ingeniería Social

• Es una de las formas más comunes para penetrar sistemas de “alta seguridad”. Uso de trucos psicológicos, por parte de un atacante externo, sobre usuarios legítimos de un sistema para obtener información (usenames y passwords) necesaria para acceder a un sistema. Se basa en ataques como: usurpación de identidad, hurgar en la basura, inocencia de la gente, relaciones humanas, etc.

Virus

Un virus se define como una porción de código de programación cuyo objetivo es implementarse a si mismo en un archivo ejecutable y multiplicarse sistemáticamente de un archivo a otro. Además de esta función primaria de "invasión" o "reproducción", los virus están diseñados para realizar una acción concreta en los sistemas informáticos sin la autorización del usuario.

Variantes relacionadas con virus

En ocasiones de habla de estas variantes como si de virus se tratara, cuando en realidad son conceptualmente diferentes. Algunos antivirus pueden detectarlos. Estas variantes son:

- ✓ Troyanos
- ✓ Gusanos
- ✓ Bomba lógica
- ✓ Spyware
- ✓ Adware

Los gusanos

Un *gusano* (informático) es un programa que produce copias de sí mismo de un sistema a otro a través de la red. En las máquinas que se instala, produce enormes sobrecargas de procesamiento que reducen la disponibilidad de los sistemas afectados.

Diferencias que hay que tomar en cuenta para protegerse mejor:

- ✓ Virus

- El programa por sí sólo se ejecuta, Vive dentro de otro programa y Escala en memoria

✓ Backdoor
- Cuando el equipo ha sufrido un incidente, alguien "entró" al sistema, es común que se instale para seguir con el control sobre el sistema.

✓ Caballo de troya
- Se le envía al usuario para que lo ejecute aparentando que se trata de un programa con una funcionalidad diferente

Bombas lógicas

Una bomba lógica es una modificación en un programa que lo obliga a ejecutarse de manera diferente bajo ciertas circunstancias. Bajo condiciones normales, el programa se comporta como previsto y, la bomba no puede ser detectada. Ejemplo de pseudocódigo.

```
If eth0=down THEN bomba= "rm –rf /"
                ELSE bomba=./running
```

DOS (Denial of Service) [3] [4] [5]

Las ***Negaciones de Servicio*** (conocidas como ***DoS, Denial of Service***) son ataques dirigidos contra un recurso informático (generalmente una máquina o una red, pero también podría tratarse de una simple impresora o una terminal) con el objetivo de degradar total o parcialmente los servicios prestados por ese recurso a sus usuarios legítimos; constituyen en muchos casos uno de los ataques más sencillos y contundentes contra todo tipo de servicios, y en entornos donde la disponibilidad es valorada por encima de otros parámetros de la seguridad global puede convertirse en un serio problema, ya que un pirata puede interrumpir constantemente un servicio sin necesidad de grandes conocimientos o recursos, utilizando simplemente sencillos programas y un módem y un PC caseros. Las negaciones de servicio más habituales suelen consistir en la inhabilitación total de un determinado servicio o de un sistema completo, bien porque ha sido realmente bloqueado por el atacante o bien porque está tan degradado que es incapaz de ofrecer un servicio a sus usuarios. En la mayor parte de sistemas, un usuario con acceso *shell* no tendría muchas dificultades en causar una negación de servicio que tirara un servicio de la máquina o la ralentizara enormemente; esto no tiene por qué ser - y de hecho en muchos casos no lo es - un ataque intencionado, sino que puede deberse a un simple error de programación. Por ejemplo, piénsese en el siguiente *shellscript* (funciona en Linux):

```
root:~# cat /usr/local/bin/lanzador

#!/bin/sh
ps -ef|grep calcula|grep -v grep 2>&1 >/dev/null
if [ $? -eq 1 ]; then
        /usr/local/bin/calcula &
fi
luisa:~#
```

[3] http://www.dcc.uchile.cl
[4] http://his.sourceforge.net/proy_his/glosario/
[5] http://es.tldp.org/Manuales-LuCAS/doc-unixsec/unixsec-html/node275.html

Como se puede ver, este *script* comprueba si un determinado programa está lanzado en la máquina, y si no lo está lo lanza él; algo completamente inofensivo a primera vista, y planificado habitualmente en muchos sistemas para que se ejecute - por ejemplo, cada cinco minutos - desde ***crond***. Sin embargo, uno se puede parar a pensar qué sucedería bajo algunas circunstancias anormales: ¿y si en el arranque de la máquina, por el motivo que sea, no se ha montado correctamente el directorio /proc/? Si esto sucede, la orden `ps' generará un error, la condición se cumplirá siempre, y cada cinco minutos se lanzará una copia de calcula; si este programa consume mucha CPU, al poco tiempo se tendrá un elevado número de copias que cargarán enormemente el sistema hasta hacerlo inutilizable. Un ejemplo perfecto de negación de servicio.

Todo ataque que busca aparentar una alta demanda sobre un servicio válido con el objeto de saturar la capacidad de respuesta y generar una caída del sistema victima.

DDOS (Distributed Denial of Service)

De un tiempo al presente - en concreto, desde 1999 - se ha popularizado mucho el término ***`Negación de Servicio Distribuida' (Distributed Denial of Service, DDoS)***: en este ataque un pirata compromete en primer lugar un determinado número de máquinas y, en un determinado momento, hace que todas ellas ataquen masiva y simultáneamente al objetivo u objetivos reales enviándoles diferentes tipos de paquetes; por muy grandes que sean los recursos de la víctima, el gran número de tramas que reciben hará que tarde o temprano dichos recursos sean incapaces de ofrecer un servicio, con lo que el ataque habrá sido exitoso. Si en lugar de cientos o miles de equipos atacando a la vez lo hiciera uno sólo las posibilidades de éxito serían casi inexistentes, pero es justamente el elevado número de `pequeños' atacantes lo que hace muy difícil evitar este tipo de negaciones de servicio.

A pesar de las dificultades con las que se pueden encontrar a la hora de prevenir ataques de negación de servicio, una serie de medidas sencillas pueden ayudar de forma relativa en esa tarea; las negaciones de servicio son por desgracia cada día más frecuentes, y ninguna organización está a salvo de las mismas. Especialmente en los ataques distribuidos, la seguridad de cualquier usuario conectado a Internet (aunque sea con un sencillo PC y un módem) es un eslabón importante en la seguridad global de la red, ya que esos usuarios se convierten muchas veces sin saberlo en satélites que colaboran en un ataque masivo contra organizaciones de cualquier tipo.

En febrero/marzo del 2000, varias empresas que apoyan su estrategia en Internet fueron atacadas. Yahoo! estima pérdidas por US$500,000 dls por dejar de dar servicio durante 3 horas. Entre ellas destacan:

- ✓ CNN (Agencia Noticiosa)
- ✓ Amazon (Venta de libros, discos, etc.)
- ✓ e-Bay (Venta de artículos en remate)
- ✓ e-Trade (compra y venta de acciones)
- ✓ Yahoo (Correo gratuito)

Algunos Ejemplos de ataques DoS pueden ser Ping de la muerte, Inundación Syn, Spoofing, Smurf, Fraggle o DoS Distribuido

SYN

El mnemotécnico del carácter 22 de ASCII que representa la inactividad sincrónica que se utiliza con frecuencia para controlar monitores de pantalla, impresoras y dispositivos módem. También, un lenguaje de especificación sintáctica para COPS.[6]

[6] http://www.maccare.com.ar

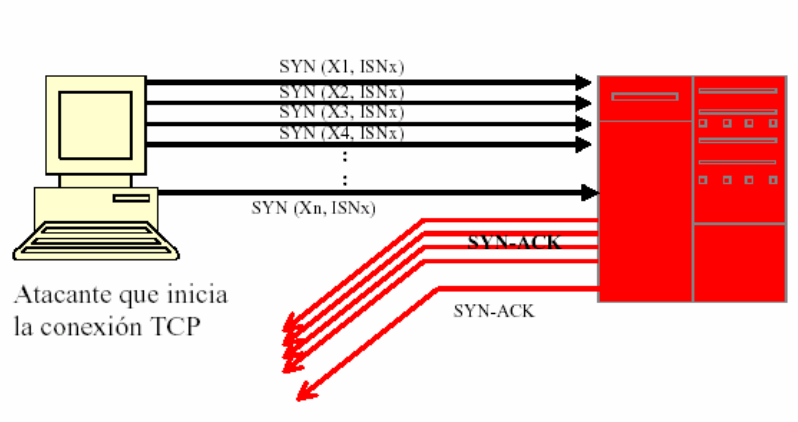


Figura 4.3 - Ejemplo SYN

Contramedidas inundación SYN

- ✓ Aumentar el tamaño de la cola de conexión
- ✓ Disminuir período tiempo establecimiento conexión
- ✓ Filtrado de paquetes
- ✓ direcciones IP no “alcanzables”
- ✓ Emplear parches software de los fabricantes de los SO

Smurfing

Smurf es uno de los ataques de DoS más temidos. Requiere 3 actores: La víctima, el atacante y la red amplificadora:

- ✓ El atacante originará un paquete ICMP hacia la dirección de broadcast de la red amplificadora, haciendo aparecer que su origen es una interfaz de la red de la víctima
- ✓ Cada interfaz de la red amplificadora enviará respuestas a la supuesta interfaz de origen

Previniendo smurf

Para no permitir ser utilizado como red amplificadora debe deshabilitar el paso de mensajes destinados a broadcast a través de los routers de frontera

- ✓ Cisco: no ip direct-broadcast

Para limitar el daño ocurrido por un ataque de este tipo sobre su red, limite el tráfico ICMP a un valor razonable de acuerdo a su disponibilidad de ancho de banda. Verifique si realmente necesita permitir tráfico de entrada ICMP a toda su red

Spoofing

Spoofing es la creación de paquetes de comunicación TCP/IP usando una dirección IP de alguien más. Lo anterior permite entrar en un sistema haciéndose pasar por un usuario autorizado. Una vez dentro del sistema, el atacante puede servirse de éste como plataforma para introducirse en otro y así sucesivamente.

Secuestro de sesiones (Termino en inglés: hijacking)

Tipo de ataque en que el atacante toma control de una comunicación tal y como un secuestrador de aviones tomo control del avión.

- ✓ Entre dos entidades y haciendo pasar por una de ellas

En un tipo de ataque (man in the middle)

- ✓ el atacante toma control de la conexión mientras esta se produce.

El objetivo es robar una conexión generada por un aplicación de red iniciada por un cliente (p.e. telnet)

Escaneo de Puertos

El escaneo de puertos es un método que consiste en conectarse a los puertos UDP y TCP del sistema objetivo, con el proposito de determinar cuales se encuentran escuchando.

Nmap

Nmap es un programa *open source* que sirve para efectuar port scanning (escaneo de puertos), distribuido por insecure.org. Esta orientado a la indentificación de puertos abiertos en una computadora objetivo, determinando que servicio(s) esta ejecutando la misma, e intenta determinar que sistema operativo utiliza dicha computadora, (esta técnica es también conocida como *fingerprinting*). Ha llegado a ser uno de las herramientas imprescindibles para todo administrador de sistemas, y es usado para pruebas de penetración y tareas de seguridad informática en general. Como muchas herramientas usadas en el campo de la seguridad informática, Nmap puede ser utilizado tanto por los administradores de sistema como por *Crackers* o *Script Kiddies*.

***Nota:** esta herramienta puede ser utilizada para realizar auditorias de seguridad en una red, pero también puede ser utilizada para fines delictivos, ya que esta herramienta pone al descubierto, puertos abiertos en las computadoras de una red, así como también es posible conocer como se encuentra organizada, y de cuantas computadoras consta una red.

Alguien dijo una vez, que la mejor forma de comprobar la seguridad de la red, es tratar de romper uno mismo dicha seguridad. Sólo de esta forma se podrá conocer y quitarse falsas creencias de que la red es lo suficiente segura. *Además, si se cree que la red es insegura, no se preocupe, que si existe una forma fácil y rápida de protegerse: por favor desconecte el cable de red.* Nmap puede bajarse del sitio oficial en de esta dirección: http://www.insecure.org/nmap/

Nmap realiza un escaneo para verificar que puertos se encuentran disponibles en el servidor objetivo.

```
nmap 132.248.75.135
```

Si se desea que nmap sólo realice un barrido sobre puertos TCP

```
nmap -sT 132.248.75.135
```

Para que nmap sólo realice un barrido sobre puertos UDP

```
nmap -sT 132.248.75.135
```

Para que nmap haga un barrido de IPs

```
nmap -sP 132.248.75.*
```

Si se desea que nmap sólo realice un barrido con conexiones semiabiertas.

```
nmap -sS 132.249.75.135
```

Estados TCP

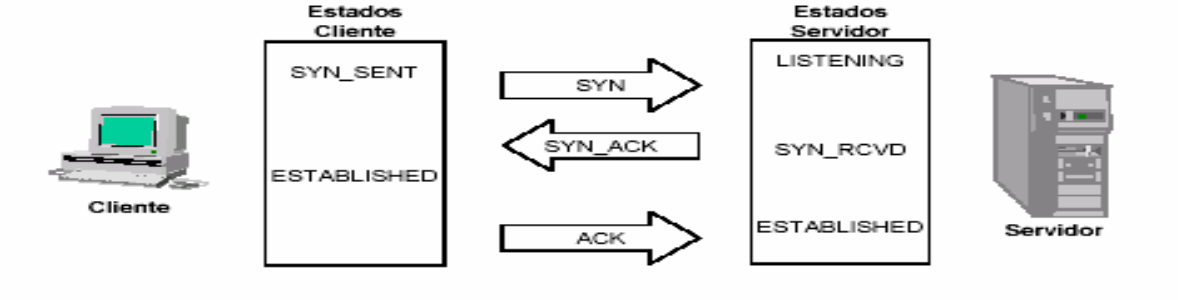


Figura 4.4 – Diagrama de los Estdos TCP

- Estado Cerrado – **CLOSED** : no-estado antes conexión empiece.
- En Modo Escucha – **LISTEN** : host espera petición conexión.
- En Sincronìa de Envío de Datos - **SYN-SENT**: host envió paquete syn y espera syn_ack.
- En sincronía de Recepción de Datos - **SYN-RCVD**: host recibió paquete syn y respondio con syn_ack.
- En modo Estable - **ESTABLISHED**: host iniciador recibió un syn-ack y el host receptor recibió paquete ack.

Sniffers

Un ***sniffer*** es un proceso que "olfatea" el tráfico que se genera en la red. Puede leer toda la información que circule sólo por el tramo de red en el que se ubique. Se pueden capturar claves de acceso, datos que se transmiten, números de secuencia, etc.

Un analizador de protocolos es un *sniffer* al que se le ha añadido funcionalidad suficiente como para entender y traducir los protocolos que se están hablando en la red. Debe tener suficiente funcionalidad como para entender las tramas de nivel de enlace, y los paquetes que transporten.

Normalmente la diferencia entre un sniffer y un analizador de protocolos, es que el segundo está más orientado al análisis del tráfico y no a la escucha de los datos transmitidos. Ejemplos de Sniffers pueden ser ***Dsniff***, ***Sniffit*** o ***snifsol***. Ejemplos de Analizadores de protocolos pueden ser ***Ethereal*** o ***Tcpdump***. Algunas herramientas más elaboradas pueden utilizarse como sniffers, como ***snort***.

Snort

Snort es un sniffer de paquetes y un detector de intrusos. Es un software muy flexible que ofrece capacidades de almacenamiento de sus bitácoras tanto en archivos de texto como en bases de datos abiertas como lo es MySQL. Así mismo existen herramientas de terceros para mostrar

reportes en tiempo real (ACID) o para convertirlo en un Sistema Detector y Preventor de Intrusos. Es una Herramienta que funciona de 3 modos, como sniffer, packet logger y como herramienta de detección de intrusos. Se pueden detectar ataques hacia un *host* determinado y se ve quien es por medio de los '*logs*'.

El paquete PCRE contiene librerías de expresiones regulares compatibles con Perl. Son útiles para implementar búsquedas de patrones de expresiones regulares usando las mismas sintáxis y semántica que Perl 5.

El funcionamiento de snort, en modo sniffer, es leer los paquetes que transitan por la Red e imprimirlos en la Consola. Por ejemplo:

- ✓ **snort -v** → Imprime en pantalla la IP y las cabeceras TCP/UDP/ICMP.
- ✓ **snort -dv** → Imprime, además, los datos que pasan por la interface de Red.
- ✓ **snort -dev** → Igual que en el anterior, pero ahora muestra la información de manera más detalla.

En modo **packet logger** → Igual que el anterior, con la excepción de que en vez de enviar la salida a la Consola, la envía a ficheros, al directorio que se le indique.

Tipos de Sniffers

- ✓ Pasivos → Sniffers que no realizan actividad alguna, sólo capturan paquetes
- ✓ Activos → Los sniffers intentan apoderarse de las sesiones, uso de técnicas para lograr lo anterior

- ✓ **Detección de sniffers** → Los Sniffers son difíciles de detectar y combatir ya que son programas pasivos, no generan bitácoras; cuando se usan propiamente, no usan mucho disco ni memoria; es posible localizarlos a nivel local, para ello es necesario verificar qué se está ejecutando; A nivel red, algunos pueden localizarse mediante herramientas como: antisnif.
- ✓ **Lectura a nivel enlace** → El sniffer se dedica a leer TRAMAS de red, los datos que se obtienen de él serán tramas que transportarán paquetes (IP, IPX, etc). En estos paquetes se incluyen los datos de capas superiores, entre ellos los de la capa de aplicación (posiblemente claves de acceso).
- ✓ **Medios no confiables** → Dado que los medios de transmisión no son confiables, es necesario establecer mecanismos para evitar que la información sea capturada durante su transmisión. Para lo cual se utiliza **SSH**, **HTTPS**, **POPS** o **IMAPS**.

4.2 Definición de Seguridad

Un sistema de cómputo es seguro si se puede confiar en que se comportará como se espera que lo haga, y que la información en él se mantendrá inalterada y accesible durante el tiempo que su dueño lo desee para los usuarios que él mismo decida.

Servicios de Seguridad

Un servicio de seguridad es una característica que debe tener un sistema para satisfacer una política de seguridad. El estándar ISO 7498-2, que cubre las comunicaciones seguras entre sistemas abiertos define cinco clases de servicios de seguridad:

- ✓ **Confidencialidad** → Sólo el propietario del secreto es capaz de descifrar la información.

- ✓ **Autenticación** → Se asegura de la identidad de la persona del otro lado de la línea, se puede realizar mediante alguno de los siguientes mecanismos: "algo que se sabe", "algo que se tiene" o "algo que se es".
- ✓ **Integridad** → Se asegura que la información no ha cambiado durante la transmisión.
- ✓ **Autorización (Control de Acceso)** → Se asegura que la información estará sólo para determinados usuarios con la posibilidad de manejar diferentes niveles para cada uno de ellos.
- ✓ **No Repudio** → Se asegura que el emisor de la información no puede negar haberla enviado. Existen otros estándares de seguridad como es el caso del British standard 7799 que actualmente es conocido como BS ISO/IEC 17799:2000, BS 7799-1:200. Aún cuando no forman parte de un estándar existen otros servicios que con frecuencia son considerados:
- ✓ **Disponibilidad** → Asegurar que el sistema este disponible cuando se le requiera.
- ✓ **Auditoria** → Asegurar que se defina un registro cronológico de los eventos exitosos y fallidos, que proporcionen evidencia de la actividad del sistema.

Niveles de Confianza I

El *"National Computer Security Center"* publicó en 1985 un libro conocido como el *"U.S. Deparment of Defense Trusted Computer System Evaluation Criteria"* mejor conocido como "***Orange Book***" el cual define cuatro divisiones y siete niveles de confianza para ambientes de cómputo. Está a nivel de Sistema Operativo.

La *"Common Criteria Organization"* ha proporcionado siete niveles de aseguramiento para un sistema, conocidos como *"Evaluation Assurance Levels"* EALs, con los cuales se clasifica a los ambientes de cómputo.

4.3 Criptología

La Criptografía del griego "Kryptós" oculto y "gráphein" escritura, "escritura oculta" se encarga de convertir un texto normal y comprensible en un formato incomprensible a menos que se posea un conocimiento secreto. Se encarga de convertir texto comprensible a formato incomprensible, a menos que se posea un conocimiento *Secreto*.

- ✓ En épocas recientes la criptología se define como la ciencia de usar las matemáticas para cifrar y descifrar información.
- ✓ El criptoanálisis es la ciencia de analizar y romper la comunicación segura mediante el análisis de un algoritmo empleado.
- ✓ La Criptología involucra tanto criptoanálisis como la criptografía.

Algoritmos Criptográficos

Los algoritmos criptográficos pueden estar basados en:

Criptografía Simétrica:

- ✓ Requiere que ***el emisor y el receptor compartan una clave secreta*** "Llave", la cual es utilizada para cifrar el mensaje cuando se envía y descifrar el mensaje al recibirlo.
- ✓ El gran inconveniente se presenta en el proceso de intercambiar de forma segura la Llave. En este esquema de seguridad es importante que el emisor disponga de un canal seguro para realizar la entrega de la llave al inicio de la transmisión.

Criptografía Asimétrica (también conocida como de llave pública):

- ✓ Resuelve el problema de intercambiar el secreto utilizando para ello un algoritmo basado en dos llaves, tanto el emisor como el receptor utilizan un par de llaves una "pública" y otra "privada". La privada es mantenida en secreto por un individuo, la pública esta disponible para que otro individuo sea capaz de cifrar la información sensible con ella.
- ✓ Cuando el emisor desea enviar un mensaje lo cifra con la llave pública del receptor, siendo este el único capaz de descifrarlo mediante el uso de su llave privada. En este caso conseguimos asegurar la confidencialidad del mensaje.
- ✓ Rivest Shamir Adleman (RSA) y Diffie-Hellman (D-H) son dos sistemas de llaves públicas utilizados actualmente.

  - ➢ **Diffie-Hellman** → El algoritmo de clave pública Diffie-Hellman fue desarrollado en 1976, por Whitfield Diffie y Martín Hellman. Esta basado en un algoritmo llamado de exponencial de claves.

  - ➢ **RSA** → El algoritmo de clave pública RSA, fue desarrollado por Rivest Shami Adelman en 1977, es el criptosistema de clave pública de uso más común en la actualidad, Se utiliza para cifrado y autorización, tiene extensiones de 768 , 1024 y mayores. La solidez de RSA proviene de la dificultad para factorizar números primos grandes, RSA se utiliza en aplicaciones tan populares como: PGP y S/MIME.

Message Digest

Conocidos también como Hashes, checksums son el resultado de algoritmos unidireccionales que permiten generar una "huella digital" de un mensaje. Así dan Integridad al Mensaje. Permiten obtener una cadena de longitud fija que es una representación condensada de un mensaje. Permite garantizar la integridad de un mensaje, si un sólo carácter del mensaje original cambia el Hash será totalmente distinto. Por ejemplo SHA-1, MD5. En el Anexo 2 se describen más detalle estos elementos.

Firma Digital[7]

Una firma digital es un conjunto de datos asociados a un mensaje que permite asegurar la identidad del firmante y la integridad del mensaje. La firma digital no implica que el mensaje esté encriptado, es decir, que este no pueda ser leído por otras personas; al igual que cuando se firma un documento holográficamente, éste sí puede ser visualizado por otras personas.

El procedimiento utilizado para firmar digitalmente un mensaje es el siguiente: el firmante genera mediante una función matemática una huella digital del mensaje. Esta huella digital se encripta con la clave privada del firmante, y el resultado es lo que se denomina firma digital la cual se enviará adjunta al mensaje original. De esta manera el firmante va a estar adjuntando al documento una marca que es única para ese documento y que sólo él es capaz de producir. El receptor del mensaje podrá comprobar que el mensaje no fue modificado desde su creación y que el firmante es quién dice serlo a través del siguiente procedimiento: en primer término generará la huella digital del mensaje recibido, luego desencriptará la firma digital del mensaje utilizando la clave pública del firmante y obtendrá de esa forma la huella digital del mensaje original; si ambas huellas digitales coinciden, significa que el mensaje no fue alterado y que el firmante es quien dice serlo.

¿Si el emisor cifra con su llave privada en lugar de usar la pública? Si el emisor cifra con su llave privada, todo aquel que disponga de la llave pública puede conocer el mensaje, por lo que el objetivo en este caso no es asegurar confidencialidad.

[7] http://ca.sgp.gov.ar/faq.html

De lo anterior, se obtiene:

- ✓ Autenticación dado que sólo el propietario de la llave privada pudo haber generado el mensaje.
- ✓ Integridad considerando que el mensaje requiere llegar sin cambios para que el algoritmo de descifrado pueda operar.
- ✓ No repudio considerando que la llave privada esta sólo en posesión del emisor, por lo que no puede negar su autoría.

Certificados Digitales

La norma X.509 es el estándar para formatos de certificados con llave pública. Un certificado X.509 consiste de la llave pública de un usuario y la firma de un tercero para la identificación en el bloque de identificación de ese usuario.

Autoridades Certificadoras:

El esquema de firmas digitales requiere de alguien que autentifique que un individuo es quien dice ser, instituciones gubernamentales o financieras, se encargan de emitir certificados digitales, en los cuales se integra la llave pública del individuo, piezas de información sobre el individuo e identificadores de la autoridad certificadora quien finalmente se encarga de firmar digitalmente con su llave privada de la CA.

Autoridades Certificadoras

El esquema de firmas digitales requiere de alguien que autentifique que un individuo es quien dice ser, instituciones gubernamentales o financieras, se encargan de emitir certificados digitales, en los cuales se integra la llave pública del individuo, piezas de información sobre el individuo e identificadores de la autoridad certificadora quien finalmente se encarga de firmar digitalmente con su llave privada de la CA (Certified Authority). La norma X.509 es el estándar para formatos de certificados con llave pública. Un certificado X.509 consiste de la llave pública de un usuario y la firma de un tercero para la identificación en el bloque de identificación de ese usuario. Si se desea elevar el nivel de confianza sobre un certificado se puede:

- ✓ Recurrir a una Certify Authority y solicitar que nos firme nuestro certificado.
  - ➢ www.verisign.com
  - ➢ www.geotrust.com

Se convierte en una CA y firma uno mismo los propios certificados, sin embargo en este caso:

- ✓ Se debe instalar en cada navegador el ca.crt
- ✓ Y sólo es factible para el interior de las organizaciones.
- ✓ Al inicio de la navegación en el sitio solicitar al visitante su confianza y solicitarle instale el certificado propio

GNU PG

Gnu Privacy Guard (GnuPG o GPG) es un sistema de codificación de código libre y desarrollo abierto. Es un reemplazo libre a PGP (*Pretty Good Privacy*) originalmente creado por Phil R Zimmermann.

Usando GPG

Para empezar a utilizar ***gpg***, es necesario generar un usuario, como se vio en el capítulo II, y cambiarse de sesión **su - usuario**

Ya estando dentro del home del usuario se ejecuta el comando (gpg)

Se genera un nuevo par de llaves (privada y pública) para el usuario. Ahora se inicia la creación de la llave

```
gpg  --gen -key
```

Y se despliega una serie de datos que conforman la generación de dicha llave:

1. Selection → 1
2. Keysize de 1024 a 4096 bits → 2048
3. Key is valid for (0) → 10 (días) Periodo de Válidez para que tendrá la llave generada
4. Correct → Yes Datos Correctos.
5. ID:
   - ✓ Real Name : Juventino Rosas
   - ✓ e-mail : juveRos@correo.servidor.com

Para mostrar las llaves con las que cuenta el "llavero"

```
gpg  --list-keys
```

Después, se exporta la llave pública del usuario a un archivo.

```
gpg --export  --armor > MiLlave.key
```

Ahora, se importa la llave pública de otro usuario

```
gpg --import < LlaveDeUnAmigo.key
```

Hay que cambiar el usuario propietario del archivo que se ha importado

```
Chown  usuario.1000  LlaveDeUnAmigo.key
```

Se firma la llave pública de otro usuario y se considera confiable.

```
gpg --sign-key  <id>
```

Donde <*id*> es el nombre que tiene el usuario del que se ha importado dicho archivo.

Ahora, se cifra un Mensaje.txt con la llave de un usuario.

```
gpg -e Mensaje.txt
```

Se Descifra el Mensaje.txt con la llave privada del usuario.

```
gpg -d Mensaje.txt.gpg
```

Secure Socket Layer

Es una propuesta de estándar para encriptado y autenticación en el Web.

◆Diseñado en 1993 por Netscape. Es un esquema de encriptado de bajo nivel usado para encriptar transacciones en protocolos de nivel aplicación como HTTP, FTP, etc. Con SSL puede autentificarse un servidor con respecto a su cliente y viceversa.

Se basa en un esquema de llave pública para el intercambio de llaves de sesión. Las llaves de sesión son usadas para cifrar las transacciones sobre HTTP. Cada transacción usa una llave de sesión. Esto dificulta al "atacante" el comprometer toda una sesión.

Objetivos de SSL

- ✓ **Seguridad criptográfica**: Se sugiere el uso de SSL para establecer conexiones seguras entre dos partes.
- ✓ **Interoperabilidad**: Programadores deben poder desarrollar aplicaciones basadas en SSL, que intercambien parámetros criptográficos sin tener conocimiento de los códigos de los programas de cada uno.
- ✓ **Extensibilidad**: SSL provee un marco donde pueden incorporarse métodos criptográficos según se necesite.

Open Secure Socket Layer

El proyecto de ***OpenSSL*** es un esfuerzo de colaboración para desarrollar una ***Toolkit*** (Caja de Herramientas) una robusta, comercial, completamente equipada, y de código abierto, para implementar ***Secure Socket Layer*** (Zócalos en ejecución Seguros, SSL v2/v3) y ***Transport Layer Security*** (Potocolos de Seguridad de la Capa de Transporte, TLS v1), así como una biblioteca de criptografía robusta con el nivel de una herramienta profesional. El proyecto es manejado por una comunidad mundial de voluntarios que utilizan Internet para comunicar, planear, y desarrollar el *Toolkit* de OpenSSL y su documentación relacionada. OpenSSL[8] se basa en la biblioteca excelente de SSLeay desarrollada por Eric A. Young y Tim J. Hudson. La caja de herramientas de OpenSSL se certifica bajo licencia de Open Source, al estilo de Apache, que básicamente son los medios que en lo que se está libre conseguirla y utilizar para propósitos comerciales y no comerciales conforme a algunas condiciones simples de la licencia. La última versión de OpenSSL es la 0.9.8 que ya está disponible. Para mayor información http://www.openssl.org/

Para instalar **Apache con soporte para SSL**, el módulo ***mod_ssl*** proporciona el soporte necesario para implementar SSL sobre HTTP. Cabe recordar que, hay que tener previamente instalado el paquete de OpenSSL.

HTTPs

Es una versión segura del protocolo HTTP. El sistema HTTPS utiliza un cifrado basado en las Secure Socket Layers (SSL) para crear un canal cifrado (cuyo nivel de cifrado depende del servidor remoto y del navegador utilizado por el cliente) más apropiado para el tráfico de información sensible que el protocolo HTTP. Es utilizado principalmente por entidades bancarias, tiendas en línea, y cualquier tipo de servicio que requiera el envío de datos personales o contraseñas.
El puerto estándar para este protocolo es el 443. Para ello, se realizan los siguientes pasos:

1. El cliente establece una conexión a un servidor que soporta HTTPS.
2. Se negocian los parámetros de comunicación.
   - ✓ El cliente notifica al servidor cuales son los parámetros que soporta y el servidor selecciona cuales de ellos utilizarán. Entre los parametros tenemos: que algoritmos de cifrado se utilizarán, la versión del protocolo, etc.
3. El servidor envía al cliente su certificado digital (x509).
4. El cliente utiliza su copia de la llave pública de la CA, la fecha y el nombre del servidor para validar el certificado.
5. Se realiza un acuerdo de llave, El cliente genera un valor aleatorio "Clave premaestra", lo cifra con la llave pública del servidor y se lo envía.
6. El servidor debe descifrar con su llave privada la "clave premaestra" (Autenticación)

[8] http://www.openssl.org/

7. Tanto el cliente como el servidor deben generar la “clave maestra” a partir de la “premaestra”.
8. El cliente y el servidor intercambia cifrado el Message Digest de la “clave maestra”, para asegurar que obtuvieron lo mismo. (Integridad y Confidencialidad)
9. A partir de la “clave maestra” ambos generán la clave de sesión con la cual se cifrará la comunicación.

4.4 Políticas de Seguridad

A las *reglas* en la que se incluyan las propiedades de *confidencialidad, integridad y disponibilidad*, en la medida requerida por una organización, se les conoce como Política de Seguridad. Por lo anterior, las políticas deben ser **sucintas**, **claras y entendibles**, **prácticas**, **cooperativas** y **dinámicas**. Es importante conocerlas y respetarlas, por que puede ser un hueco en la seguridad de la empresa. Puede haber sanciones a quienes no cumplan las políticas establecidas por el departamento de seguridad, Ayuda al fortalecimiento de la seguridad de la empresa. Cabe recordar que 80% son políticas y el 20% son mecanismos. La correcta implementación de estas reglas, mediante servicios o mecanismos de seguridad, garantizará la seguridad del sistema. Si algún usuario incurre en dar mal uso a su cuenta, y por ende a los recursos que están asociados a ella, podrá en un momento dado ser sancionado por el Consejo de Seguridad de su organización. Las Políticas de Seguridad, ejecutan las siguientes acciones:

- ✓ Especifica las características de seguridad que un sistema debe observar y proveer
  - ➢ Conjunto de reglas que deben respetarse para mantener la seguridad de la información.
- ✓ Especifica las amenazas contra las que la organización debe protegerse y cómo debe protegerse
- ✓ Depende de los objetivos y metas de la organización.
- ✓ Generalmente es expresada en un lenguaje o idioma.
- ✓ Típicamente establecida en términos de *sujetos* y *objetos.*

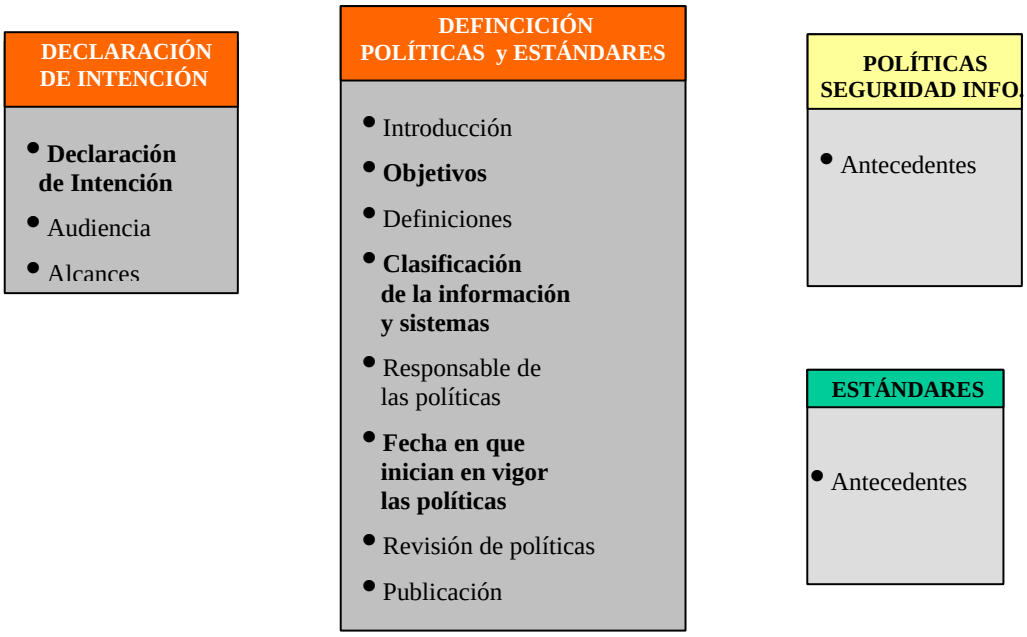


Figura 4.5 - Estructura de las Políticas

Ejemplo de una política.

Las actividades que se mencionan a continuación están prohibidas en general. Bajo ninguna circunstancia los empleados de Institución X podrán realizar actividades ilegales de ningún tipo. Las siguientes actividades están estrictamente prohibidas, sin excepciones, salvedades ni desviaciones.

1. Violar los derechos de cualquier persona o compañía protegidos por derechos de autor, *copyright*, marcas registradas.
2. La instalación o distribución de productos "piratas" o "pirateados" u otros productos de software que no estén autorizados y debidamente licenciados para el uso expreso de la empresa X
3. La introducción e instalación de programas maliciosos en cualquier medio, dispositivo, servidor o estación de trabajo de la red de cómputo de la empresa X. (Ej. Programas *sniffers*, de desencriptación de *passwords*, virus, gusanos, caballos de Troya, etc.).
4. Revelar el USER ID y el *password* a cualquier persona, incluso dentro de la organización o permitir el uso de la cuenta de usuario por otras personas..
5. Buscar, generar, explotar brechas de seguridad informática en los sistemas de información de la empresa X
6. Realizar, generar, explotar o realizar interrupciones en la red de comunicaciones de la empresa X.
7. Búsquedas de puertos o de vulnerabilidades de seguridad informática está explícitamente prohibido.
8. Ejecutar cualquier tipo de monitoreo a las redes de cómputo y comunicaciones de la empresa X. Que intercepten información que no está destinada para el empleado.
9. Burlar o esquivar el proceso de identificación y autenticación de usuario en cualquier sistema de información de la empresa X.
10. Realizar y perpetrar ataques de negación de servicios contra cualquier sistema de información de la empresa X.
11. Las actividades que se mencionan a continuación están prohibidas en general. Bajo ninguna circunstancia los empleados de Institución X podrán realizar actividades ilegales de ningún tipo. Las siguientes actividades están estrictamente prohibidas, sin excepciones, salvedades ni desviaciones.

4.5 Mecanismos de seguridad

Son la parte más visible de un sistema de seguridad. Se convierten en la herramienta básica para garantizar la protección de los sistemas o de la propia red.

Prevención

Incluye funciones de seguridad en hardware y software, también debe incluir la definición y observancia de políticas de seguridad, incluye controles administrativos y es la estrategia más ampliamente usada.

Mecanismos Prevención

- ✓ Aumentan la seguridad de un sistema durante el funcionamiento normal de éste.
- ✓ Previenen la ocurrencia de violaciones a la seguridad
- ✓ Ejemplos mecanismos:
  - ➢ Cifrado durante la transmisión de datos
  - ➢ Passwords difíciles de adivinar
  - ➢ Firewalls
  - ➢ Biométricos

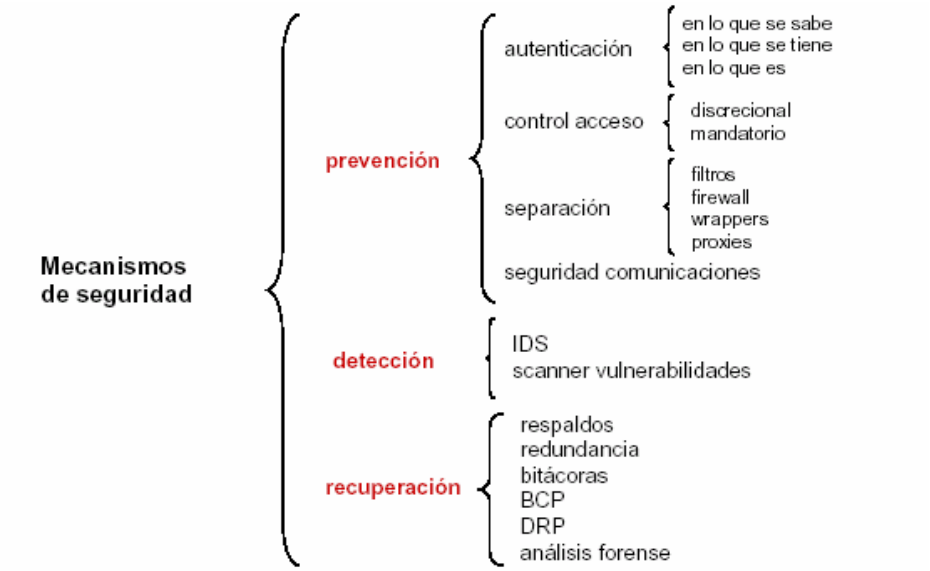


Figura 4.6 – Diagra de los Mecanismos de Seguridad

Mecanismos de prevención más habituales

Mecanismos de autenticación → Hacen posible identificar entidades del sistema de una forma única. Posteriormente, una vez identificadas, son autenticadas (comprobar que la entidad es quién dice ser)

- ✓ **Basados en algo que se sabe** → Se basan en passwords, frases y números de identificación personal, NIP. Siguen siendo el sistema de autenticación más usado hoy en día.
  - ➢ Los primeros sistemas de autenticación se basaron en claves de acceso: nombre usuario y una clave de acceso.
  - ➢ Son fáciles de usar y no requieren de un hardware especial.
  - ➢ Siguen siendo el sistema de autenticación más usado hoy en día.

- ✓ **Basados en algo que se es** → Son Biométricas y comportamiento. Se realiza una medición física y se compara con un perfil almacenado con anterioridad,

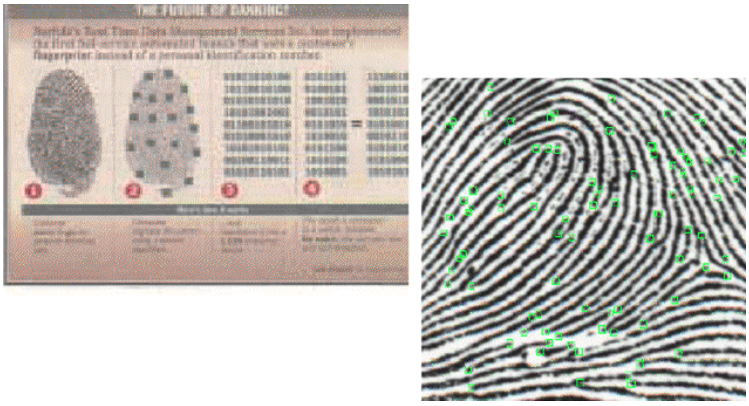


Figura 4.7 - Biométricas

- ✓ **Basadas en algo que se tiene** → Usan un objeto físico que llevan consigo y que de alguna forma comprueba la identidad del portador. Pueden ser tokens, tarjetas inteligentes y pases.

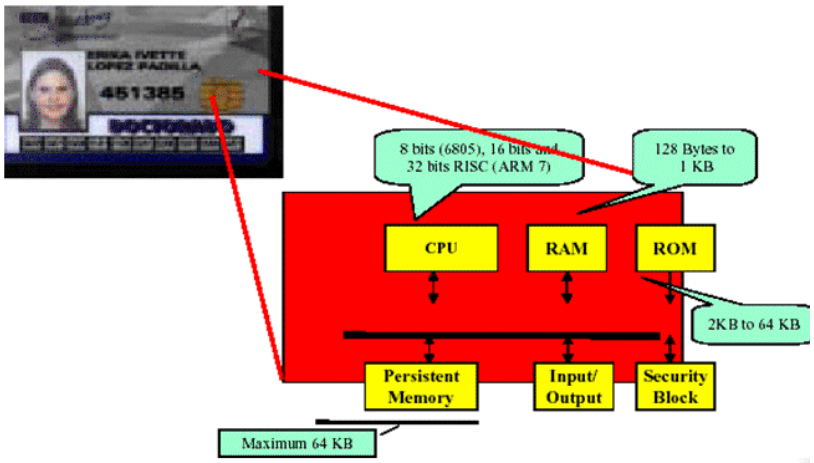


Figura 4.8 – Tarjeta Inteligente

Mecanismos de control de acceso → Usados para proteger objetos del sistema (archivos, recursos..). Controlan todos los tipos de acceso sobre el objeto por parte de cualquier entidad del sistema. Hay dos tipos: discrecional (DAC) y mandatorio/obligatorio (MAC)

- ✓ La autenticación pretende establecer quién eres.
- ✓ La autorización (o control de accesos) establece qué puedes hacer con el sistema.
- ✓ Dos modelos: DAC y MAC

- ✓ Control de acceso discrecional (DAC),
  - ➢ Un usuario bien identificado (típicamente, el creador o propietario' del recurso) decide cómo protegerlo estableciendo cómo compartirlo, mediante controles de acceso impuestos por el sistema.
- ✓ Control acceso mandatorio (MAC) tiene una etiqueta de seguridad.
  - ➢ Es el sistema quién protege los recursos.
  - ➢ Todo recurso del sistema, y todo usuario

Mecanismos de separación → Permiten separar los objetos dentro de cada nivel, evitan el flujo de información entre objetos y entidades de diferentes niveles siempre que no exista una autorización expresa del mecanismo de control de acceso.

- ✓ Definición de un perímetro de seguridad
- ✓ Definir las zonas “abiertas” y las zonas cerradas.
  - ➢ DMZ: Zona desmilitarizada
- ✓ Mecanismos que sirven para delimitar una frontera
  - ➢ Filtros de paquetes
  - ➢ switch
  - ➢ Ruteadores
  - ➢ Proxies

Mecanismos de seguridad en las comunicaciones → Protegen la integridad y privacidad de los datos cuando se transmiten a través de la red, la mayor parte se basan en la criptografía y, manejan el uso de protocolos seguros

Detección

Busca descubrir incidentes al momento en que ocurren o lo antes posible, debe permitir detectar eventos para reducir el daño, permite identificar y perseguir culpables, revela vulnerabilidades.

Mecanismos de detección

Son aquellos que se utilizan para detectar violaciones de la seguridad o intentos de violación. Por ejemplo

- ✓ IDS
- ✓ Tripwire
- ✓ Proxies
- ✓ Detectores de vulnerabilidades
  - ➢ Nessus
  - ➢ Nikto
- ✓ Pen-test
  - ➢ CORE IMPACT
  - ➢ CANVAS
  - ➢ TWISTER

Recuperación

Se pone en marcha después de un incidente de seguridad, incluye la restauración de los recursos dañados, debe remediar las vulnerabilidades que permitieron el incidente.

Mecanismos de recuperación

Son aquellos que se aplican cuando una violación del sistema se ha detectado, para retornar a éste su funcionamiento correcto. Por ejemplo:

- ✓ **Respaldos** → Copia de los datos escrita en cinta u otro medio de almacenamiento duradero. De manera rutinaria se recuerda a los usuarios de computadoras que respalden su trabajo con frecuencia. Los administradores de sitios pueden tener la responsabilidad de respaldar docenas o incluso cientos de máquinas
- ✓ **Redundancia** → Se refiere al procedimiento a través del cual un sistema operativo registra eventos conforme van ocurriendo y los preserva para un uso posterior. Es posible configurar los sistemas de tal forma que los eventos:
  - ➢ Se escriban en uno o en distintos archivos,
  - ➢ Se envien a través de la red a otra computadora,
  - ➢ Se transmitan a algún dipositivo.

  Algunos comandos en unix
  - ➢ lastlog
  - ➢ last

DRP (Disaster Recovery Planning)

Busca recuperar la operación de los servicios computacionales y de telecomunicaciones después de un desastre. El Desastre es un evento no planeado que ocasiona la “no disponibilidad” de los servicios informáticos por un periodo de tiempo tal que, para restablecer estos servicios, es necesario utilizar facilidades alternas de cómputo y telecomunicaciones en otra localidad

BCP (Business Continuity Planning)

Es la capacidad para mantener la continuidad de las operaciones, y está dirigido a situaciones catastróficas (no problemas rutinarios)

Planes de contingencia

- ✓ Consiste en un análisis pormenorizado de las áreas que componen una organización para establecer una política de recuperación ante un desastre:
  - ➢ Es un conjunto de datos estratégicos de la empresa y que se plasma en un documento con el fin de protegerse ante eventualidades.
- ✓ Además de aumentar su seguridad la empresa también gana en el conocimiento de fortalezas y debilidades.
- ✓ Si no lo hace, se expone a sufrir una pérdida irreparable mucho más costosa que la implantación de este plan.

Las empresas dependen hoy en día de los equipos informáticos y de todos los datos que hay allí almacenados (nóminas, clientes, facturas, ...). Dependen también cada vez más de las comunicaciones a través de redes de datos.

Si falla el sistema informático y éste no puede recuperarse, la empresa puede desaparecer porque no tiene tiempo de salir nuevamente al mercado con ciertas expectativas de éxito, aunque conserve a todo su personal.

Perdidas por no contar un plan

- ✓ Pérdida de clientes.
- ✓ Pérdida de imagen.
- ✓ Pérdida de ingresos por beneficios.
- ✓ Pérdida de ingresos por ventas y cobros.
- ✓ Pérdida de ingresos por producción.
- ✓ Pérdida de competitividad.
- ✓ Pérdida de credibilidad en el sector.

Servicios propuestos por OSI

- ✓ Norma 7498-2
- ✓ Autentificación.
- ✓ Control de Acceso.
- ✓ Confidencialidad.
- ✓ Integridad de Datos.
- ✓ No Repudiación.

Autenticación:

La autentificación se refiere a demostrar la identidad de las entidades involucradas en la transacción. Evita que alguien tome la identidad de otro. Generalmente toma dos formas:

- ✓ Autentificación del proveedor de bienes o servicios.
- ✓ Autentificación del cliente.

Control de acceso:

- ✓ Permite definir quién puede tener acceso a ciertos recursos, dependiendo de los privilegios o atributos que posea.
- ✓ Permite proteger los recursos del sistema contra el uso no autorizado.
- ✓ Se basa en credenciales o atributos.
- ✓ Se aplica a los usuarios y procesos que ya han sido autentificados.

Confidencialidad:

- ✓ Servicio que garantiza la privacidad de los datos:
  - En local.
  - En modo no conectado.
  - En flujo de datos
  - En conexiones.
  - En campos selectos.
- ✓ Principal mecanismo para implementar este servicio es la criptología.

Integridad de datos:

- ✓ Permite proteger los datos contra ataques activos.
  - Con recuperación de datos.
  - Sin recuperación de datos.
  - En campos selectos.
- ✓ Los mecanismos principales son:
  - CRCs
  - huellas digitales.

No repudiación:

- ✓ Permite comprobar las acciones realizadas por el origen o destino de los datos.
  - Con prueba de origen.
  - Con prueba de entrega.
- ✓ Los mecanismos principales son los certificados, la notarización y las firmas digitales.

Es necesario garantizar que alguien que haya recibido un pago no pueda negar este hecho. Y es necesario garantizar que alguien que haya efectuado un pago no pueda negar haberlo hecho.

4.6 Mecanismos de Confianza

Redes Privadas Virtuales (VPNs *Virtual Private Networks*)

VPN es una tecnología de red que permite una extensión de la red local sobre una red pública o no controlada, como por ejemplo Internet.

El ejemplo más común es la posibilidad de conectar dos o más sucursales de una empresa utilizando como vínculo Internet, permitir a los miembros del equipo de soporte técnico la conexión desde su casa al centro de cómputo, o que un usuario pueda acceder a su equipo doméstico desde un sitio remoto, como por ejemplo un hotel. Todo esto utilizando la infraestructura de Internet.

Para hacerlo posible de manera segura es necesario proveer los medios para garantizar la autenticación, integridad y confidencialidad de toda la comunicación:

- ✓ **Autenticación** y **autorización**: ¿Quién está del otro lado? Usuario/equipo y qué nivel de acceso debe tener.
- ✓ **Integridad**: La garantía de que los datos enviados no han sido alterados.

- ✓ **Confidencialidad**: Dado que los datos viajan a través de un medio potencialmente hostil como Internet, los mismos son susceptibles de interceptación, por lo que es fundamental el cifrado de los mismos. De este modo, la información no debe poder ser interpretada por nadie más que los destinatarios de la misma.

Metodología VPN básica

- ✓ Concepto básico
  - ➢ Asegurar vía mecanismos de cifrado el canal comunicación.
- ✓ Comunicación puede asegurarse a diferentes capas:
  - ➢ Aplicación (PGP o SSH)
    - ▪ Varios programas trabajan de host a host
    - ▪ Sólo protegen el payload del paquete y no el paquete
  - ➢ Transporte (SSL)
    - ▪ Contenido comunicación es protegido, pero paquetes no
  - ➢ Red (IPSec)
    - ▪ No sólo cifra el payload sino la información TCP/IP
    - ▪ Posible si los dispositivos usan encapsulación
  - ➢ Enlace de datos: Layer 2 Tunneling Protocol (L2TP)
    - ▪ Cifrado de paquetes sobre PPP

Ventajas/Desventajas VPN

- ✓ Beneficios de VPN
  - ➢ Seguridad
  - ➢ Ventajas de implementación
  - ➢ Costos
- ✓ Desventajas de VPN
  - ➢ *overhead* (sobrecarga) en el procesamiento
  - ➢ *overhead* en los paquetes
  - ➢ aspectos de implementación
  - ➢ aspectos de control y manejo de errores
  - ➢ aspectos de disponibilidad Internet
  - ➢ Difícil de monitorear si una máquina cliente de nuestra VPN ha sido comprometida.

4.7 Sistemas de Detección de Intrusos (Intrusion Detection System IDS)

Se trabaja mediante un software específicamente diseñado para reconocer los patrones de un comportamiento no deseado. Puede proporcionar un medio para registrar intentos, detener intentos en progreso y cerrar hoyos que coincidan con algunos patrones de ataques, bloqueando la secuencia para que ya no ocurra. Son un conjunto de herramientas para identificar y manejar riesgos.

El sistema ideal de detección de intrusos notificará al administrador de sistema/red la existencia de un ataque en proceso:

- ✓ Con 100% de exactitud.
- ✓ Inmediatamente (t<1min).
- ✓ Con un diagnóstico completo del ataque.
- ✓ Con recomendaciones de como bloquear el ataque.

Objetivos de un IDS

100% de exactitud y 0% de falsos positivos → Un falso positivo ocurre cuando el sistema genera una falsa alarma. Se podría ejemplificar como el síndrome de Juanito y el Lobo…

Generar un 0% de falsos negativos es un objetivo adicional → No permitir que ningún ataque pase desapercibido.

Para que haya una notificación oportuna, el canal de notificación debe estar protegido; el atacante puede bloquear/inutilizar el mecanismo de notificación.

Un IDS que usa e-mail como canal de notificación va a tener problemas al informar que el servidor de correo esta siendo atacado.

Diagnósticos

Idealmente el IDS categorizará/identificará el ataque:

- ✓ Demasiado detalle para el administrador de red, al tener que conocer íntimamente los ataques.
- ✓ Difícil de implementar:
- ✓ Especialmente cuando las cosas "se ven raras" y no concuerdan con ataques conocidos.

La utopía

- ✓ El IDS perfecto no sólo identificará un ataque, también:
  - ➢ Evalúa la vulnerabilidad del blanco.
  - ➢ Si el blanco es vulnerable informará al administrador.
  - ➢ Si la vulnerabilidad tiene un "parche" conocido, indicará al administrador como aplicarlo.
- ✓ Esto requiere cantidades tremendas de conocimiento incorporado al sistema.

Características de los IDS

- ✓ Un IDS razonablemente efectivo puede identificar:
  - ➢ Ataques internos.
  - ➢ Ataques externos.
- ✓ Permite al administrador medir la cantidad de ataques que está sufriendo la infraestructura.
- ✓ Puede funcionar como respaldo de seguridad perimetral en caso de falla de otros sistemas (firewall, filtros).

Detección de ataques contra Intrusos

Idealmente efectuar dos detecciones:

- ✓ Siendo realista, primero efectuar detección de intrusos y después detección de ataques.
- ✓ liberar primero la detección de ataque para justificar la decisión ante la directiva, después liberar la detección de intrusos.

La pregunta importante tiene que ver con los costos para reaccionar ante alarmas generadas por el sistema de detección de ataques.

Limitantes de los IDS

- ✓ No pueden monitorear el tráfico que viaja cifrado.
- ✓ Muchos datos a analizar y más si es un IDS basado en red.
- ✓ Los IDS de aplicación sólo funcionan para un protocolo en particular.
- ✓ Los IDS basados en firmas no reconocerán ataques que no se encuentren en sus bases de datos.

- ✓ Para los IDS basados en host necesitan mucho poder de cómputo y cuando detectan un nuevo ataque este ya se ejecuto con éxito.
- ✓ Problemas con los ataques “UNICODE”

Ejemplos de IDS

- ✓ Snort
- ✓ ISS Real Secure
- ✓ Cisco NetRanger
- ✓ EMERALD
- ✓ TRIPWIRE
- ✓ IDS-SQUID

Firewall

Básicamente examina el tráfico en la red, tanto entrante como saliente y lo examina en base a ciertos criterios, para determinar si lo deja pasar o lo descarta. Si detectan algo anormal pueden tener procedimientos a seguir o poner en aviso al administrador. Existen 2 Tipos básicos de Firewall:

- ✓ Hardware Firewall: Normalmente es un ruteador, tiene ciertas reglas para dejar o no dejar pasar los paquetes.

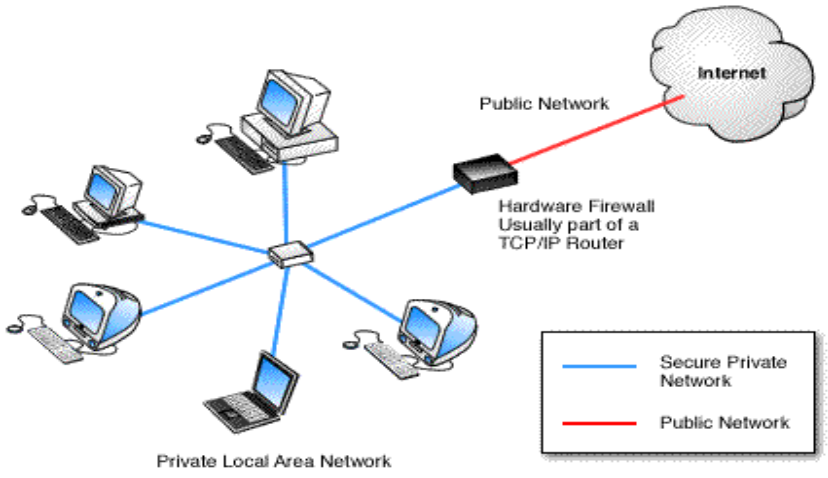


Figura 4.9 – Firewall por Hardware

- ✓ Software Firewall: Es un programa que esta corriendo preferentemente en un bastioned host, que verifica los paquetes con diferentes criterios para dejarlos pasar o descartarlos.

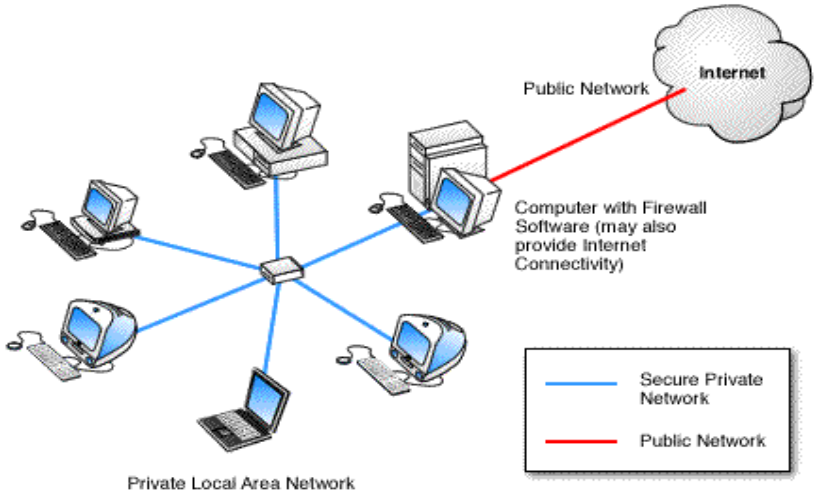


Figura 4.10 – Firewall por software

Stateful firewall → Firewalls que intentan dar seguimiento a una conexión cuando se tiene filtrado de paquetes. Se pueden considerar entre un filtro de paquetes y un proxie. Predominantemente examinan capa 4 e información paquetes más baja frecuentemente verifican sólo capa 7 (aplicación). Si el paquete coincide con regla del firewall que permite su paso, se crea una entrada en la tabla de estados paquetes posteriores de la misma conexión son permitidos sin realizar más inspecciones

Un Firewall trabaja en multiples capas, dependiendo del uso que se le quiera dar a éste.

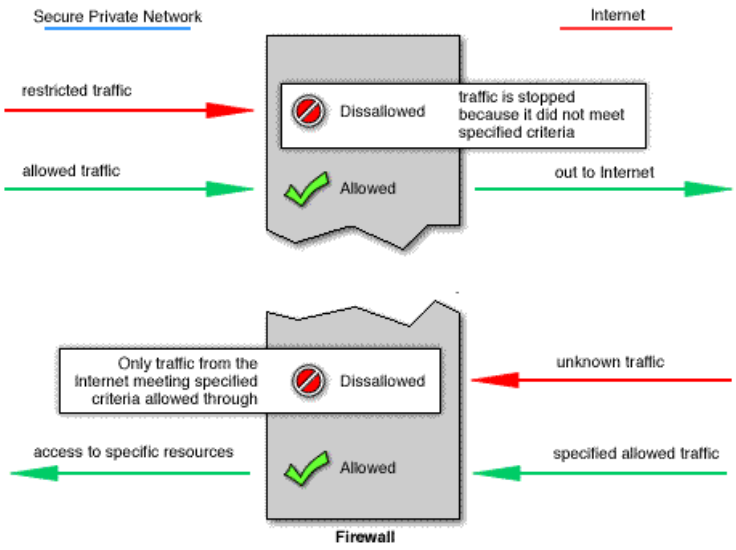


Figura 4.11 – Operación de un Firewall

Para implementar un Firewall es necesario:

- ✓ Determinar el nivel de Seguridad requerido.
- ✓ Determinar el tráfico que va a entrar a la red.
- ✓ Determinar el tráfico que va a salir de la red.
- ✓ Alternativas.

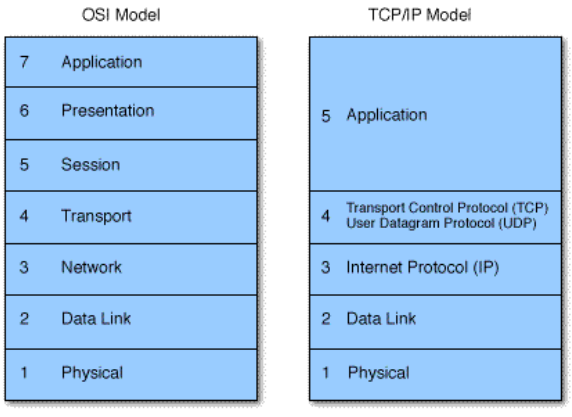


Figura 4.12 – Operación de un Firewall

Un ***Proxy/Server*** es un intermediario entre la red interna y el internet, puede implementar ciertos criterios de seguridad, así como también caché, lo que significa que si el proxy tiene una petición, lo primero que hace es buscar en su cache, si lo encuentra responde a la petición.

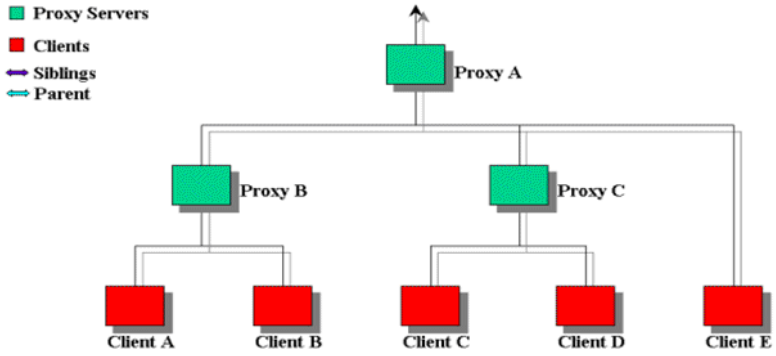


Figura 4.13 – Distribución de Proxy/Servers

Firewalls en Linux

Iptables / Netfilter son las dos piezas principales de producto firewall disponibles gratuitamente para distribuciones Linux. IPTables es usado para construir las reglas, Netfilter es puente entre núcleo linux y las IPTables. IPTables es como se conoce al módulo Netfilter (Herramienta estándar actual de firewall de Linux). Los administradores especifican que reglas que protocolos o tipos de tráficos se deben seguir. Linux cuenta con filtrado paquetes (IPFW) incorporado en su núcleo desde versión 1.1 (Adaptación herramienta ipfw del sistema operativo BSD).

Netfilter/Iptables

Los dos piezas principales de producto firewall disponibles gratuitamente para distribuciones Linux

- ✓ IPTables es usado para construir las reglas.
- ✓ Netfilter es puente entre núcleo linux y las IPTables

IPTables es como se conoce al módulo Netfilter, que es la herramienta estándar actual de firewall de Linux. Los Administradores especifican que reglas, que protocolos o tipos de tráficos se deben seguir. Cuando se empieza la conexión con protocolos IPTables se añade una entrada de estado para la conexión en cuestión

El Filtrado de IPtables desarrolla diferentes actividades:

- ✓ Filtrado de dirección remota
- ✓ Filtrado de dirección destino local
- ✓ Filtrado de puerto de origen remoto y destino local
- ✓ Filtrado del estado de la conexión TCP
- ✓ Sondeos y exploraciones de puertos
- ✓ Ataque DoS

La utilería *iptables*

Es una Herramienta para crear las reglas del *firewall,* si se desea un conocimiento exhaustivo de todas las opciones de la utilidad iptables, lo mejor es consultar el manual. También se cuenta con un HowTo

Cabe recordar que:

- ✓ Las Tablas están compuestas de listas o reglas (cadenas).
- ✓ Regla es un par condición/acción sobre atributos paquetes.
- ✓ Paquete pasa secuencialmente por cada una de las reglas.
- ✓ Si el paquete cumple con regla se realiza una acción regla.
- ✓ Si el paquete no cumple con ninguna regla, se ejecuta la acción por default asociada a dicha cadena.

Con lo visto en este capítulo, se observa cual es, de manera general, la forma de trabajo en que la seguirad opera. También los métodos y técnicas que se utilizan para garantizar que la integridad de la información y del servidor se conserver en perfecto estado, que sean inalterables por agentes externos y dañinos para los datos. Con ello se puede comenzar a trabajar de manera dedicada y segura con el servidor y los servicios que éste brinde, programando páginas web, manejando bases de datos, etc.

CAPITULO V

CREACIÓN DE PÁGINAS WEB

HTML es un lenguaje de marcado de hipertexto en el que se escriben las páginas a las que se accede a través de navegadores WWW. Admite componentes hipertextuales y multimedia. Ha sido revisado varias veces, en los cuales se han modificado los estándares y las especificaciones para una creación más efectiva y sencilla de sitios web.

HTML es un lenguaje de Markup basado en el SGML[1] (Standard Generalized Markup Language), al cual se le agregaron vínculos de hipertexto. Cada browser presenta la página escrita en HTML de acuerdo a su estructura de Markup. La forma exacta de interpretación depende de cada browser, por ejemplo los browsers visuales tales como Internet Explorer y Netscape Navigator presentan las páginas en pantalla, en cambio, los browsers para no videntes van a leer el contenido de acuerdo al Markup. Una página escrita en HTML puede ser creado mediante cualquier editor de texto ya que se trata de un simple archivo de texto.

Historia de HTML

• **1989**: Tim Berners-Lee en el CERN (*European Laboratory for Particle Physics – Laboratorio Eurpeo para Fisicos Independientes*) define una versión muy simple de HTML basado en ***SGML*** , como parte de un sistema basado en red para compartir documentos por medio de navegadores (*browser*s) de texto y que fuera simple de aprender y de mostrar.

• **1992-3**: un grupo del NCSA (*National Center for Supercomputing Applications, USA*) desarrolla el **MOSAIC**, un *browser* visual y gráfico, agregando al HTML soporte para imágenes, listas anidadas y formularios. Fue el inicio del WEB.

• **1994**: Varias personas del grupo de MOSAIC ayudan a empezar a Netscape.

• **1994**: Se forma el **W3 Consortium (W3C)** con sede en el MIT, como una organización con soporte industrial para la estandarización y desarrollo del Web.

• **1997**: Se crea el primer estándar de HTML: HTML 3.2

• **1999**: (Diciembre): HTML 4.01 es la recomendación del W3C, que empieza a separar claramente la estructura del documento de los aspectos de la presentación y que especifica claramente las relaciones entre HTML y los client-side scripts (JavaScripts).

HTML 4 viene en 3 diferentes tipos: ***Strict*****,** ***Transitional*** **y** ***Frameset***. Los últimos dos permiten a los autores pasar del estándar estricto enfatizando la estructura sobre la presentación y usando elementos de presentación y atributos que están marcados como *depreciados* (*deprecated* - cuyo uso se abandonará a corto plazo).

Se pueden especificar estilos de presentación para elementos individuales y agregar diferentes estilos de presentación para diferentes tipos de elementos con *style sheets.* HTML 4.01 no está basado en XML. Agregarle a HTML la sintaxis estricta del XML tiene las siguientes ventajas:

- ✓ • Los elementos de XHTML se pueden usar con otros elementos definidos por XML
- ✓ • Las páginas XHTML se pueden procesar fácilmente con herramientas de XML.

[1] Las siglas de "Standard Generalized Markup Language" o "Lenguaje de Marcación Generalizado". Consiste en un sistema para la organización y etiquetado de documentos. La Organización Internacional de Estándares (ISO) ha normalizado este lenguaje en 1986. El lenguaje SGML sirve para especificar las reglas de etiquetado de documentos y no impone en sí ningún conjunto de etiquetas en especial.

• **2000** (Enero): el W3C lanza XHTML 1.0 como una reformulación de HTML 4 .01 bajo la sintaxis estricta de XML. Para soportar los tres tipos de HTML 4, XHTML 1.0 provee tres **DTD's** (*Document Type Definitions*) y que indican el tipo en la declaración DOCTYPE:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd>

En el primero no se aceptan elementos o atributos *deprecated.* Muchos *browser*s no respetan estas disposiciones y los presentan igual.

• **2001** (Abril): el W3C lanza la recomendación *Modularization of XHTML* agrupando elementos de XHTML relacionados en módulos definidos por *XML DTD Fragments.* Esta organización modular hace que XHTML sea más flexible y extensible.

• **2002** (Agosto): se lanza el *XHTML 2.0 working draft* que actualiza los módulos, elimina el soporte para elementos *deprecated* y arma una base para la evolución futura de XHTML

Criterios de Diseño

Para el diseño de páginas es necesario establecer algunos de los parámetros más importantes a considerar como son los siguientes:

a) Objetivo principal de la página
b) Información necesaria para el desarrollo de la página
c) Software empleado para el desarrollo de la página
d) Manipulación y tratamiento de imágenes
e) Equipo necesario para el montaje y/o desarrollo de la página
f) Publicación y difusión de la página.
g) Actualización y mantenimiento de la página.

Es importante definir el objetivo principal de la página WEB a crear, ya que este parámetro permitirá al diseñador el establecer las prioridades en su desarrollo, así como el enfoque de la información y la presentación de la misma.

Una vez establecido el objetivo de la página, se procede a hacer una recopilación del material escrito y gráfico que permitirá conformar el contenido de la misma, dicha información puede obtenerse haciendo un estudio del tema o bien apoyándose en información y archivos que pueden ser proporcionados por la empresa o de la propia Internet.

Editores de HTML

Los editores de HTML son un software que facilita al usuario la creación de páginas de WEB, empleando para su construcción la filosofía de los procesadores de palabra que consiste en permitir que el usuario vaya capturando el texto y después, mediante las acciones de marcar y seleccionar opciones o botones, se va dando formato y presentación, además de que cuenta con

una serie de asistentes que facilitan al usuario la inserción imágenes y vínculos o ligas de hypertexto hacia otras páginas y servicios. Algunos ejemplos de editores de HTML son Front Page (Microsoft). Hot Dog, Hot Metal, Web Editor Pro, Edit Plus, Quanta Plus

URL

URL es el acrónimo de *(Uniform Resource Locator)*, localizador uniforme de recursos y permite localizar o acceder de forma sencilla a cualquier recurso de la red desde el navegador de la WWW. Las URL se utilizarán para definir el documento de destino de los hiperenlaces, para referenciar los gráficos y cualquier otro fichero que se desee incluir dentro de un documento HTML. Cada elemento de internet tendrá una URL que lo defina, ya se encuentre en un servidor de la WWW, FTP, gopher o las News. El formato de una URL será:

servicio://máquina.dominio:puerto/camino/fichero

Donde:

El *servicio* → Será alguno de los de internet, estos pueden ser:

- ✓ **http**: *(HyperText Transport Protocol)*, es el protocolo utilizado para transmitir hipertexto. Todas las páginas HTML en servidores WWW deberán ser referenciadas mediante este servicio. Indicará conexión a un servidor de la WWW.
- ✓ **https**: *(HyperText Transport Protocol Secure)*, es el protocolo para la conexión a servidores de la WWW seguros. Estos servidores son normalmente de ámbito comercial y utilizan encriptación para evitar la intercepción de los datos enviados, usualmente números de tarjeta de crédito, datos personales, etc ..., realizará una conexión a un servidor de la WWW seguro.
- ✓ **ftp**: *(File Transfer Protocol)*, utilizará el protocolo FTP de transferencia de ficheros. Se utilizará cuando la información que se desee acceder se encuentre en un servidor de ftp. Por defecto se accesará a un servidor anónimo *(anonymous)*, si se desea indicar el nombre de usuario se usará: *ftp://máquina.dominio@usuario*, y luego le pedirá la clave de acceso.
- ✓ **telnet**: Emulación de terminal remota, para conectarse a máquina multiusuario, se utiliza para acceder a cuentas públicas como por ejemplo la de biblioteca. Lo normal es llamar a una aplicación externa que realice la conexión. En este caso se indicará la máquina y el login: *telnet://máquina.dominio@login*.
- ✓ **mailto**: Se utilizará para enviar correo electrónico, todos los navegadores no son capaces. En este caso sólo se indicará la dirección de correo electrónico del destino: *mailto://alias. correo@domino*.

La *máquina.dominio* → Indicará el servidor que proporciona el recurso, en este caso se utilizará el esquema IP para identificar la máquina, será el nombre de la máquina y el dominio. En el caso de la Universidad el dominio siempre será unam.mx. Por tanto un nombre valido de máquina será www.unam.mx.

Es muy importante indicar siempre el dominio, ya que se supone que se conectarán a las páginas desde servidores externos a la red local por tanto si no se indica el dominio, las URL que se especifiquen no podrían ser seguidas por los navegadores externos. Si en vez de www.unam.mx se utiliza www será perfectamente accesible por cualquier máquina de la red local pero si se referenciara desde una red con distinto dominio la máquina www será la máquina llamada así en el dominio remoto si existiera, que no es a la que se desea referenciar.

El ***puerto*** TCP → Es opcional y lo normal es no ponerlo si el puerto es el mismo que se utiliza normalmente por el servicio. Sólo se utilizará cuando el servidor utilice un puerto distinto al puerto por defecto.

El *camino* → Será la ruta de directorios que hay que seguir para encontrar el documento que se desea referenciar, para separar los subdirectorios se utilizará la barra de UNIX ***/***, se usa por convenio al ser este tipo de máquinas las más usadas como servidores. El nombre de los subdirectorios y del fichero referenciado puede ser de más de ocho caracteres y se tendrá en cuenta la diferencia entre mayúsculas y minúsculas en el nombre.

La *extensión de los ficheros* → Será también algo importante, ya que por ella sabe el servidor el tipo de documento que se accede e indica al cliente (navegador) el modo en que debe tratarse ese documento. Para definir los tipos de documentos se utiliza los tipos MIME[2].

El navegador de la WWW, realiza una acción para cada tipo de fichero, sólo los que sean del tipo text/html serán mostrados como documentos HTML. En el caso de que el tipo no sea conocido por el cliente se considerará por defecto como un documento de texto normal.

Si no se indica un fichero y sólo se hace referencia a un directorio, se accesa a la página html por defecto de ese directorio. En el servidor están definidos unos ficheros para ser usados si no se indica un fichero concreto, el nombre que debe tener este fichero es *default.htm* o *default.html*. En caso que no exista este fichero se mostrará un listado de todos los documentos que forman el directorio. Este fichero es la página inicial (*home page*) del servidor o del espacio Web. Algunos ejemplos de URL podrían ser:

5.1 Etiquetas de HTML.

Las etiquetas de HTML son el conjunto de instrucciones o comandos que después son interpretados por los visualizadores de WEB, desplegando como resultado de esta interpretación las páginas WEB. Las Etiquetas de HTML se interpretan en el orden en que se ponen y van encerradas entre pico paréntesis (< >). Estas etiquetas no son sensitivas de tal forma que se pueden escribir con mayúsculas o minúsculas indistintamente.

Estructura básica de una página de HTML

Una página Web contiene dos partes: ***etiquetas de identificación del Markup***, llamados **tags** y contenido. Los tags están enmarcados entre los símbolos **<** y **>** (pico-paréntesis) y pueden tener atributos o modificadores

Como los tags pueden contener a su vez a otros tags (en forma de un árbol), deberá entonces existir un tag de inicio y otro de fin. La diferencia entre los mismos lo da la barra "/" antes del nombre, en el caso del tag de finalización. Si el tag no tiene contenido, entonces la barra estará antes del signo mayor.

Un documento HTML está definido por una etiqueta de apertura <HTML> y una etiqueta de cierre </HTML>. Dentro de éste se dividen dos partes fundamentales la cabecera, delimitada por la etiqueta <HEAD> y el cuerpo, delimitado por la etiqueta <BODY>. Por tanto la estructura de un documento es:

<html>

[2] **MIME** (Multi-Purpose Internet Mail Extensions, Extensiones de correo Internet multipropósito), son una serie de convenciones o especificaciones dirigidas a que se puedan intercambiar a través de Internet todo tipo de archivos (texto, audio, vídeo, etc.) de forma transparente para el usuario. Una parte importante del MIME está dedicada a mejorar las posibilidades de transferencia de texto en distintos idiomas y alfabetos.
En 1991 la IETF (*Internet Engineering Tasking Force* - Grupo de Trabajo en Ingeniería de Interne) comenzó a desarrollar esta norma y desde 1993 todas las extensiones MIME están especificadas de forma detallada en diversos documentos oficiales disponibles en Internet (RFC 1521, 1522, 1523 y otros).

```
<head>
  <title>Espacio para título de la ventana</title>
</head>
<body>
  Espacio para escribir todo el contenido de la página.....
</body>
</html>
```

<html>.....</html> → Es la primera y la última etiqueta en un documento html y le indica al navegador que despliegue el documento como Hipertexto y no como texto sencillo.

<head>…...</head> → La etiqueta encabezado identifica la sección inicial del documento html. Entre otras cosas escribe el titulo del documento de esa sección

<title>...</title> → La etiqueta titulo contiene el titulo que aparece en la barra de titulo del navegador web.

<body>...</body> → La etiqueta de cuerpo abarca la parte más grande de una página Web. Contiene todo lo que el usuario ve cuando entra a esa página

La mayoría de las etiquetas se colocan dentro de BODY. Ninguno de estos elementos es obligatorio, pudiendo componer documentos HTML que se muestren sin ningún problema sin incluir estas etiquetas de identificación. Si se utilizan elementos que forzosamente deban ser incluidos en la cabecera (como la etiqueta de titulo), no serán reconocidos correctamente si no se incluyen entre las etiquetas de <HEAD>.

No todos los usuarios pueden apreciar las páginas web de la misma manera, esto se debe a varios motivos:

- ✓ Pequeñas diferencias entre los browsers.
- ✓ No todos los usuarios tienen instaladas las mismas fuentes.
- ✓ Diferentes resoluciones de pantalla.
- ✓ Tamaño de la ventana.

Comentarios

En HTML para insertar los comentarios dentro de un documento HTML se deberán encerrar entre "**<!--**" y "**-->**", definiéndose un comentario de la forma:

```
<!-- Esto es un comentario que no se imprime en pantalla-->
```

Los comentarios son útiles para identificar el documento, pudiendo indicar al comienzo del documento su titulo, autor y la fecha en el que fue realizado, esto se hace antes de la etiqueta <HTML>. También aunque ya con menos utilidad para comentar cualquier instrucción o circunstancia del documento. Los comentarios no se muestran en el documento HTML y el único modo de verlo es viendo el código HTML fuente del documento, cosa que permiten algunos navegadores de la WWW.

Cabecera de un documento HTML

La cabecera de un documento HTML está delimitado por las etiquetas **<HEAD>** y **</HEAD>**, en esta se incluirán las definiciones generales que afectarán a todo el documento. Todas sus etiquetas son opcionales y se utilizarán sólo en casos muy determinados, sólo la etiqueta TITLE tiene un uso general y aunque es opcional se recomienda incluirla en todos los documentos que se crean.

A continuación se citan los distintos componentes que pueden formar la cabecera de un documento HTML.

<TITLE>: Título del documento → Especificará el título del documento HTML, todo documento HTML debe tener un titulo. El titulo debe guardar relación con el contenido del documento y definirlo de forma general, su tamaño no está limitado aunque se recomienda que no sea excesivamente extenso. Se utiliza principalmente para etiquetar e identificar la página en los *bookmarks* (marcadores) y las *history list* (lista de documentos accedidos) y también se utiliza por algunos servidores de búsqueda como resultado de una búsqueda para poder intuir el contenido del documento. El titulo deberá guardar relación con el contenido del documento y definirlo de forma general. La etiqueta <TITLE>, debe ser usada dentro de las etiquetas que definen la cabecera de la siguiente forma:

```
<HEAD>
<TITLE>Titulo del documento HTML</TITLE>
</HEAD>
```

Cuerpo de un documento HTML

El cuerpo de un documento HTML estará delimitado por las etiquetas <BODY> y </BODY> y en él se incluirán todas las instrucciones HTML y el texto que forman el documento, es similar al *BEGIN* o *{* de un lenguaje de programación. Al igual que la cabecera (HEAD) es opcional, pero se recomienda para la buena identificación de las distintas zonas del documento. Si un documento no presenta ninguna de las etiquetas de identificación de sus distintas partes (<*HTML*>, <*HEAD*> o <*BODY*>) se considerará que todo lo que se defina pertenece al cuerpo del documento.

Atributos de <BODY> → La etiqueta BODY presenta algunos atributos que son de definición global para todo el documento, estos definirán los colores y el fondo del documento HTML. Los atributos de BODY son:

```
<BODY BACKGROUND="URL" BGCOLOR=#rrvvaa TEXT=#rrvvaa LINK=#rrvvaa VLINK=#rrvvaa
>
```

Donde:

- ✓ **BACKGROUND="URL"** → Definirá la imagen que se utilizará de fondo del documento HTML, la URL definida será el camino a una imagen. Esta se muestra debajo del texto y las imágenes del documento HTML. En el caso de que la imagen no rellene todo el fondo del documento, ésta será reproducida tantas veces como sea necesario.
- ✓ **BGCOLOR="#rrvvaa" o "*nombre del color*"** → Indicará el color del fondo del documento HTML, sólo se utilizará si no se ha definido una imagen de fondo, o si esta imagen no puede obtenerse.
- ✓ **TEXT="#rrvvaa"** o **"*nombre del color*"** → Especificará el color del texto normal dentro del documento HTML. Por defecto será normalmente negro.
- ✓ **LINK="#rrvvaa"** o **"*nombre del color*"** → Indicará el color que tendrán los hiperenlaces que no han sido accedidos. Por defecto será azul.

- ✓ **VLINK= "#rrvvaa"** o **"*nombre del color*"** → Color de los enlaces que ya han sido visitados. Por defecto es un color azul más oscuro.

Espaciados y saltos de línea

El espaciado en HTML

En HTML no está permitido más de un elemento blanco (espacios, tabuladores, saltos de línea) separando cualquier elemento o texto, todos estos son convertidos a un único espacio blanco y el resto se omiten en la representación del documento. En el documento fuente se puede usar el espaciado que se desee, y no deberá estar bien formateado, este se conseguirá con las etiquetas HTML. Existen unas etiquetas especiales HTML para definir estos elementos de control de texto. A continuación se detallará cada una de ellas.

<P> Cambio de párrafo

Definirá un párrafo, se usará al comienzo o al final de un párrafo de texto e introducirá un espaciado de separación (normalmente dos líneas) con el próximo texto que se exprese. El efecto se conseguirá introduciendo la etiqueta <P> en el punto en el que se desea que se produzca el espaciado. La etiqueta de fin de párrafo </P> es opcional no siendo necesario incluirla.

Existen elementos HTML que ya introducen separaciones de líneas, tanto antes como despues, por tanto en estos casos no es necesario, pero sí posible, introducir el elemento de nuevo párrafo. No es necesario utilizar esta etiqueta ni antes ni después de cabeceras <H*n*>, después de <HR> (reglas horizontales), <ADDRESS>, <BLOCKQUOTE>, <PRE>. Tampoco es necesario dentro de listas tras los elementos <LI>, <DT> ni <DD>, que se utilizan para separar los distintos elementos de una lista. Por ejemplo

**
 Salto de línea**

Su utilidad es similar al anterior pero en este caso el espaciado del texto es menor, se pasará a la línea siguiente, sin dejar una línea de separación. En este caso será un cambio de línea y no de parrafo. Igualmente no es necesario usarlo tras los elementos que llevan implícitos un salto de línea, ni tampoco es necesaria la etiqueta de fin <BR />.

<HR> Regla Horizontal

<HR ALIGN=*LEFT*|*RIGTH*|*CENTER* NOSHADE SIZE=*n* WIDTH=*n*>

Se usa para dividir un documento en distintas secciones, mostrará una línea horizontal de tamaño determinable. Se asemejará al salto de página dentro de un documento. Si no se especifican atributos dibujará una línea que ocupe el ancho de la pantalla del navegador e introduciendo una separación con el texto anterior y siguiente, equivalente a cambio de párrafo. No es necesaria la etiqueta de fin </HR>. Con los atributos se puede especificar su forma y tamaño, esto atributos son:

- ✓ **ALIGN=LEFT, RIGTH** o **CENTER** → Indicará la forma en la que se alineará la regla en el caso de no ocupar todo el ancho de la pantalla del navegador, a la izquierda, derecha o centrada.
- ✓ **NOSHADE** → No muestra la sombra de la barra, evitando el efecto en tres dimensiones.
- ✓ **SIZE=*n*** → Indicará la altura de la regla en puntos de la pantalla.
- ✓ **WIDTH=*n*** o ***n*%** → Especificará el ancho de la regla, se puede especificar en tanto por ciento del ancho de la ventana <HR WIDTH=50%> o en valor absoluto <HR WIDTH=75> en puntos de la pantalla.

<PRE> Texto preformateado

Muestra una porción de texto en el que se respetan los saltos de línea, tabuladores y espacios en blanco. Todo lo que se encuentre entre las etiquetas de inicio y fin de texto preformateado se mostrará tal y como se expresa en el fuente del documento html. Para mostrar este texto se utiliza una fuente de espaciado fijo más pequeña que el texto normal.

<CENTER> Centrado de texto e imágenes

Se utilizará para centrar líneas de texto, imágenes o cualquier otro elemento HTML (tablas, listas, etc.) Todo lo que se encuentre entre las etiquetas de inicio y fin aparecerá centrado en el navegador.

** ** (Espacios en blanco) → Con esta secuencia de caracteres se conseguiran espacios en blanco que se mostrarán de forma efectiva, pudiendo mostrar más de un espacio en blanco de separación. Se incluirán tantas expresiones   como espacios en blanco se desee conseguir.

Caracteres Especiales

Los caracteres acentuados y algunos caracteres especiales que usa el lenguaje HTML para definir sus etiquetas no se pueden incluir en un documento de manera normal, se deben utilizar una serie de secuencias de escape que al mostrar el documento se sustituyen por el caracter deseado. Estas secuencias de escape comienzan todas con el símbolo ***ampersand*** (**&**), seguido de un texto (siempre en minúsculas) que define el carácter deseado y termina con el símbolo punto y coma (**;**). El error más común cuando se usan estas secuencias de escape es no utilizar el punto y coma final, en cuyo caso se mostrará el literal que define la secuencia, en lugar del carácter deseado. No es necesario dejar espacios en blanco ni antes ni después de los caracteres especiales, para que queden perfectamente encuadrados en la palabra. Las listas y descripciones de los caracteres especiales aparecen en el Anexo 3.

Cabeceras (Headers)

<H1> - <H6> Cabeceras de títulos

En un documento HTML es posible definir seis tipos distintos de cabeceras que serán normalmente el título del documento y los distintos subapartados que lo forman. La etiquetas que definen las cabeceras serán <H1>, <H2>, <H3>, <H4>, <H5>, <H6>. El texto indicado entre las etiquetas de inicio y de fin será el afectado por las cabeceras: **<H1>Este texto aparecerá aumentado</H1>, y** el resultado será:

Este texto aparecerá aumentado

Tamaño y Color de las fuentes de caracteres

En el apartado anterior se vio el modo de definir los distintos títulos de documento, mediante cabeceras. Estas son poco flexibles y predeterminadas. Existe otra etiqueta HTML que permite una más sencilla adaptación del tamaño de las fuentes y permite además modificar su color. Con ésta se puede incluir texto resaltado en medio de una frase, con las cabeceras no es posible ya que estas introducen automáticamente un salto de línea detrás, y permitirá incluir párrafos de distintos tamaños o colores, proporcionando una mayor versatilidad y pudiendo crear

efectos más espectaculares. La etiqueta que permite esto se llama FONT y presenta atributos que permiten modificar el tamaño y color del texto incluido entre la etiqueta de inicio y fin.

<FONT SIZE=*n*> : Tamaño de la fuente

El atributo SIZE permite indicar el tamaño de la fuente, su valor puede estar entre 1 y 7. Incrementándose de tamaño progresivamente desde 1, que es la fuente de menor tamaño, hasta 7 que la fuente de mayor tamaño. El texto normal equivale a la fuente de tamaño 3, por tanto los valores menores que 3 serán fuentes más pequeñas que el texto normal y las mayores que 3 serán de mayor tamaño. El tamaño también puede indicarse de forma relativa, indicando el incremento o detrimento a partir de la fuente base. Por defecto la fuente base será 3, por tanto si se indica como valor +1 la fuente será de tamaño 4. Los elementos de tamaño de fuentes pueden ser definidos para todo un documento, teniendo validez dentro de elementos tales como listas y formularios, pero no tendrán validez global en las tablas, debiendo definir cada una de las celdas al tamaño de fuente deseado.

Existe una etiqueta que redefine la fuente por defecto, esta etiqueta es: <BASEFONT SIZE ...>

<BASEFONT SIZE=*n*> : Fuente por defecto

Definirá el tamaño de la fuente que se considerará como base para definir los tamaños de fuente relativos. Esta etiqueta no define el tamaño de la fuente por defecto, para el texto normal, y sólo se usa para cálculos de tamaño de fuente relativos. La utilidad no es muy amplia, y se podría utilizar cuando en un documento en el que todos los tamaños estén definidos de forma relativa y se desee cambiar de forma global el tamaño de las fuentes. Si no incluye esta etiqueta el valor base para estos cálculos es 3, con esta se puede variar.

<FONT COLOR=*texto color* ó *rrvvaa(código de color)*> : Color de la fuente

El atributo COLOR permite definir el color que tendrá el texto incluido entre las etiquetas de inicio y fin. Este atributo sólo funciona en el *Internet Explorer de Microsoft* y en el Netscape versión 2.0 o superior. El modo de definir los colores es similar al explicado para los atributos de BODY. Este atributo puede ser definido de forma conjunta con el atributo SIZE, o de forma independiente, siendo atributos de la misma etiqueta. Si se define el atributo SIZE sólo, el color del texto que define será el por defecto, si se define con la etiqueta COLOR sólo, el tamaño será el de la fuente base. Al igual que en el caso del tamaño de la fuente pueden ser utilizados junto con otros elementos del lenguaje HTML como listas y formularios y pueden ser utilizados conjuntamente con los elementos de resalte. El cambio de color no afectará al texto o elementos incluidos en un hiperenlace, utilizándose para estos el color por defecto definido en la etiqueta BODY. Por ejemplo, algunos ejemplos de la definición de colores de fuente podrían ser:

Estilos físicos y lógicos

Se define como estilos a los distintos efectos que se pueden aplicar al texto normal. Estos efectos son los resaltes (negrita, cursiva, subrayado, etc.) que se pueden usar. En HTML existen dos grupos principales de estilos los lógicos y los físicos.

Los físicos son aquellos que siempre causan un mismo efecto (negrita, cursiva, etc.) y los lógicos indicarán un tipo de texto (cita, nombre de persona) que por sus características tiene un modo de mostrarse propio. Se pueden utilizar ambos estilos para especificar un mismo efecto, por ejemplo para obtener una frase resaltada podemos usar el estilo físico (negrita) o el estilo lógico (STRONG) y en ambos casos el resultado puede ser el mismo.

Es más recomendado, sin embargo, utilizar el estilo lógico, ya que en éste el modo en que se ven los distintos efectos puede ser definido por el usuario, de forma que se pueda obtener una presentación personalizada. Además en un futuro los navegadores podrían añadir modos más

sofisticados de presentar los distintos estilos y si están definidos de forma lógica la adaptación es más sencilla. Y por último y más importante, al utilizar estilos lógicos el modo de escribir HTML se hace independiente de como se presentará finalmente el texto, se utilizará el estilo según el tipo de texto que sea y no según como se debe presentar.

La tendencia actual de los navegadores es el uso de los estilos físicos olvidando un poco la versatilidad de los estilos lógicos, pero sin embargo ambos estilos pueden ser usados indistintamente, sin ningún problema.

Estilos Físicos → El efecto se aplicará al texto expresado entre la etiquetas de inicio y fin. Los estilos se pueden combinar entre sí obteniendo cualquier efecto deseado.

Estilos lógicos → En este caso se definen las situaciones o tipos de frases y estas tomarán la representación más adecuada a cada caso.

Las listas en HTML

El lenguaje HTML proporciona un modo sencillo de representar elementos en forma de lista. Dentro de una lista de puede incluir cualquiera de los elementos HTML, e igualmente una lista puede incluirse dentro de formularios, tablas, etc.
Existen principalmente tres tipos de listas: las listas no ordenadas, las listas ordenadas y las listas de definiciones.

<UL> Listas no ordenadas

Este tipo de listas se usan para enumerar elementos que no tengan un orden definido. Se especificará con el elemento <UL>, todo lo que se incluya dentro de esta etiqueta y la de cierre formará la lista. Con los elementos <LI> se indicarán cada uno de los componentes de la lista. El formato es el siguiente:

<UL TYPE = DISK ó CIRCLE ó SQUARE >
<LH> *Titulo de la lista* </LH>
<LI> *Elemento 1*
<LI> *Elemento 2*
. . .
<LI> *Elemento n*
</UL>

Para marcar los distintos elementos de la lista se usarán unos símbolos, que pueden ser un disco (DISK), un círculo (CIRCLE) o un cuadrado (SQUARE), seleccionables con el atributo TYPE. Con la etiqueta <LH> se define el titulo de la lista, este es opcional y aparecerá en la parte superior de ésta. Igualmente es posible definir listas de varios niveles, que será listas anidadas, listas dentro de listas. Y combinar este tipo de lista con las que se explicarán en los proximos apartados.

<OL> Listas ordenadas

Estas listas serán similares al caso anterior pero en éste se usa un número para indicar el orden de cada elemento de la lista. El ser ordenada no significa que ordene los elementos alfabéticamente sino que los elementos guardan un orden que es el número que indexa la lista.

<OL START = *n* TYPE = A *ó* a *ó* I *ó* i *ó* 0 >

<LH> *Titulo de la lista* </LH>

<LI> *Elemento 1*

<LI> *Elemento 2*

. . .

<LI> *Elemento n*

</OL>

El punto de comienzo siempre será el 1 si no se indica en START con otro valor de comienzo y el tipo de numeración puede seleccionarse con el atributo TYPE. Sus posibles valores son:

- ✓ A : Letras mayúsculas.
- ✓ a : Letras minúsculas.
- ✓ I : Número romanos en mayúsculas.
- ✓ i : Número romanos en minúsculas.
- ✓ 0 : Números (es por defecto por tanto no hay que indicarlo).

<DL> Listas de definiciones

En esta lista existirán dos elementos la definición y el término, se usará principalmente para listas en las que se incluirán una palabra o frase y su definición (diccionario). El término aparecerá pegado en el borde izquierdo y la definición aparecerá ligeramente tabulada a partir del borde izquierdo. Se puede usar de forma separada la definición y el término, pudiendo por tanto incluir sólo definiciones o sólo términos. El incluir sólo términos podría usarse para sangrar el comienzo de un párrafo.

<DL>

<LH>*Titulo de la lista* </LH>

<DT>*Termino 1*

<DD>*Definición 1*

<DT>*Termino 2*

<DD>*Definición 2*

. . .

<DD>*Termino N*

<DT>*Definición N*

</DL>

El titulo de las lista <LH> como en los casos anteriores es opcional. Es posible anidar cualquier tipo de lista entre sí, incluso listas de distinto tipo.

Hiperenlaces

Es la utilidad básica del hipertexto, permite indicar zonas de texto o imágenes que si son seleccionados por el lector del documento nos traslada a otros documentos HTML o otras zonas del documento actual.

Para definir un hiperenlace se utiliza cualquier elemento HTML, no debe ser texto necesariamente, tambien se pueden usar, cabeceras (<Hn>), cualquiera de los estilos, una

imagen, etc. Un hiperenlace igualmente podrá definirse dentro de cualquier elemento HTML: listas, párrafos de texto, tablas, formularios. El texto del hiperenlace aparece normalmente resaltado sobre el texto normal, en azul y subrayado, en el caso de las imágenes, si tienen definido un borde este aparecerá resaltado en color azul. La mayoría de los navegadores cuando la zona por la que pasa el cursor es sensible, este cambia de aspecto indicándolo y en la parte baja de la pantalla se indica el hiperenlace que se activa al pulsar en esa zona. Mediante los atributos de BODY es posible cambiar el color de los hiperenlaces.

El texto que define el hiperenlace debe ser clarificador sobre el documento al que referenciamos, debemos evitar referencias específicas que hagan al texto poco legible. Igualmente deberá concordar con el texto del párrafo donde se englobe y, se puede evitar el uso de pulse aquí. El usuario que lea el texto ya sabrá al verlo resaltado que puede pulsar ahí.

<A HREF=...> Hiperenlace

Son los enlaces con documentos externos al actual. En este caso se indicará una URL que definirá el documento al que se accede si se sigue el enlace. La forma de indicarlo será:

<A HREF="*URL a la que se accede*"> *Texto del Hiperenlace*</A>

El texto indicado entre las etiquetas de comienzo y de fin se presentará de forma resaltada y en el caso de seleccionar este texto el documento actual cambiará por el especificado en la URL. Igualmente se puede indicar una imagen como enlace, en este caso entre las etiquetas del hiperenlace indicamos la inclusión de la imagen, también dentro de la etiqueta del hiperenlace se pueden incluir cualquiera de las etiquetas del lenguaje HTML para cambiar el aspecto del texto afectado por el hiperenlace. Un ejemplo sería:

<A HREF="*URL a la que se accede*"><IMG SRC="Imagen"> y también texto</A>

En este caso aparecerá la imagen con el borde resaltado para indicar que es un hiperenlace.

Además de enlaces con el servicio http se podrán utilizar cualquiera de los servicios de internet que se puede expresar en una URL. Es posible por tanto indicar enlaces a servidores FTP, servidores GOPHER o de NEWS, indicando como URL aquella que identifica al servidor que se desea acceder. Es importante indicar el nombre completo de la máquina a la que se accede, es decir el nombre y el dominio. Si se indica www, las computadoras de la red local si podrán localizarlo, pero para pcs's externas la referencia podría ser a otra máquina. Por tanto se debe indicar www2.uca.es para asegurar que es accesible independientemente del lugar desde el que se conecte el cliente. Se utilizan los hiperenlaces para dividir de forma conveniente la información. Hay que evitar los documentos excesivamente largos, estos se pueden dividir en distintos documentos a los que acceder por hiperenlaces, a partir de un índice.

URL absolutas y relativas

Para definir la URL, se pueden utilizar direcciones absolutas o relativas. Por direcciones absolutas se entenderá la URL completa, es decir, http://maquina.dominio/camino/fichero.html. En el caso de las relativas todo lo que no se escriba de la URL se tomará de la URL del documento que contiene el hiperenlace, por ejemplo si no se indica el servidor, se considerará el actual y si sólo seindica un fichero html se considerará que se encuentra en el servidor y directorio del documento que lo referencia.

Una URL relativa comenzará siempre por un nombre de directorio o por un fichero, ya que en este caso se asume que el servicio y el servidor (http://máquina.dominio) es el mismo del documento actual. Se puede comenzar de alguna de las siguientes formas:

- ✓ Por una barra /, que indica que el camino del fichero se especifica desde el directorio raiz del servidor.
- ✓ Por dos puntos y una barra ../, significa subir en la estructura del directorio. Se considerará a partir del directorio anterior.
- ✓ Por un nombre de fichero o directorio, considerandose éste a partir del directorio actual.

<A NAME=...> Elemento Ancla

Sirve para indicar puntos de un documento que son accesibles directamente. Marcará las distintas zonas de un documento. La forma de indicarlo es:

<A NAME="*Id. del ancla*">Texto del ancla</A>

A cada ancla se le dará un nombre distinto que será el que se utilice luego para referenciarlo. El texto que define a la etiqueta no tendrá ningún tipo de resalte, y sólo se utiliza como punto de destino del hiperenlace. La forma de especificar un enlace que acceda a un punto determinado del mismo documento es:

<A HREF="#*Id. del ancla*"> *Texto del enlace al ancla*</A>

Será un enlace a la zona del documento actual que se había marcado con la etiqueta indicada. También es posible acceder a un ancla de un documento externo de la forma:

<A HREF="*Dirección URL#Id. del ancla*"> *Texto del enlace al ancla*</A>

De esta forma se podrá acceder a puntos determinados o secciones de un documento de forma directa. La utilidad principal es la creación de índices en documentos largos, al comienzo del documento se especifica el índice con enlaces a las distintas anclas y dentro del documento se indica el comienzo de cada apartado con el ancla que lo define.

Imágenes en los documentos HTML

Una de las características principales del lenguaje HTML y de la WWW es la introducción de elementos multimedia, en este apartado se verá como introducir gráficos y ficheros de imágenes en un documento HTML.

En un documento HTML se puede incluir cualquier imagen en alguno de los siguientes formatos gráficos: GIF, JPEG, PNG, BMP, XBM, etc. El formato más extendido y práctico es el GIF, todos los navegadores representativos de la Web soportan este formato, además existen gran cantidad de programas de tratamiento, grabado o convertidor de imágenes. El formato GIF, utiliza algoritmos de compresión que hacen que sus ficheros sean de corto tamaño y apropiados para su transmisión por la red. El formato GIF es más recomendado para iconos, gráficas, etc., y el formato JPEG es más útil para imágenes reales como paisajes, personas, etc. Para poder utilizar otro formato de imagen, necesitará usar un enlace cuyo destino sea el fichero dicha imagen. Al seguir el enlace se le pedirá que indique un programa externo que se encargue de mostrar los ficheros de ese formato imagen, por lo tanto no pueden insertarse dentro de documentos HTML. Existen casos especiales en las imagenes GIF, que son las imágenes transparentes y animadas.

<IMG SRC=...> Inclusión de Imágenes

La etiqueta encargada de mostrar imágenes en HTML es IMG y tiene el siguiente formato:

<IMG SRC="*URL de la Imagen*" ALT="*Texto alternativo a la imagen*">

En el punto del fichero HTML en el que se quiera que se muestre la imagen se incluye esta etiqueta. Se puede mostrar, al comienzo del documento, al final o intercalada en el texto. Igualmente se puede insertar dentro de cualquiera de las estructuras del lenguaje HTML como listas, tablas o formularios. Esta etiqueta no necesita la etiqueta de fin, ya que el efecto de la etiqueta no su produce sobre un texto o algún elemento HTML.

El atributo ***SRC*** indica el fichero o directorio de la imagen que se incluirá en el documento. Se indicará el la ruta hasta el gráfico en formato URL, el dicrectorio deberá tener una extensión apropiada a su formato, por ejemplo si es GIF la extensión será .gif, si es JPEG la extensión será .jpg o .jpeg, si no se cumple esto, no se mostrará de forma correcta la imagen. Al definir la ubicación del gráfico o imagen como una URL, no es necesario que se encuentre en el servidor local, pudiendo especificar el camino completo y el servidor donde se encuentra la imagen. En este caso al igual que en los hiperenlaces es posible indicar direcciones de URL relativas al documento actual, como se vio en el apartado anterior. Lo normal es referenciar una imagen que se encuentre en el servidor local, ya que el acceso a imágenes en servidores externos puede ser más lento. Por tanto conviene copiar las imágenes e iconos que se usen al servidor local. A continuación se explica la utilidad de cada unos de los atributos de la etiqueta IMG.

<IMG ... ALT=...> Texto alternativo

El atributo ALT, indicará un texto alternativo a mostrar si no fue posible mostrar el gráfico. Se usa para navegadores que no permitan mostrar imágenes, sean sólo texto o tenga inhabilitado el mostrarlas. Se recomienda cuando existan en el documento elementos visuales usados como botones, para mostrar un texto en vez del botón en navegadores que no puedan mostrar gráficos, de esta forma se consigue que todos los usuarios puedan consultar sus páginas.

<IMG .. ALIGN=...> Alineación de la imagen

Indica como se alinea el texto que sigue al imagen con respecto a esta. Indicará si la primera frase del texto se colocará en la parte alta del dibujo, TOP, en el punto medio de la imagen, MIDDLE, o en la parte de abajo de la ilustración, BOTTOM.

También se pueden utilizar alineaciones un poco más avanzada, TEXTTOP se alinea justo al comienzo del texto más alto de la línea (TOP se alinea al tamaño del primer carácter de la línea). ABSMIDDLE, es el centro real de la imagen, con MIDDLE se coloca el texto a partir del punto medio, con ABSMIDDLE el texto aparece centrado con la imagen. ABSBOTTOM coloca el texto justo al final de la imagen. Se recomienda que se usen estos últimos al ser más precisa la alineación, aunque sólo son válidos para los navegadores más avanzados.

<IMG .. BODER=...> Borde de la imagen

Indicará el tamaño del borde de la imagen, si la imagen se encuentra dentro de un hiperenlace el borde de ésta será del color apropiado para indicarlo, en caso contrario el borde será invisible. Si se desea que no se muestre el borde cuando la imagen sea un enlace se usará BORDER=0.

<IMG .. HEIGHT=... WIDTH=...> Tamaño de la imagen

Es posible cambiar el tamaño, de forma que pueda ajustarse como se desee, pudiendo ampliar o disminuir éste. El atributo HEIGHT Determina el alto de la imagen a mostrar, se especifica en puntos de la pantalla (pixeles) o en tanto por ciento sobre el tamaño del documento. En caso de que la imagen sea mayor o menor se escalará a este tamaño. El atributo WIDTH indica el ancho al que se mostrará la imagen, escalándola a este tamaño. También se expresará en pixeles o en tanto por ciento.

No es necesario indicar el ancho y el alto, se puede especificar sólo uno de las dimensiones, ajustándose la otra a la proporción de la imagen. Es recomendable indicar sólo uno de estos parámetros para que la imágen no se muestre deformada. Igualmente se debe usar el valor relativo en tanto por ciento si se desea que la imágen guarde siempre una misma proporción con respecto al texto. Si se desea mostrar dos imágenes y queremos que una ocupe un cuarto de la pantalla y la otra el espacio restante, se indica en una el ancho (WIDTH) del 25% y en la otra del 74%, no indicando en ningún caso el alto (HEIGHT), de esta forma independientemente como sea el tamaño de la ventana del navegador e independientemente del monitor del cliente, siempre se podrán mostrar ambas imágenes en la misma línea.

TABLAS

Permiten la representación de datos por filas y columnas, en forma tabular. La definición es muy flexible indicando sólo los elementos que forman cada fila y columna, calculándose de forma automática el tamaño que deben tener las celdas. En una tabla se pueden introducir todo tipo de elementos del lenguaje HTML como imágenes, enlaces, texto, listas, cabeceras, formularios, etc.
No es necesario definir inicialmente el número de filas o columnas ya que estas se calculan según se va definiendo la tabla. En el caso que una fila tenga más columnas que otra, en las otras filas las columnas se representarán vacias, no siendo necesario que todas las filas sean iguales.

<TABLE> Definición de la Tabla

Para definir una tabla se usa la etiqueta <TABLE>, que tiene el siguiente formato:

<TABLE BORDER="*tamaño del borde*" >

... Definición de la tabla ...

</TABLE>

En la etiqueta inicial TABLE se definen los atributos que afectarán a toda la tabla, todos los atributos son opcionales. Todo lo que se incluya dentro de la instrucción de tabla se considerará como tal, pudiendo definir tablas anidadas (tablas dentro de tablas). Algunos de sus atributos se muestran en el Anexo 3.

Dentro de la instrucción de la tabla se incluirán las diversas etiquetas que defininen el contenido de la tabla.

<CAPTION> Titulo de la tabla

Especifica el titulo de la tabla, este se mostrará resaltado en la parte superior externa de la tabla. Siempre se mostrará centrado horizontalmente.

<CAPTION ALIGN=TOP|BOTTOM>Titulo de la tabla</CAPTION>

Con el atributo ALIGN indicaremos si el titulo debe aparecer arriba (TOP) o debajo (BOTTOM) de la tabla.

<TR> Fila de la tabla

Definirá cada una de las filas de la tabla y especificará los parámetros que afectarán a todas las celdas de la fila. Por cada elemento TR que se incluya se creará una fila de la tabla. No es necesario indicar la etiqueta de cierre </TR>. En caso de tablas anidadas será necesario incluir la etiqueta de cierre. Los atributos se muestran en el Anexo 3.

<TH> y <TD> Una celda de la tabla

Define cada una de las celdas de una fila de la tabla, **TH** se usará **para definir una celda de tipo cabecera**, en este caso se mostrarán destacados en negrita y **TD** para **definir una celda de datos**. Estos elementos deben aparecer tras los elementos TR para definir cada una de las columnas de la fila. Existirá una columna por cada elemento TD ó TH que se defina. Aunque se puede indicar la etiqueta de cierre, no es necesario al tomarse implicitamente. Se utilizan los elementos TH para los titulos de las filas o columnas y los elementos TD para los datos.

FORMULARIOS (FORM)

Los formularios son plantillas que permiten la creación de documentos HTML con peticiones de datos. La principal utilidad de los formularios es la posibilidad de crear cuestionarios, encuestas, páginas de comentarios o cualquier documento en la que se desee una interacción por parte del usuario. Se podrán definir distintos tipos de recuadros de diálogo, botones de selección, menús de múltiples opciones, etc. Para permitir obtener los datos de una manera más intuitiva.

<FORM> Definición de formularios

Existe una instrucción HTML para la creación de formularios esta es FORM, que tiene la siguiente estructura:

< FORM ACTION="fichero que trata el formulario" METHOD= POST | GET >

...

Elementos que forman el formulario

...

< /FORM>

Dentro de la etiqueta de formulario se definirán los distintos elementos de petición de datos. Estas instrucciones que se explicarán a continuación definirán los tipos de botones, cajas de diálogo y ventanas para la introducción de datos. Y definirán las variables que almacenarán los datos introducidos por el usuario. Estas etiquetas se incluirán entre la de definición del formulario y la etiqueta de final de formulario. Los atributos que presenta la etiqueta FORM son los siguientes:

- ✓ **ACTION** → Indica el programa que se encargará de tratar los datos del formulario. Este programa debe encontrarse en el servidor y estar escrito en algún lenguaje de programación. A este programa se pasará como parámetros los datos introducidos en el formulario y retornará un código HTML que se mostrará tras procesar el formulario. A este tipo de programas se les llama cgi-bin, y se explican en el último apartado de este manual: (CGI-BIN).
- ✓ **METHOD** → Indica el protocolo usado para el envío de los datos. Con POST envía los datos en la entrada estándar del programa que trata el formulario y con GET los datos se pasan por parámetro, en la línea de comandos, al programa. El usar uno o otro método vendrá determinado por como son tratados los parámetros en el formulario en el (CGI-BIN). El método de uso más normal será POST.

Una vez definidas las características globales del formulario incluiremos los distintos botones y cajas de diálogo que lo constituyen. Dentro de la instrucción del formulario podrán incluirse cualquier texto o instrucción HTML, siendo recomendado a fin de poder etiquetar las opciones de entrada y especificar cualquier dato importante relacionado con el formulario. Igualmente un formulario puede ser incluido en algunas instrucciones HTML como las listas, tablas, etc.

<INPUT> Entrada básica de datos

La etiqueta INPUT se utiliza para definir gran variedad de tipos de campos de entrada de datos. Por lo general serán entradas de texto corto (a lo sumo una frase) o opciones. El formato básico es el siguiente:

< INPUT TYPE = (TEXT | PASSWORD | CHECKBOX | RADIO | HIDDEN | SUBMIT | IMAGE | RESET) NAME = "*Variable que toma el valor*" VALUE = "*Valor de Inicialización*" >

El atributo TYPE se usa para determinar el tipo de recuadro de diálogo de entrada que se está definiendo, a continuación se explicarán por separado cada una de las opciones. El atributo NAME especifica el nombre de la variable que se define. Este nombre será pasado al programa que trata el formulario junto con el valor que le asigno el usuario del formulario. El atributo VALUE suele especificar el valor de inicialización, que será el valor por defecto.

En el Anexo 3 se describen los distintos tipos de instrucciones de entrada.

<TEXTAREA> Texto en múltiples líneas

Permite la introducción de un texto que puede abarcar varias líneas, introduciendo éste en forma de párrafo. El formato general será:

<TEXTAREA NAME="*variable*" ROWS=*Filas* COLS=*Columnas*>
Texto de Inicializaci& oacute;n que puede
incluir varias líneas.
</TEXTAREA>

En este caso se presenta una ventana del tamaño especificado con los atributos ROWS, filas, y COLS, columnas. El texto expresado entre la etiqueta de inicio y de final sirve para indicar que texto aparecerá inicialmente en la ventana, en este texto se podrán incluir las marcas del lenguaje HTML, pero sólo se tendrán en cuenta para incluir acentos y otro tipo de efectos. Los atributos ROWS y COLS determinan el tamaño de la ventana visible, el texto se podrá extender más allá de estos límites.

<SELECT> Elección entre múltiples opciones

Se usa para menús simples o múltiples. Define menús de tipo pop-up (menú de barras) y ofrece una alternativa más compacta al uso de botones RADIO o CHECKBOX. Su formato es el siguiente:

<SELECT NAME="*variable*">
< OPTION SELECTED VALUE=*valor1*> *Opción Primera*
< OPTION VALUE=*valor2*> *Opción Segunda*
. . .
< OPTION VALUE=*valorn*> *Opción Enésima*
</SELECT>

Si se desea que sea un menú múltiple se deberá incluir el atributo MULTIPLE en la etiqueta de SELECT, en este caso se mostrarán todas las opciones en forma de tabla, en el otro caso se mostrará la opción activa y un botón para poder modificar esta opción. En ambos casos sólo podrá seleccionarse una de las opciones. Cuando se envía el resultado del formulario la variable NAME tomará el valor de la opción que esté activa. La etiqueta OPTION que contenga el atributo SELECTED será considerada la opción por defecto, caso de no especificarse ninguna se considerará la primera de las opciones.

Documentos con Frames

Con las frames es posible dividir la ventana del navegador en varias subregiones (*frames*), permitiendo mostrar una URL distinta en cada una de ellas. En cada frame se nos permite:

- ✓ Mostar su propia URL, diferenciada del resto de las frames de la pantalla, de esta forma un hiperenlace puede tener como destino un documento y la frame en el que se mostrará.
- ✓ Tendrán asociado un nombre, que las distinguirán del resto de las frames de la pantalla y permitirá usarlo en los hiperenlaces.
- ✓ En el caso que se cambie el tamaño de la ventana, se podrá determinar si la frame se ajusta a este tamaño o mantiene su tamaño intacto.

Esto permite crear nuevos tipos de documentos, en los que por ejemplo se mantendrá una región estática (lista de enlaces, barra de botones, formulario) y otra zona dinámica en la que se mostrará el resultado. De esta forma una de las principales utilidades de las frames es la creación de páginas con un índice (por ejemplo un manual) o una cabecera estática, consiguiendo así una mejora de la navegación al poder acceder al índice de una manera más rapida y efectiva.

El uso de las frames es útil para cierto tipo de documentos, pero puede llegar a dificultar la navegación, dentro de un documento con frames no tendrá utilidad los botones de documento previo (back) ni documento siguiente (forward), ya que ambos trasladarán fuera del documento con frames. Para ver el documento previo en una de las frames se debe utilizar el botón derecho del ratón sobre ella y seleccionar la opción volver en el frame (Back in Frame). Esto hace que cuando se utilicen frames haya que cuidar la correcta transición entre documentos.

No todos los navegadores pueden mostrar documentos con frames, sólo son interpretables por el Netscape 2.0 o superior y el Internet Explorer 3.0. Por tanto en este caso si se quiere que el documento sea accesible por gran cantidad de usuarios de debe crear un documento con frames y otro sin ellas, usando la etiqueta NOFRAMES.

Un documento con frames se define de manera diferente a un documento normal, siendo la estructura del documento distinta, en este caso no se define la etiqueta BODY. Su estructura es la siguiente:

```
<HTML>
<HEAD>
Definiciones de la cabecera
</HEAD>
<FRAMESET>
Definición de las frames que forman el documento y de los fichero que incluye cada una.
</FRAMESET>
<NOFRAMES>
```

Instrucciones HTML que se mostrará en los navegadores que no soporten frames.

</NOFRAMES>

</HTML>

Dentro de la etiqueta NOFRAMES se podrá incluir una explicación de que el documento sólo es visible con navegadores que soporten frames, o bien incluir una versión del documento que se muestre sin necesidad de frames.

<FRAMESET ...> Definición de las frames

Con esta instrucción se definen los frames que formarán el documento, su sintaxis es similar a la de las tablas, permitiendo definir muy distintos tipos de frames. Su formato es el siguiente:

<FRAMESET ROWS=*Lista de las Filas* COLS=*Lista de las Columnas*>

<FRAME SRC=*URL_1* NAME="*Nombre de la frame1*">

<FRAME SRC=*URL_2* NAME="*Nombre de la frame1*">

. . .

<FRAME SRC=*URL_N* NAME="*Nombre de la frameN*">

</FRAMESET>

Se definirá sólo uno de los atributos, o la lista de filas (ROWS) o la lista de columnas (COLS).

ROWS → Se definirá separado por comas el tamaño de cada una de las frames. De esta forma se dividirá la pantalla de forma horizontal, según cada una de las filas definidas. El tamaño de la frame, puede expresarse de las siguientes formas:

- ✓ En valor absoluto, que indicará el tamaño en puntos de la pantalla. En este caso si todas las frames se indican de este modo, los valores se ajustarán para que las frames ocupen la totalidad del espacio de la ventana del navegador, no guardando siempre la proporción con la que se definen las frames.
- ✓ En tanto por ciento sobre el tamaño de la ventana, en este caso si los porcentajes suman un valor distinto del 100%, se ajustarán para que coincidan con el tamaño de la ventana. Se podrá combinar con el apartado anterior de forma que algunas frames se definan en valor absoluto y otras en porcentaje.
- ✓ De forma relativa con el símbolo * que indica el tamaño restante de la ventana. Si se indica una frame como 2* y otra como *, la primera ocupará dos tercios del espacio restante y la segunda un tercio del espacio restante. Se puede combinar con las definiciones anteriores.

COLS → Toma los mismos posibles valores que ROWS, pero en este caso para las columnas, se definirán las frames de forma vertical.

Una vez definida el número de frames por fila o por columna se definirá el contenido de cada una de éstas con la etiqueta FRAME, pero igualmente se podría definir frames dentro de frames, de forma que cada una de las definidas anteriormente podría estar divida en varias frames, esto se hará incluyendo dentro de la instrucción FRAMESET, nuevas instrucciones FRAMESET que dividirán ésta en los frames indicados.

<FRAME ...> Definición de cada una de las frames

Como se ha visto en el apartado anterior con la etiqueta FRAME se define el documento que se mostrará en la frame y su nombre. Esta etiqueta presenta además algunos otros atributos:

<FRAME SRC="*URL*" NAME="*Nombre de la frame*" MARGINWIDTH="*ancho del margen*" MARGINHEIGHT="*alto del margen*" SCROLLING=YES | NO | AUTO NORESIZE>

Los elementos y atributos contenidos dentro de la etiqueta Frame se muestran en el Anexo 3.

TARGET

Es un atributo para indicar la frame de destino. El uso de frames introduce un nuevo atributo a alguna de las etiquetas especificadas con anterioridad, este atributo es TARGET que indicará la frame de destino de la operación. Normalmente, en páginas sin frames, cuando se seguía un hiperenlaces este se mostraba en la ventana del navegador sustituyendo el documento actual, con las frames se puede especificar que frame será la de destino, no siendo necesario que sea la frame del documento actual.

Como nombre del frame se usará el nombre que se especifico en el atributo NAME de la etiqueta FRAME. Estas instrucciones se utilizarán normalmente en los documentos que se incluyen dentro de las frames.

En el Anexo 3 se enuncian las etiquetas que permiten el uso de TARGET

Al comprender la estructura de un documento HTML, el uso y manejo de las etiquetas comprendidas dentro de él, se tiene la posibilidad de crear, organizar y modificar la estructura de un sitio Web, adaptandolo a las necesidades que se requieran en base al tema para el que se desarrolle dicho sitio Web. Así el Web Master se da a la tarea de hacer un sistio que agrade y sea útil para lso usuarios a los que esta dirigido dicho sitio.

CAPITULO VI

PROGRAMACIÓN CON PHP

PHP (acrónimo de Hypertext Preprocessor) es uno de los lenguajes de "código abierto" interpretado, de alto nivel, embebido en páginas HTML y ejecutado de lado servidor más extendidos en la web. Nacido en 1994, se trata de un lenguaje de creación relativamente reciente que ha tenido una gran aceptación en la comunidad de webmasters debido sobre todo a la potencia y simplicidad que lo caracterizan. PHP permite incluir su código dentro de una página HTML y realizar determinadas acciones de una forma fácil y eficaz sin tener que generar programas elaborados íntegramente en un lenguaje distinto al HTML. Por otra parte, y es aquí donde reside su mayor interés con respecto a los lenguajes pensados para los CGI, PHP ofrece un sinfín de funciones para la explotación de bases de datos de una manera llana, sin complicaciones.

PHP, aunque multiplataforma, ha sido concebido inicialmente para entornos UNIX y es en este sistema operativo donde se pueden aprovechar mejor sus prestaciones. Las tareas fundamentales que puede realizar directamente el lenguaje son definidas en PHP como funciones. Presenta una filosofía totalmente diferente y, con un espíritu más generoso, es progresivamente construido por colaboradores desinteresados que implementan nuevas funciones en nuevas versiones del lenguaje.

Alcances de PHP

Existe una multitud de lenguajes concebidos o no para Internet. Cada uno de ellos explota más a fondo ciertas características que lo hacen más o menos útiles para desarrollar distintas aplicaciones. La versatilidad de un lenguaje está íntimamente relacionada con su complejidad. Un lenguaje complicado en su aprendizaje permite en general el realizar un espectro de tareas más amplio y más profundamente. En el dominio de la red, los lenguajes de lado servidor más ampliamente utilizados para el desarrollo de páginas dinámicas son el ASP, PHP y PERL. PHP resulta bastante útil para la explotación de bases de datos y su aprendizaje resulta accesible para una persona profana de la programación. Es un lenguaje que resulta una muy buena opción a la hora de hacer evolucionar un sitio web realizado en HTML.

Herramientas relacionadas con el desarrollo

Para escribir una página dinámica se puede hacer del mismo modo que si se hiciera en HTML. En realidad, el código está constituido exclusivamente de texto y lo único que se tiene que hacer por lo tanto es guardar el archivo texto con una extensión que pueda ser reconocida posteriormente por el servidor. Así, por ejemplo, las páginas de PHP son reconocidas por su extensión "php" u otras en las que se especifica la versión utilizada ("php3" o "php4"). En muchos casos el servidor permite seleccionar qué tipo de extensión debe ser reconocida para un determinado lenguaje por lo que estas extensiones no están totalmente generalizadas aunque son sin duda las más utilizadas.

6.1 SINTAXIS BÁSICA

PHP se escribe dentro de la propia página web, junto con el código HTML y, como para cualquier otro tipo de lenguaje incluido en un código HTML, en PHP se necesita especificar cuáles son las partes constitutivas del código escritas en este lenguaje. Esto se hace, como en otros casos, delimitando el código por etiquetas. Se pueden utilizar distintos modelos de etiquetas en función de las preferencias y costumbres. Hay que tener sin embargo en cuenta que no necesariamente todas están configuradas inicialmente y que otras sólo están disponibles a partir de una determinada versión (3.0.4. y posterior). Estos modos de abrir y cerrar las etiquetas son:

```
<?          y          ?>
<%          y          %>
<?php       y          ?>
<script languaje="php">
```

Este último modo está principalmente aconsejado para aquellos que trabajen con *Front Page* ya que, usando cualquier otro tipo de etiqueta, se corre el riesgo de que la aplicación se borre sin más, debido a que se trata de un código incomprensible para ésta.

El modo de funcionamiento de una página PHP, a grandes rasgos, no difiere del clásico para una página dinámica de lado servidor: El servidor reconoce la extensión correspondiente a la página PHP (*phtml, php, php4*, etc.) y antes de enviarla al navegador se encarga de interpretar y ejecutar todo aquello que se encuentre entre las etiquetas correspondientes al lenguaje PHP. El resto, lo enviara sin más ya que, asumirá que se trata de código HTML absolutamente comprensible por el navegador. Otra característica general de los scripts en PHP es la forma de separar las distintas instrucciones. Para hacerlo, hay que acabar cada instrucción con un punto y coma ";". Para la última expresión, la que va antes del cierre de etiqueta, este formalismo no es necesario.

6.2 Variables

Una variable consiste en un elemento al cual se le da un nombre y se le atribuye un determinado tipo de información. Las variables pueden ser consideradas como la base de la programación.

Para PHP, las variables se definen anteponiendo el símbolo pesos ($) al nombre de la variable que se están definiendo. Por ejemplo:

```
$variable
```

TIPOS DE DATOS y expresiones

Dependiendo de la información que contenga, una variable puede ser considerada de uno u otro tipo:

Variables numéricas Almacenan cifras		
Enteros	**Números sin decimales**	**$entero = 2002;**
Real	**Números con o sin decimal**	**$real = 3.14159;**

Variables alfanuméricas Almacenan textos compuestos de números y/o cifras		
Cadenas	**Almacenan variables alfanuméricas**	**$cadena = "Hola amigo";**

Arreglos Almacenan series de informaciones numéricas y/o alfanuméricas		
Arrays	**Son las variables que guardan varios valores**	**$sentido[1]="ver"; $sentido[2]="tocar"; $sentido[3]="oír"; $sentido[4]="gusto"; $sentido[5]="oler";**

Objetos
Se trata de conjuntos de variables y funciones asociadas. Presentan una complejidad mayor que las variables anteriores.

A diferencia de otros lenguajes, PHP posee una gran flexibilidad a la hora de operar con variables. En efecto, cuando se define una variable asignándole un valor, el compilador le atribuye un tipo. Si por ejemplo se define una variable entre comillas, la variable será considerada de tipo cadena:

```
$variable = "5";    //esto es una cadena
```

Sin embargo, si se pide en el script realizar una operación matemática con esta variable, no se mostrará un mensaje de error, sino que la variable cadena será convertida a numérica:

```
<?php
$cadena = "5"; //esto es una cadena
$entero = 3; //esto es un entero
echo $cadena + $entero
?>
```

Este script dará como resultado "8". La variable *cadena* ha sido asimilada en entero (*aunque su tipo sigue siendo cadena*) para poder realizar la operación matemática. Del mismo modo, se puede operar entre variables tipo entero y real. No hay que preocuparse de nada, PHP se encarga durante la ejecución de interpretar el tipo de variable necesario para el buen funcionamiento del programa. Sin embargo, en contraste, hay que tener cuidado en no cambiar mayúsculas por minúsculas ya que, en este sentido, PHP es de tipo *case sentive*. Conviene por lo tanto trabajar ya sea siempre en mayúsculas o siempre en minúsculas para evitar este tipo de malentendidos a veces muy difíciles de localizar.

6.3 OPERADORES

Las variables, como base de información de un lenguaje, pueden ser creadas, modificadas y comparadas con otras por medio de los llamados operadores. En los apartados anteriores se han utilizado en algunos de ellos. A continuación, se pretende listar los más importantes y así dar constancia de ellos para futuros ejemplos.

Operadores aritméticos

Permiten realizar operaciones numéricas con las variables:

+	**Suma**
-	**Resta**
*****	**Multiplicación**
/	**División**
%	**Devuelve el residuo de la división**

Operadores de comparación

Se utilizan principalmente en condiciones para comparar dos variables y verificar si cumple o no la propiedad del operador.

==	Igualdad	<=	Menor igual que
!=	Desigual	>	Mayor que
<	Menor que	>=	Mayor igual que

Operadores lógicos

Se usan en combinación con los operadores de comparación cuando la expresión de la condición lo requiere.

And	Y	Or	O	!	No

Operadores de Incremento

Sirven para aumentar o disminuir de una unidad el valor de una variable

++$variable	Aumenta de 1 el valor de $variable
$variable++	Aumenta de 1 el valor de $variable
--$variable	Reduce de uno el valor de $variable
$variable--	Reduce de uno el valor de $variable

Operadores combinados

Una forma habitual de modificar el valor de las variables es mediante los operadores combinados:

$variable += 10	Suma 10 a $variable
$variable -= 10	Resta 10 a $variable
$variable .= "añado"	Concatena las cadenas $variable y "añado"

Este tipo de expresiones no son más que abreviaciones de otras formas más clásicas:

```
$variable += 10
```

es lo mismo que:

```
$variable = $variable+10
```

6.4 BLOQUES Y SENTENCIAS

La programación exige en muchas ocasiones la repetición de acciones sucesivas o la elección de una determinada secuencia y no de otra dependiendo de las condiciones específicas de la ejecución. Como ejemplo, se puede hacer alusión a un script que ejecute una secuencia diferente en función del día de la semana en el que nos encontramos. Este tipo de acciones pueden ser llevadas a cabo gracias a una paleta de instrucciones presentes en la mayoría de los

lenguajes. A continuación se describirán someramente algunas de ellas propuestas por PHP y que resultan de evidente utilidad.

La condicional *if*

Cuando se quiere que el programa, llegado a un cierto punto, tome un camino concreto en determinados casos y otro diferente si las condiciones de ejecución difieren, se sirve del conjunto de instrucciones if, else y elseif. La estructura de base de este tipo de instrucciones es la siguiente:

```
if (condición)
{
  Instrucción 1;
  Instrucción 2;
  ...
}
else
{
  Instrucción A;
```

Llegado este punto, el programa verificará el cumplimiento o no de la condición. Si la condición es cierta las instrucciones 1 y 2 serán ejecutadas. De lo contrario (*else*), las instrucciones A y B serán llevadas a cabo. Esta estructura de base puede complicarse un poco más si se tiene en cuenta que no necesariamente todo es blanco o negro y que muchas posibilidades pueden darse. Es por ello que otras condiciones pueden plantearse dentro de la condición principal. Se habla por lo tanto de condiciones anidadas que tendrían una estructura del siguiente tipo:

```
if (condición1)
{
  Instrucción 1;
  Instrucción 2;
  ...
}
else
{
  if (condición2)
  {
    Instrucción A;
    Instrucción B;
    ...
  }
  else
```

De este modo se introducen tantas condiciones como se quieran dentro de una condición principal. También se puede usar la instrucción elseif que permite en una sola línea introducir una condición adicional. Este tipo de instrucción simplifica ligeramente la sintaxis que se acaba de ver:

```
if (condición1)
{
  Instrucción 1;
  Instrucción 2;
  ...
}
elseif (condición2)
{
  Instrucción A;
  Instrucción B;
  ...
}
else
```

El uso de esta herramienta resultará claro con un poco de práctica. Pongase un ejemplo sencillo de utilización de condiciones. El siguiente programa permitiría detectar el idioma empleado por el navegador y visualizar un mensaje en dicho idioma.

6.5 Ciclos

Las computadoras, como cualquier máquina, están diseñadas para realizar tareas repetitivas. Es por ello que los programas pueden aprovecharse de este principio para realizar una determinada secuencia de instrucciones un cierto número de veces. Para ello, se utilizan las estructuras llamadas ciclos que ayudan, usando unas pocas líneas, a realizar una tarea incluida dentro del ciclo un cierto número de veces definido por nosotros mismos.

Ciclo while

Sin duda el ciclo más utilizado y el más sencillo. Se usa para ejecutar las instrucciones contenidas en su interior siempre y cuando la condición definida sea verdadera. La estructura sintáctica es la siguiente:

```
while (condición)
{
  instruccion1;
  instruccion2;
  ...
}
```

Ciclo do/while

Este tipo de ciclo no difiere en exceso del anterior. La sintáxis es la siguiente:

```
do
{
  instruccion1;
  instruccion2;
  ...
}
while (condición)
```

La diferencia con respecto a los ciclos *while* es que este tipo de ciclo evalúa la condición al final con lo que, incluso siendo falsa desde el principio, éste se ejecuta al menos una vez.

Ciclo for

PHP está provisto de otros tipos de estructuras de control que también resultan muy prácticos en determinadas situaciones. El más popular de ellos es el ciclo *for* que, como para los casos anteriores, se encarga de ejecutar las instrucciones entre llaves. La diferencia con los anteriores radica en cómo se plantea la condición de finalización del ciclo. Para aclarar su funcionamiento vamos a expresar el ejemplo de ciclo while visto anteriormente en forma de ciclo for:

```
<?php
for ($size=1;$size<=6;$size++)
{
  print "<font size=$size>Tamaño $size</font><br>\n";
}
?>
```

Las expresiones dentro del paréntesis definen respectivamente:

- ✓ Inicialización de la variable. Válida para la primera vuelta del ciclo.
- ✓ Condición de evaluación a cada vuelta. Si es cierta, el ciclo continúa.
- ✓ Acción a realizar al final de cada vuelta de ciclo.

Ciclo foreach

Este ciclo, implementado a partir de la versión de PHP4, ayuda a recorrer los valores de un *array* lo cual puede resultar muy útil por ejemplo para efectuar una lectura rápida del mismo. Hay que recordar que un *array* es una variable que guarda un conjunto de elementos (valores) catalogados por claves. La estructura general es la siguiente:

```
foreach ($array as $clave=>$valor)
{
  instruccion1;
  instruccion2;
  …;
}
```

break y **continue**

Estas dos instrucciones se introducen dentro de la estructura y sirven respectivamente para escapar del ciclo o saltar a la iteración siguiente. Pueden resultar muy prácticas en algunas situaciones.

6.6 FUNCIONES

Una función puede 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 se realiza la llamada de la forma apropiada y se especifican 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, se pueden crear funciones propias para conectarse a una base de datos o crear los encabezados o etiquetas meta de un documento HTML. Para una aplicación de comercio electrónico se pueden crear por ejemplo funciones de cambio de una moneda a otra o de cálculo de los impuestos para añadir al precio del artículo. En definitiva, es interesante crear funciones para la mayoría de acciones más o menos sistemáticas que realizan en los programas.

Aquí se ve un ejemplo de creación de una función que, llamada al comienzo de un script, crea el encabezado del documento HTML y coloca el titulo que se quiere a la página:

```
<?php
function hacer_encabezado($titulo)
{
$encabezado="<html>\n<head>\n\t<title>$titulo</title>\n</head>\n";
print $encabezado;
}
?>
```

Esta función podría ser llamada al principio de todas las páginas de la siguiente forma:

```
$titulo="Mi web";
hacer_encabezado($titulo);
```

De esta forma se automatiza el proceso de creación del documento. Se puede, por ejemplo, incluir en la función otras variables que ayudasen a construir la etiqueta meta y de esta forma, con un esfuerzo mínimo, se crearían los encabezados personalizados para cada una de las páginas. De este mismo modo nos es posible crear cierres de documento o formatos diversos para los 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 se h.a creado personalmente. Esto en realidad no pone ninguna regla ya que puede ser incluida desde un archivo en el que se irán almacenando las definiciones de las funciones que se vayan creando o recopilando. Estos archivos en los que se guardan las funciones se llaman *librerías*. La forma de incluirlos en el script es a partir de la instrucción *require* o *include*:

```
require("libreria.php") o include("libreria.php")
```

Quedaria de la siguiente manera, se tendría un archivo libreria.php como sigue:

```
<?php
//función de encabezado y colocación del titulo
function hacer_encabezado($titulo)
{
$encabezado="<html>\n<head>\n\t<title>$titulo</title>\n</head>\n";
print $encabezado;
}
?>
```

Por otra parte, se tiene el script principal de la página.php (por ejemplo) con el siguiente código:

```
<?
include("libreria.php");
$titulo="Mi Web";
hacer_encabezado($titulo);
?>
<body>
El cuerpo de la página
</body>
</html>
```

Se pueden introducir todas las funciones que se vayan 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 ayudará a poder localizarlas antes para corregirlas o modificarlas, permiten también cargar únicamente el tipo de función que necesitamos para el script sin recargar éste en exceso además de permitir 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 permitirá al programador reconocerlas enseguida cuando se lea el script sin tener que recurrir a la memoria para descubrir su utilidad.

No obstante, antes de empezar a crear una función propia, merece la pena darle una checada a la documentación de PHP[1] para ver si dicha función ya existe o se puede aprovechar de alguna de las existentes para aligerar el trabajo. Así, por ejemplo, existe una función llamada *header* que crea un encabezado HTML configurable lo cual evita tener que crearla uno mismo.

6.7 CONSTANTES

Las constantes en PHP tienen que ser definidas por la función *define()* y además no pueden ser redefinidas con otro valor.

Además, existen una serie de variables predefinidas denominadas:

FILE :	**Archivo que se está procesando.**
LINE :	**Línea del archivo que se está procesando**
_PHP_VERSION :	**Versión de PHP.**
PHP_OS :	**Sistema operativo del cliente.**
TRUE :	**Verdadero.**
FALSE :	**Falso.**
E_ERROR :	**Error sin recuperación.**
E_WARNING :	**Error recuperable.**
E_PARSE :	**Error no recuperable (sintaxis).**
E_NOTICE :	**Puede Tratarse de un error o no. Normalmente permite continuar la ejecución.**

Nota: Todas las constantes que empiezan por "E_"se utilizan normalmente con la función error_reporting().

Por ejemplo:

```
<?php
define("CONSTANTE", "hello world.");
print CONSTANTE;
?>
```

6.8 COMENTARIOS

Se pueden incluir comentarios como en todo lenguaje de programación. Un comentario es una frase o palabra que se incluye en el código para comprenderlo más fácilmente al volverlo a leer un tiempo después y que el compilador ignora ya que no va dirigido a él sino a uno mismo. Los comentarios tienen una gran utilidad ya que es muy fácil olvidarse del funcionamiento de un script

[1] La documentación de funciones de PHP la puede encontrar en la siguiente URL http://www.php.net/manual/es/funcref.php

programado tiempo atrás y resulta muy útil si se quiere hacer rápidamente comprensible el código a otra persona.

La forma de incluir estos comentarios es variable dependiendo si se quiere escribir una línea o más. Si se usa doble barra (//) o el símbolo # se pueden introducir comentarios de una línea. Mediante /* y */ se crean comentarios multilínea. Por supuesto, nada impide usar estos últimos en una sola línea. De paso nos se dará uno cuenta de que las variables en PHP se definen anteponiendo un símbolo de pesos ($) y que la instrucción *print* sirve para sacar en pantalla lo que hay escrito a continuación. Todo el texto insertado en forma de comentario es completamente ignorado por el servidor. Resulta importante acostumbrarse a dejar comentarios, es algo hace fácil el mantenimiento del sitio con el tiempo.

6.9 TRABAJANDO CON ARREGLOS

Otro tipo de variable con que cuenta PHP son los arrays. Un array es una variable que está compuesta de varios elementos cada uno de ellos catalogado dentro de ella misma por medio de una clave.

Una forma muy práctica de almacenar datos es mediante la creación de arrays multidimensionales (tablas). Supóngase el ejemplo siguiente: Se quieren almacenar dentro de una misma tabla el nombre, moneda e idioma hablado en cada país. Para hacerlo se puede emplear un *array* llamado *país* que vendrá definido por estas tres características (*claves*). Para crearlo, se debe escribir una expresión del mismo tipo que la vista anteriormente, en la que meteremos un array dentro del otro. Este *proceso de incluir una instrucción dentro de otra* se llama *anidar* y es muy común en programación:

```
<?php
$pais=array
(
"espana" =>array
  (
  "nombre"=>"España",
  "idioma"=>"Español",
  "moneda"=>"Peseta"
  ),
"francia" =>array
  (
  "nombre"=>"Francia",
  "idioma"=>"Francés",
  "moneda"=>"Franco"
  )
);
print $pais["espana"]["moneda"] //Saca en pantalla: "Peseta"
?>
```

Antes de entrar en el detalle de este pequeño script, se comentarán algunos puntos referentes a la sintaxis. Como puede verse, en esta secuencia de script, no se ha introducido punto y coma ";" al final de cada línea. Esto es simplemente debido a que lo que se ha escrito puede ser considerado como una sola instrucción. En realidad, es uno quien decide cortarla en varias líneas para, así, facilitar su lectura. La verdadera instrucción acabaría una vez definido completamente el array y es precisamente ahí donde se han colocado el único punto y coma. Por otra parte, se observar como se ha jugado con el tabulador para separar unas líneas más que otras del principio. Esto también se hace por cuestiones de claridad, ya que permite ver qué partes del código están incluidas dentro de otras. Es importante acostumbrarse a escribir de esta forma (escalonada) del mismo modo que a introducir los comentarios ya que la claridad de los scripts es fundamental a la hora de depurarlos. Un poco de esfuerzo a la hora de crearlos puede ahorrar muchas horas a la hora de corregirlos o modificarlos meses más tarde.

Pasando ya al comentario del programa, como se puede ver, que permite almacenar tablas y, a partir de una simple petición, visualizarlas un determinado valor en pantalla.

Lo que es interesante es que la utilidad de los arrays no acaba aquí, sino que también se pueden utilizar toda una serie de funciones creadas para ordenarlos por orden alfabético directo o inverso, por claves, contar el número de elementos que componen el array además de poder moverse por dentro de él hacia delante o atrás.

De gran utilidad es también el ciclo *foreach* que recorre de forma secuencial el array de principio a fin como ya se mencionó anteriormente.

6.10 MANEJO DE CADENAS

Una de las variables más corrientes a las que hace frente en la mayoría de los scripts son las *cadenas*, que no son más que información de carácter no numérico (textos, por ejemplo).

Para asignar a una variable un contenido de este tipo, se escribe entre comillas dando lugar a declaraciones de este tipo:

```
$cadena="Esta es la información de mi variable"
```

Si se quiere ver en pantalla el valor de una variable o bien un mensaje cualquiera, se usa la instrucción print como ya se ha visto anteriormente. Se puede yuxtaponer o concatenar varias cadenas poniendo para ello un punto entre ellas:

```
<?php
$cadena1="Perro";
$cadena2=" muerde";
$cadena3=$cadena1 . $cadena2;
print $cadena3 //El resultado es: "Perro muerde"
?>
```

También, se pueden introducir variables dentro de la cadena, lo cual puede ayudar mucho en el desarrollo de los scripts.

La pregunta que se plantea uno ahora es: ¿Cómo se hace entonces para que en vez del valor "55" me salga el texto "$a"? En otras palabras, cómo se hace para que el símbolo ***$*** no defina

una variable sino que sea tomado tal cual. Esta pregunta es tanto más interesante cuanto que en algunos scripts, este símbolo debe ser utilizado por una simple razón comercial (pago en pesos por ejemplo) y si se escribe tal cual, la computadora va a pensar que lo que viene detrás es una variable siendo que no lo es.

Pues bien, para meter éste y otros caracteres utilizados por el lenguaje dentro de las cadenas y no confundirse, lo que hay que hacer es escribir una diagonal invertida delante:

Carácter	Efecto en la cadena
\$	**Escribe un signo de pesos en la cadena**
\"	**Escribe comillas en la cadena**
****	**Escribe una diagonal invertida en la cadena**
\8/2	**Escribe 8/2 y no 4 en la cadena**

Además, existen otras utilidades de esta diagonal invertida que permiten introducir en el documento HTML determinados eventos:

Carácter	Efecto en la cadena
\t	**Introduce una tabulación en nuestro texto**
\n	**Cambiamos de línea**
\r	**Retorno de carro**

Estos cambios de línea y tabulaciones tienen únicamente efecto en el código y no en el texto ejecutado por el navegador. En otras palabras, si se quiere que el texto ejecutado cambie de línea se ha de introducir un *print "<br>"* y no *print "\n"* ya que este último sólo cambia de línea en el archivo HTML creado y enviado al navegador cuando la página es ejecutada en el servidor.

Las cadenas pueden asimismo ser tratadas por medio de funciones de todo tipo y existen muchas posibles acciones que se pueden realizar sobre ellas: Dividirlas en palabras, eliminar espacios sobrantes, localizar secuencias, remplazar caracteres especiales por su correspondiente en HTML o incluso extraer las etiquetas META de una página web.

6.11 EXPRESIONES REGULARES

Las expresiones regulares permiten hacer una manipulación avanzada de las cadenas. Muy comúnmente no son cosas que luzcan con pies y cabeza, por ejemplo:

^.+@.+\..+$

Esto hace que muchos programadores no deseen utilizarlas, ya que a simple vista parece algo muy complejo. Sin embargo si se le dedica un poco de tiempo a aprenderlas se dará cuenta que facilitan mucho el manejo de cierto tipo de cadenas. Sobre todo porque la manipulación de cadenas es parte de la vida diaria de un desarrollador de webs:

- ✓ Validar entradas del usuario.
- ✓ Buscar caracteres dentro de cadenas.
- ✓ Comparar cadenas.

Utilizando expresiones regulares se puede saber si una cadena concuerda con un patrón específico o si una cadena contiene una subcadena que cumpla un determinado patrón. Las expresiones regulares soportadas por PHP cumplen con el estándar POSIX y están implementadas a través de la biblioteca regex de Henry Spencer.

Sintáxis de las expresiones regulares:

Hay que empezar por lo más facil:

"abc"

Una expresión regular válida consiste en una o más ramas, separadas por "|". La expresión resultará concordante con cualquier cosa que concuerde con una de sus ramas. Por tanto la expresión "abc" será concordante con cualquier cadena que contenga "abc".

La expresión regular "abc|def" concordará con cualquier cadena que contenga "abc" o "def".

Una rama consiste en una o más piezas. Una pieza es un conjunto de cosas relevantes. Véase el siguiente ejemplo:

"[abc]"

Esto es denominado una expresión entre corchetes (*bracket expression*), seguramente por que está englobada por corchetes. Como los corchetes especifican un conjunto de caracteres, no una cadena, esta expresión concordará con cualquier cadena que contenga "a", "b" o "c".

Si dos caracteres en esta lista estuviesen separados por "-", se estaría indicando el rango completo de caracteres entre los dos especificados, ambos inclusive: "[0-9]" concuerda con cualquier carácter numérico, "[a-Z]" concuerda con cualquier carácter desde la "a" minúscula hasta la "Z" mayúscula.

Nota: Para especificar un guión literal como parte de tu conjunto de caracteres, escríbelo en la última posición del mismo: "[abc-]".

Si un conjunto empieza con "^" (Por ejemplo: "[^abc]"), concordará con cualquier carácter no presente en dicho conjunto.

Se pueden utilizar ciertas abreviaturas para especificar clases de caracteres. Algunas de estas clases y abreviaturas de ven en el Anexo 4.

Las expresiones entre corchetes y los caracteres ordinarios pueden ir seguidos de "+", "*", "?" o de un conjunto (llaves).

Existen algunos caracteres especiales que pueden utilizarse en las expresiones regulares de POSIX:

- ✓ **"."** → Concuerda con un sólo carácter, cualquiera que sea.
- ✓ **"^"** → El comienzo de la cadena.
- ✓ **"$"** → El final de la cadena.

Nota: Para utilizar estos caracteres como caracteres ordinarios debes prefijarlos con la secuencia de escape ".^?".

6.12 PHP Y EL SISTEMA DE ARCHIVOS

PHP tiene opciones que permiten crear, modificar, leer y obtener información sobre archivos de cualquier tipo.

Para crear o modificar archivos se utiliza la instrucción:

```
$f1=fopen(archivo,modo)
```

$f1 es una variable que guarda el identificador del recurso, un valor importante para el posterior uso del archivo. **archivo** es el nombre (con extensión y probablemente la ruta[2]) del archivo a tratar y deberá escribirse entre comillas. **modo** es una cadena (se debe poner entre comillas) que guarda las diferentes opciones de apertura del archivo. A continuación se presenta una tabla con los diferentes modos de abrir el archivo:

Valores del parámetro modo de la función fopen	
Valor	**Funcionalidad**
r	Abre el archivo en modo lectura y coloca el puntero al comienzo del archivo
r+	Abre el archivo en modo lectura y escritura y coloca el puntero al comienzo del archivo
w	Abre el archivo en modo escritura y coloca el puntero al comienzo del archivo, reduce su tamaño a cero y si el archivo no existe intenta crearlo
w+	Abre el archivo en modo lectura y escritura y coloca el puntero al comienzo del archivo, reduce su tamaño a cero y si el archivo no existe intenta crearlo
a	Abre el archivo en modo escritura y coloca el puntero al final del archivo y si no existe intenta crearlo
a+	Abre el archivo en modo lectura y escritura y coloca el puntero al final del archivo y sino existe intenta crearlo

Una vez finalizado el uso de un archivo resulta conveniente cerrarlo. Para ello, PHP dispone de la siguiente instrucción:

```
fclose($f1)
```

Esta función –que devuelve un valor booleano-- permite cerrar el archivo especificado en $f1 que como se recordará es el valor del identificador de recurso que le fue asignado automáticamente por PHP en el momento de la apertura.

[2] Si se incluye junto con el nombre del archivo una ruta se debe de tomar en cuenta que bajo Windows hay que usar siempre el separador diagonal invertida (\).

Leyendo archivos

PHP permite obtener información de archivos en dos modalidades: sin apertura previa o requiriendo apertura previa.

Lectura de archivos sin apertura previa → Existen algunas funciones que permiten la lectura de archivos sin necesidad de que hayan sido abiertos previamente. Son las siguientes:

Función	Descripción
readfile(archivo)	Escribe directamente en el punto de inserción del script el contenido completo del archivo sin necesidad de ir precedido por print. Si se pone print o se recoge en una variable, además del contenido del archivo añadirá un número que indica el tamaño del archivo expresado en bytes.
$var=file(archivo)	Esta función crea sobre la variable $var un array escalar cuyos valores son los contenidos de cada una de las líneas del archivo. Una línea termina allí donde se haya insertado un salto de línea en el archivo original.

Lectura de archivos con apertura previa → Es obvio que para la utilización de estas funciones, los archivos han de ser abiertos en un modo que permita la lectura.

Función	Descripción
fpassthru($f1)	Esta función permite la lectura completa del archivo. Tiene un par de peculiaridades importantes: Cierra el archivo automáticamente después de la lectura. Por ello, si se escribe la función fclose dará error. Si se guarda en una variable, o si va precedido de ***'echo'*** además de escribir el contenido del mismo, añadirá al final el número de bytes que indican el tamaño del archivo.
fgets($f1,long)	Extrae del archivo señalado por el $f1 una cadena que comienza en la posición actual del archivo y cuya longitud esta limitada por el menor de estos tres valores: El valor (en bytes) indicado en long. La distancia (en bytes) de la posición actual del puntero al final del archivo. La distancia (en bytes) de la posición actual del puntero al primer salto de línea.
fgetc($f1,long)	Extrae un sólo caracter a partir de la posición actual del puntero en el archivo señalado por el identificador de recurso $f1.

Escribiendo en un archivo

Una vez abierto un archivo –en modo que permita escritura-- la función PHP que permite escribir es la siguiente:

```
fwrite($f1,"texto",long)
```

Donde: *$f1* sigue siendo el identificador de recurso; texto la cadena de texto a insertar en el archivo y long el número máximo de caracteres a insertar. Si la cadena de texto tiene, menor o igual longitud que el parámetro *long* la escribirá en su totalidad, en caso contrario sólo escribirá el número de caracteres indicados. También es posible utilizar:

```
fputs($f1,"texto",long)
```

Que en realidad es un alias de la función anterior.

Se debe tomar en cuenta que estas funciones realizan la inserción de la cadena a partir de la posición actual del puntero en el momento de ser invocadas. Si el archivo ya existiera y contuviera datos, los nuevos datos se sobrescribirían en el contenido anterior salvo en el caso de que el puntero estuviera en el final del archivo preexistente.

Borrando un archivo

Para borrar archivos, PHP dispone de la siguiente instrucción:

```
unlink(archivo)
```

archivo es una cadena que contiene el nombre y la extensión del archivo y en su caso también el *path*.

Copiando un archivo

La función:

```
copy(archivo1, archivo2)
```

Realiza una copia del archivo *archivo1* (indicar nombre y extensión) a otro archivo cuyo nombre y extensión son indicados en la cadena *archivo2*. Esta función, además de realizar la copia, devuelve un valor booleano indicando si la copia se ha realizado con éxito (TRUE) o (1) o FALSE (null) si por alguna razón no ha podido realizarse la copia.

Renombrando un archivo

La función:

```
rename(archivo1, archivo2)
```

Cambia el nombre del archivo archivo1 (indicar nombre y extensión) por el indicado en la cadena archivo2. También devuelve TRUE o FALSE. Si tratamos de renombrar un archivo inexistente, dará error.

Funciones informativas

PHP dispone de funciones que nos facilitan información sobre archivos. Se ven en el Anexo 4.

6.13 CGI

CGI (Common Gateway Interface) es una interfaz entre aplicaciones externas y servicios de información.

Un documento HTML, es algo estático, permanente, lo que no se adecua a las necesidades de interactividad, acceso a información en constante actualización, consulta a bases de datos o seguimiento, control o recuperación de resultados de un determinado proceso. Por ello, a veces es necesario accesar a una información que se está generando "en tiempo real". Así pues, es necesario contar con algún tipo de pasarela como la que define CGI.

La interfaz define una forma cómoda y simple de ejecutar programas que se encuentran en la máquina en la que se aloja el servidor. Para el cliente presenta una ventaja en el aspecto de la seguridad, ya que no tendrá que ejecutar ningún programa de efectos desconocidos en su sistema local. Además de eliminar la necesidad de aprendizaje, se resuelven los problemas de mantenimiento, operación y distribución de clientes ya que el acceso se realizará a través de cualquier cliente estándar de WWW y la comunicación se realizará según el protocolo HTTP.

Es necesario tener en cuenta algunos aspectos sobre seguridad, ya que tener un programa CGI equivale a que "TODO EL MUNDO podrá ejecutar un programa en un sistema propio". Esto se solventa incluyendo los programas CGI en un directorio determinado, denominado generalmente cgi-bin, directorio que no suele ser accesible a todos los usuarios. Cuando un servidor de WWW recibe como petición un documento alojado en este directorio, entenderá que se trata de un programa ejecutable.

Pero CGI plantea como inconveniente la necesidad de ejecutar programas en el servidor, de los cuales es necesario tener una copia por cliente ya que no permite hacer uso de recursos compartidos y, al tratarse de una pasarela, no es posible disponer de un control directo de la comunicación. Por ello se plantean nuevas alternativas, como el NSAPI de Netscape Corporation y, por supuesto, Java.

6.14 ACERCA DE LAS VARIABLES DE AMBIENTE

Dada su naturaleza de lenguaje de lado servidor, PHP es capaz de dar acceso a toda una serie de variables que informan sobre el servidor y sobre el cliente. La información de estas variables es atribuida por el servidor y en ningún caso nos es posible modificar sus valores directamente mediante el script. Para hacerlo es necesario influir directamente sobre la propiedad que definen.

Existen muchas variables de este tipo, algunas sin utilidad aparente y otras realmente interesantes y con una aplicación directa para el sitio web. A continuación se enumeran algunas de estas variables y la información que nos aportan:

Variable	Descripción
$HTTP_USER_AGENT	Informa principalmente sobre el sistema operativo y tipo y versión de navegador utilizado por el usuario. Su principal utilidad radica en que, a partir de esta información, se puede redireccionar a los usuarios hacia páginas optimizadas para su navegador o realizar cualquier otro tipo de acción en el contexto de un navegador determinado.
$HTTP_ACCEPT_LANGUAGE	Devuelve la o las abreviaciones del idioma considerada como principal por el navegador. Esta lengua o lenguas principales pueden ser elegidas en el menú de opciones del navegador. Esta

	variable resulta también extremadamente útil para enviar al usuario a las páginas escritas en su idioma, si es que existen.
$HTTP_REFERER	Indica la URL desde la cual el usuario ha tenido acceso a la página. Muy interesante para generar botones de "Atrás" dinámicos o para crear sistemas propios de estadísticas de visitas.
$PHP_SELF	Devuelve una cadena con la URL del script que está siendo ejecutado. Muy interesante para crear botones para recargar la página.
$_GET Anteriormente ($HTTP_GET_VARS)	Se trata de un array que almacena los nombres y contenidos de las variables enviadas al script por URL o por formularios GET
$_POST Anteriormente ($HTTP_POST_VARS)	Se trata de un array que almacena los nombres y contenidos de las variables enviadas al script por medio de un formulario POST
$_COOKIE Anteriormente ($HTTP_COOKIE_VARS)	Se trata de un array que almacena los nombres y contenidos de las cookies.
$PHP_AUTH_USER	Almacena la variable usuario cuando se efectúa la entrada a páginas de acceso restringido. Combinado con $PHP_AUTH_PW resulta ideal para controlar el acceso a las páginas internas del sitio.
$PHP_AUTH_PW	Almacena la variable password cuando se efectúa la entrada a páginas de acceso restringido. Combinado con $PHP_AUTH_USER resulta ideal para controlar el acceso a las páginas internas del sitio.
$REMOTE_ADDR	Muestra la dirección IP del visitante.
$DOCUMENT_ROOT	Devuelve la ruta física en la que se encuentra alojada la página en el servidor.
$PHPSESSID	Guarda el identificador de sesión del usuario.

No todas estas variables están disponibles en la totalidad de servidores o en determinadas versiones de un mismo servidor. además, algunas de ellas han de ser previamente activadas o definidas por medio de algún acontecimiento. Así, por ejemplo, la variable $HTTP_REFERER no estará definida a menos que el usuario acceda al script a partir de un enlace desde otra página.

6.15 LENGUAJES DE PROGRAMACIÓN CGI, COMPILADOS E INTERPRETADOS

CGI, como la propia palabra indica es una interfaz entre servidores de información y programas de aplicación. Por tanto, define una serie de reglas que deben cumplir tanto las aplicaciones como los servidores para hacer posible la presentación de resultados de programas ejecutables en tiempo real a través de servicios de información estandarizados. Por ello, se habla de gateway o pasarela entre una y otra dimensión. Al tratarse de una interfaz, no existe ningún tipo de dependencia con el lenguaje de programación empleado.

En principio, cualquier lenguaje es susceptible de ser utilizado para desarrollar programas CGI, ya sea interpretado o compilado. En "la red" existen aplicaciones CGI desarrolladas en C, C++, Fortran, Perl, Tcl, Visual Basic, AppleScript y cualquier shell de UNIX. Los lenguajes interpretados como sh, tcl y, sobre todo, perl, tienen mayor facilidad de mantenimiento y depuración. Otra ventaja es que, por lo general, suelen ser de más alto nivel, con lo que no permiten realizar ciertas maniobras con la memoria y es más difícil dejar procesos sueltos descontrolados en el sistema. Uno de los lenguajes más utilizados es el Perl que, siendo interpretado, proporciona una potencia similar a la de C con una sintaxis muy parecida.

La contrapartida es que un lenguaje compilado es siempre mucho más rápido que uno interpretado. En el caso de CGI la velocidad de ejecución es importante, ya que habrá que sumar el tiempo de ejecución a la latencia de red y a la velocidad de transmisión, tiempos de espera que tendrá que sufrir el usuario del cliente.

COMO DISEÑAR UNA APLICACIÓN CGI

ENTRADA DE DATOS A PARTIR DE FORMULARIOS Y ASIGNACIÓN DE NOMBRES A LOS CAMPOS DE UN FORMULARIO

Ciclos y condiciones son muy útiles para procesar los datos dentro de un mismo script. Sin embargo, en un sitio Internet, las páginas vistas y los scripts utilizados son numerosos. Muy a menudo es necesario que distintos scripts estén conectados unos con otros y que se sirvan de variables comunes. Por otro lado, el usuario interacciona por medio de formularios cuyos campos han de ser procesados para poder dar una respuesta. Todo este tipo de factores dinámicos han de ser eficazmente regulados por un lenguaje como PHP.

Las variables de un script tienen una validez exclusiva para el script y resulta imposible conservar su valor cuando se ejecuta otro archivo distinto aunque ambos estén enlazados. Existen varias formas de enviar las variables de una página a otra, de manera que la página destino reconozca el valor asignado por el script de origen:

Este tipo de transferencia es de gran utilidad ya que permite interaccionar directamente con el usuario.

Primeramente, se presenta una página con el formulario clásico a rellenar y las variables son recogidas en una segunda página que las procesa:

Nota: No siempre se definen automáticamente las variables recibidas por el formulario en las páginas web, depende de una variable de configuración de PHP: register_globals, que tiene que estar activada para que así sea.

$_POST

Aquí es posible recopilar en una variable tipo array el conjunto de variables que han sido enviadas al script por este método a partir de la variable de sistema $HTTP_POST_VARS.

```
Print "Variable \$nombre: " . $_POST["nombre"] . "<br>\n";
```

Nota: Aunque se pueden recoger variables con este array asociativo o utilizar directamente las variables que se definen en la página, resulta más seguro utilizar $_POST por dos razones, la primera que así se asegura que esa varible viene realmente de un formulario y la segunda, que así el código será más claro cuando se vuelva a leer, porque quedará especificado que esa variable se está recibiéndo por un formulario.

$_POST → A partir de PHP 4.1.0 se pueden recoger las variables de formulario utilizando también el array asociativo $_POST, que es el mismo que $HTTP_POST_VARS, pero más corto de escribir.

UN MÉTODO DIFERENTE DE ENTRADAS (GET)

Para pasar las variables de una página a otra se puede hacer introduciendo dicha variable dentro del enlace hipertexto de la página destino. La sintáxis sería la siguiente:

```
<a href="destino.php?variable1=valor1&variable2=valor2&...">Mi enlace</a>
```

Se puede observar que estas variables no poseen el símbolo $ delante. Esto es debido a que en realidad este modo de pasar variables no es específico de PHP sino que es utilizado por otros lenguajes. Ahora la *variable* pertenece también al entorno de la página destino.php y está lista para su explotación.

Las variables también pueden enviarse por el URL usando un formulario como en el apartado anterior y en lugar de definir como *method="POST"* se define como *method="GET"*.

Nota: No siempre se definen automáticamente las variables recibidas por parámetro en las páginas web, depende de una variable de configuración de PHP: *register_globals*, que tiene que estar activada para que así sea.

$ _GET

Sin embargo es posible recopilar en una variable tipo array el conjunto de variables que han sido enviadas al script por este método a partir de la variable de sistema $HTTP_GET_VARS, que es un array asociativo. Utilizándolo quedaría así:

```
<?php
print "Variable \$saludo: $HTTP_GET_VARS["saludo"] <br>\n";
print "Variable \$texto: $HTTP_GET_VARS["texto"] <br>\n"
?>
```

Nota: Aunque se pueden recoger variables con este array asociativo o utilizar directamente las variables que se definen en la página, resulta más seguro utilizar $HTTP_GET_VARS por dos razones, la primera que así se asegura que esa variable viene realmente de la URL y la segunda, que así el código será más claro cuando se vuelva a leer, porque quedará especificado que esa variable está recibiendose por la URL.

$_GET → A partir de la versión 4.1.0 de PHP se ha introducido el array asociativo $_GET, que es idéntico a $HTTP_GET_VARS, aunque un poco más corto de escribir.

6.16 ALGUNAS FUNCIONES UTILES

COOKIES

Existen páginas en la red que tienen que guardar distintas informaciones acerca de un usuario, por ejemplo su nombre, su edad o su color preferido. Y para ello cuentan con una serie de mecanismos en el servidor como bases de datos u otro tipo de contenedores, pero hay un mecanismo mucho más interesante de guardar esa información que los propios recursos del servidor, que es la propia computadora del usuario.

En las máquinas de los usuarios se guardan muchos datos que necesitan conocer las páginas web cada vez que se entra en la página, estas pequeñas informaciones son las *cookies*, que son *estados de variables que se conservan de una visita a otra en la computadora del cliente*.

Como es un poco peligroso que las páginas web a las que se accesa se dediquen a introducir cosas en la computadora, las cookies están altamente restringidas. Para empezar, solamente se pueden guardar en ellas textos, nunca programas, imágenes, etc. Además, los textos nunca podrán ocupar más de 1 KB, de modo que nadie podría inundarnos la computadora a base de cookies. Estas restricciones, unidas a la necesidad de poner una fecha de caducidad a las cookies para que estas se guarden, hacen que el aceptar cookies no signifique un verdadero problema para la integridad de los sistemas.

Un ejemplo típico de las cookies podría ser un contador de las veces que accede un usuario a una página. Se puede poner una cookie en la computadora del cliente donde se tendría una variable que lleva la cuenta de las veces que ha accedido a la página y cada vez que se accede se incrementa en uno.

Uso de las cookies

Para trabajar con cookies se tiene que utilizar un lenguaje de programación avanzado como Javascript o ASP, PHP, etc. No se puede entonces trabajar con cookies si solamente se dedica a utilizar el HTML, que ya se sabe que es un poco limitado para cosas que se salgan de mostrar contenido en páginas estáticas.

Polémica de las cookies

Existe un problema con estas '*galletitas*' y es que cortan la intimidad del usuario. Lo que se señalaba anteriormente acerca de guardar el perfil de un usuario puede llegar a ser un problema para él, por que se está vigilado y se apunta cada uno de los movimientos de dicho usuario, lo que puede convertirse en un abuso de información que no tiene por qué pertenecer a nadie más que al usuario. Las empresas que más utilizan esta clasificación del personal son los *AD-Servers* (servidores de banners).

La utilidad principal de las cookies es la de poder identificar al navegador una vez éste visita el sitio por segunda vez y así, en función del perfil del cliente dado en su primera visita, el sitio puede adaptarse dinámicamente a sus preferencias (idioma utilizado, colores de pantalla, formularios rellenados total o parcialmente, redirección a determinadas páginas...).

Para crear un archivo cookies, modificar o generar una nueva cookie se puede hacer a partir de la función SetCookie:

```
setcookie("nombre_de_la_cookie",valor,expiracion);
```

Es importante que la creación de la cookie sea previa a la apertura del documento HTML. En otras palabras, las llamadas a la función *setcookie()* deben ser colocadas antes de la etiqueta HTML.

Por otra parte, es interesante señalar que el hecho de que definir una cookie ya existente implica el borrado de la antigua. Del mismo modo, el crear una primera cookie conlleva la generación automática del archivo texto.

Para utilizar el valor de la cookie en los scripts tan sólo se tiene que llamar la variable que define la cookie. Hay que tener cuidado sin embargo de no definir variables en el script con el mismo nombre que las cookies puesto que PHP privilegiará el contenido de la variable local con respecto a la cookie y no dará un mensaje de error. Esto puede conducir a errores realmente difíciles de detectar.

Cabe recordar que es posible recopilar en una variable tipo array el conjunto de cookies almacenadas en el disco duro del usuario mediante la variable de servidor $HTTP_COOKIES_VARS

Las cookies son una herramienta muy útil para personalizar nuestra página pero hay que ser cautos ya que, por una parte, no todos los navegadores las aceptan y por otra, se puede deliberadamente impedir al navegador la creación de cookies. Es por ello que resultan un complemento y no una fuente de variables infalible para nuestro sitio.

SESIONES

En los ejemplos vistos hasta ahora, se han utilizado variables que sólo existían en el archivo que era ejecutado. Cuando se cargaba otra página distinta, los valores de estas variables se perdían a menos que se enviaran por medio de la URL o inscribirlos en las cookies o en un formulario para su posterior explotación. Estos métodos, aunque útiles, no son todo lo prácticos que podrían en determinados casos en los que la variable que se quiere conservar ha de ser utilizada en varios scripts diferentes y distantes los unos de los otros.

Se puede pensar que ese problema puede quedar resuelto con las cookies ya que se trata de variables que pueden ser invocadas en cualquier momento. El problema, como ya se ha dicho, es que las cookies no son aceptadas ni por la totalidad de los usuarios ni por la totalidad de los navegadores lo cual implica que una aplicación que se sirviera de las cookies para pasar variables de un archivo a otro no sería 100% infalible. Es importante a veces pensar en "la inmensa minoría", sobre todo en aplicaciones de comercio electrónico donde se deben captar la mayor cantidad de clientes posibles y los scripts deben estar preparados ante cualquier eventual deficiencia del navegador del cliente.

Resulta pues necesario el poder declarar ciertas variables que puedan ser reutilizadas tantas veces como se quieran dentro de una misma sesión. Imagínese un sitio multilingüe en el que cada vez que se quiera imprimir un mensaje en cualquier página se necesita saber en qué idioma debe hacerse. Se podría introducir un script identificador del idioma del navegador en cada uno de los archivos o bien declarar una variable que fuese válida para toda la sesión y que tuviese como valor el idioma reconocido en un primer momento.

Piénsese también en un carrito de la compra de una tienda virtual donde el cliente va navegando por las páginas del sitio y añadiendo los artículos que quiere comprar a un carrito. Este carrito podría ser perfectamente una variable de tipo array que almacena para cada referencia la cantidad de artículos contenidos en el carrito. Esta variable debería ser obviamente conservada continuamente a lo largo de todos los scripts.

Este tipo de situaciones son solventadas a partir de las variables de sesión. Una sesión es considerada como el intervalo de tiempo empleado por un usuario en recorrer las páginas hasta que abandona el sitio o deja de actuar sobre él durante un tiempo prolongado o bien, sencillamente, cierra el navegador.

PHP permite almacenar variables llamadas de sesión que, una vez definidas, podrán ser utilizadas durante este lapso de tiempo por cualquiera de los scripts del sitio. Estas variables serán específicas del usuario de modo que varias variables de sesión del mismo tipo con distintos valores pueden estar coexistiendo para cada una de las sesiones que están teniendo lugar simultáneamente. Estas sesiones tienen además su propio identificador de sesión que será único y específico.

Algunas mejoras referentes al empleo de sesiones han sido introducidas con PHP4. Es a esta nueva versión a la que se hace referencia a la hora de explicar las funciones disponibles y la forma de operar. Para los programadores de PHP3 la diferencia mayor es que están obligados a gestionar ellos mismos las sesiones definir sus propios identificadores de sesión.

Las variables de sesión pues, se diferencian de las variables clásicas en que éstas residen en el servidor, son especificas de un sólo usuario definido por un identificador y pueden ser utilizadas en la globalidad de las páginas.

Para iniciar una sesión se puede hacer de dos formas distintas:

- ✓ Declarar abiertamente la apertura de sesión por medio de la función *session_start()*. Esta función crea una nueva sesión para un nuevo visitante o bien recupera la que está siendo llevada a cabo.
- ✓ Declarar una variable de sesión por medio de la función *session_register('variable')*. Esta función, además de crear o recuperar la sesión para la página en la que se incluye también sirve para introducir una nueva variable de tipo sesión.
- ✓ Las sesiones han de ser iniciadas al principio del script. Antes de abrir cualquier etiqueta o de imprimir cualquier cosa. En caso contrario se recibirá un error.

Otras funciones útiles para la gestión de sesiones son:

Función	Descripción
session_id()	Devuelve el identificador de la sesión
session_destroy()	Dá por abandonada la sesión eliminando variables e identificador.
session_unregister('variable')	Abandona una variable sesión

6.17 CORREO

PHP tiene una excelente función que permite el envió de correos electrónicos desde una página web. El único problema es que son escasos los servidores gratuitos que tienen activada esta función.

La forma más simple de la función de correo es la etiqueta :

```
mail(dest,asunto,mensaje)
```

Donde:

- ✓ ***dest*** → Es la dirección de correo electrónico del destinatario.
- ✓ ***asunto*** → Es el texto que aparecerá como Asunto en el mensaje que reciba el destinatario.
- ✓ ***mensaje*** → El texto que aparecerá en el cuerpo del mensaje. No hay que olvidar escribir entre comillas esos tres parámetros de la función.

```
<?php
  mail("usuario@servidor.com", "Esto es el asunto","Aquí el mensaje");
?>
```

Una forma un poco más completa es:

```
mail(dest,asunto,mens,cabecera)
```

Como se puede ver aquín se añade un nuevo parámetro a la función (*cabecera*) que puede contener (separadas por comas) estas (y algunas otras) cosillas:

- ✓ **From: Nombre <e-mail>** → El texto que se escribe en el parámetro Nombre será el que aparezca el campo De: cuando el destinatario reciba el mensaje.
- ✓ **Reply-To: correo** → La dirección e-mail que se escriba donde dice correo será la dirección a la que se enviará la respuesta si el destinatario -una vez recibido el correo- desea responder con la opción Responder de su programa de correo electrónico.
- ✓ **Cc: correo1,correo2,...** → Con esta línea se puede incluir en correo1, correo2, etc. etc. las direcciones de correo electrónico de las personas a las que se desee enviar una copia del mensaje. Se deben separar con comas cada una de las direcciones
- ✓ **Bcc: correo1,correo2,...** → Esta opción es idéntica -en cuanto a funcionamiento- a la anterior con la única diferencia que esas direcciones no serán visibles por los destinatarios de los mensajes.
- ✓ **X-Mailer:PHP/".phpversion()** → Esta es una línea que se puede incluir para indicar que el mail ha sido elaborado con PHP a la vez que indica la versión utilizada de este.

```
<?php
  mail("usuario@servidor.com", "Esto es el asunto","Aquí el mensaje",
       "From: Usuario2 <usuario2@servidor.com>
       Reply-To: otro_usuario@servidor.com
       Cc: fulanito@hotmail.com, sutanito@yahoo.com
       Bcc:lus@yahoo.com, tere@sun.com.mx
       X-Mailer: PHP/" . phpversion());
?>
```

Con la programación en PHP, se puede elaborar una interfaz, embebida con html, para que el usuario tenga una mayor interacción con un sitio Web. PHP ampliamente utlizado con bases de datos, obteniendo de esta manera una mayor captación de datos. Se pueden ver en la red páginas relacionadas con manejo de imágenes, tiendas virtuales, etc., viendose con ello la extensibilidad y manejo de PHP.

CAPITULO VII

INTERACCIÓIN DE WWW CON BASES DE DATOS

MySql es un gestor de Bases de Datos multiusuario que gestiona bases de datos relacionales poniendo las tablas en ficheros diferenciados. Es muy criticado porque carece de muchos elementos vitales en bases de datos relacionales y no es posible lograr una integridad referencial verdadera. Es más utilizado en plataformas Linux aunque puede usarse en otras plataformas. Su uso en un servidor web es gratuito salvo en los casos que se necesite el uso de aplicaciones especiales.[1]

7.1 Requerimientos

Para instalar el servicio manejador de Bases de Datos con MySQL, es necesario contar con los siguientes elementos:

1. Servidor para aplicaciones Web → Apache.
2. Lenguaje de Programación → Montado sobre el servidor Apache.
3. Variables Globales de PHP → Revisar si están activas.
4. MySQL → Configurado y Activo.
5. Permisos → Usuario para Apache, Permisos de Trabajo.

Hay que cerciorarse de que el servidor de Apache este Instalado y en Ejecución:

```
ps –fea  | more

ps –fea  | grep httpd
```

Si estas instrucciones fallan, revisar si esta en ***/etc/apache***, que es el lugar predeterminado en donde se instala Apache. De lo contrario es necesario instalar el servicio, como se vio en el capítulo 3.

Configuración de *httpd.conf*

Es necesario que para reconfigurar el archivo ***httpd.conf***, se tiene que hacer una copia de respaldo.

Ahora lo que hay que hacer, es buscar las líneas que hacen referencia al los servicios de PHP, y quitar los comentarios, para activarlos

```
Include  /etc/apache/mod_php.conf

Include  /etc/apache/mod:ssl.conf
```

Dentro del mismo archivo es necesario verificar si está activada da ***información de los módulos del usuario***

```
LoadModule  info_module
```

[1] *omegash.tripod.com/glo.htm*

Hay que verificar que este habilitada la opción de recuperación de la información del Sevicio de Apache

También es necesario que se tengan activas las bitácoras para detección de errores, ubicado dentro del archivo de ***httpd.conf***. Este archivo se activa para que defina el archivo donde se guardarán los errores que se generen de PHP.

```
ErrorLog  /var/log/apache/error_log
```

Hay que cambiar, también, dentro del mismo archivo, el Nivel de Error de

```
LogLevel  warn
```

Por el mensaje de Error, para detectar en que líneas del codigo PHP se están generando los errores

```
LogLevel  error
```

Hechos los cambios pertinentes al archivo anterior, se tiene que detener el Servicio de Apache, hacerle un 'test' y volverlo a activar

```
apache  stop
apache  configtest
apache  start
```

Ya que se está seguro de que el servicio se ha configurado correctamente, es necesario abrir un Navegador y en la barra de dirección verificar que se esten ejecutando los servicios.

/localhost/server-info

- ✓ **Server-info** → Apache Server Information (API), sirve para ver los Módulos que están activos. Los API son servicios que traspasan la información a las Bases de DAtos

/localhost/Server-status

- ✓ **Server-status** → Reporte del Servicio de Apache, y despliega una lista de las transacciones que va haciendo el Servidor Apache.

Para poder ver la información correspondiente a PHP, es necesario activar el Servicio de ***ExtendedStatus***.

```
ExtendedStatus On
```

Como se van a hacer transacciones desde una base de datos de una página a otra, es necesario tener activada la opción de ***Variables Globales*** que se encuentran dentro del archivo ***php.ini*** ubicado en ***/etc/apache/***. Se sugiere que se haga previamente un archivo de respaldo.

Dentro del archivo *php.ini*, y activar las siguientes líneas:

```
display_errors = On
…

register_globals = On

register_argc_argv = On
```

Configuración para MySQL

Para empezar a trabajar con el servicio de ***MySQL*** que viene integrado a la versión de ***Linux Slackware 10***, es necesario hacer lo siguiente:

Hay que localizar los Archivos de configuración de Arranque de servicios que estan ubicados en ***/etc/rc.d/***. Hacer los respaldos correspondientes a los archivos ***rc.httpd*** y ***rc.mysql***.

Cambiar los permisos de lectura, escritura y ejecución al modo ***755*** del archivo ***rc.mysql***.

En otra Terminal hay que activar el servicio de MySQL, primero, accesando como el usuario ***mysql***.

```
root@inventario:~# su  mysql
```

Ya que se está como usuario ***mysql***, es necesario generar las ***tablas de Grant***, que son las Tablas de Configuración Básica de MySQL.

```
root@inventario:~# mysql_istall_db

root@inventario:~# exit
```

Para poder entrar a la consola del servicio de MySQL

```
root@inventario:~# /usr/bin/mysqld_safe  &    ↵
↵
root@inventario:~# mysql     ↵

> status
```

Con ***status*** Se verifica la configuración de mysql, y también se puede monitorear el Servicio de MySQL.

Hay que verificar que este correcto, también, el archivo correspondiente la *bitácora que lleva un registro de las Conexiones***.** Esto se hace desde otra terminal.

```
/var/run/mysql/mysql.sock
```

7.2 Uso de MySQL

Regresando a la terminal de ***mysql***. El usuario predeterminado que tiene el servicio de MySQL es MySQL

use

Comando que se usa para accesar a una Base de Datos.

```
> use  MySQL;    ↵
database  changed
```

Para poder mostrar las tablas creadas por predeterminanción para el usuario *nobody*, se hace con la siguiente instrucción:

```
> show tables;    ↵

select  * from User;    ↵

Host | User | Password | Select_priv | Insert_priv | Update_priv | Delete_priv
```

Es necesario crear un Usuario llamado ***Nobody*** dentro de la tabla ***USER*** para trabajar con los permisos para ejecutar los comandos SELECT, INSERT, UPDATE, DELETE; con los que se puede comenzar a trabajar con la creación y población de las Bases de Datos. Para ello, se escribe como se muestra a continuación:

```
> INSERT INTO  USER (Host, User, Password, Select_priv, Insert_priv, Update_priv, Delete_priv)
VALUES (“localhost”, “Nobody”, “Y”, “Y”, “Y”, “Y”);    ↵
```

Si se quiere agregar a un usuario ‘normal’ llamado ‘***usuario***’, se puede hacer lo anterior, pero ahora para el usuario ‘***usuario***’.

```
> INSERT INTO  USER (Host, User, Password, Select_priv, Insert_priv, Update_priv, Delete_priv)
VALUES (“localhost”, “usuario”, “Y”, “Y”, “Y”, “Y”);    ↵
```

Es necesario detener y reiniciar el Servicio de Mysql para poder empezar a trabajar con los usuarios que se acaban de crear para utilizar el Servicio de MySQL. Se hace de la manera siguiente:

```
root@inventario:~# mysqladmin  shutdown    ↵
```

En otra terminal, para ver que Bases de datos existen, se escribe:

```
root@inventario:~# mysqlshow    ↵
```

Por último, es necesario arrancar el servicio, y empezar a generar y poblar las bases de datos como se muestra en el siguiente aparatado.

```
root@inventario:~# /usr/bin/mysqld_safe  &  ↵
```

Para poder ver si el servicio esta activo

```
root@inventario:~# ps –fea  | grep mysqld
```

mysqlshow

mysqlshow es un comando que al ejecutarse muestra las Bases de Datos existentes.

A continuación se describen el comando y opciones para poder ver algunas caracteristicas de las bases de datos con el servicio de MySQL:

mysqlshow nombre_de_Base_de_Datos → Para ver las tablas que existen dentro de la Base de Datos.

mysqlshow Base_de_Datos Nombre_de_Tabla → Para ver las columnas dentro de una Tabla.

Para tener una copia de una base de datos e insertarla en otra ubicación se hace de la siguiente manera:

Si se quiere copiar directamente una base de datos a la unidad de Disco Flexible (previamente montada):

```
root@inventario:~# mcopy  Base_de_Datos.sql  a:
```

Después en otro servidor se pega y se activa de la siguiente manera (tomando en cuenta que está activo el servicio de MySQL):

```
root@inventario:~# mcopy  a: Base_de_Datos.sql

root@inventario:~# mysql

> create  database  Base_de_Datos;

> exit

root@inventario:~# mysql  Base_de_Datos  <  Base_de_Datos.sql
```

Por último, para ***Respaldar*** los datos de una Base de Datos, se hace como sigue:

```
root@inventario:~# /usr/bin/mysqladmin  shutdown

root@inventario:~# /usr/bin/mysqld_safe  &

root@inventario:~# /usr/bin/mysqldump  > respaldo.sql

root@inventario:~# /usr/bin/mysqladmin  shutdown
```

Y por ultimo, para agregar la base de datos ***respaldo.sql***

```
root@inventario:~# /usr/bin/mysql  <  respaldo.sql
```

7.3 Bases de Datos y MySQL

La relación entre PHP y bases de datos, especialmente MySQL y Postgres, es muy estrecha y beneficiosa. De hecho, cuando se habla de Web y PHP, es muy difícil que no se mencione también a una base de datos. Después de todo, el Web está pensado para almacenar y permitir los accesos a cantidades enormes de información. Mientras mayor sea la cantidad de información y más alta la frecuencia de actualización de un sitio web, mayor es su valor y sus ventajas sobre otros medios.

Tal vez la mayor ventaja de PHP sobre sus competidores es la integración con los sistemas de bases de datos y el soporte nativo a las distintas bases de datos existentes, libres y comerciales. Las razones principales para usar una base de datos son:

- ✓ Evitar redundancias.
- ✓ Evitar programas complicados.
- ✓ Búsquedas.
- ✓ Seguridad.

Arquitectura *n-tier (n - niveles)*

Una arquitectura cliente/servidor es una *2-tier* (2º grado), una *n-tie*r desagrega aún más las funciones, por ejemplo en web tenemos una *3-tier*:

- ✓ Presentación: Navegador Web.
- ✓ Lógica: Servidor web + programas o *scripts* PHP.
- ✓ Almacenamiento de Datos: base de datos.

La comunicación entre el *tier-1* y el *tier-2* es a través de HTTP como soporte de comunicación y HTML como representación de datos. La comunicación entre el *tier-2* y el *tier-3* es a través del *middleware*, en este caso PHP y las funciones de MySQL que se conectan al servidor. También puede hacerse mediante ODBC.

SQL

SQL, "***Structured Query Language***" representa un método estricto y más general de almacenamiento de datos que estándares anteriores. SQL es un estándar ANSI (www.ansi.org) y ECMA (www.ecma.ch).

La estructura básica de una base de datos relacional con SQL es muy simple. Una instalación de base de datos puede contener múltiples bases de datos, cada base de datos puede contener un conjunto de tablas. Cada tabla está compuesta de un conjunto de columnas cuidadosamente diseñadas y cada elemento (o entrada) de la tabla es una fila.

Hay cuatro sentencias de manipulación de datos soportado por la mayoría de los servidores SQL y que constituyen una gran parte de todas las cosas que se pueden hacer sobre una base de datos.

- ✓ SELECT
- ✓ INSERT
- ✓ UPDATE
- ✓ DELETE

Los cuatro tipos de sentencias permiten la manipulación de datos, pero no de la estructura de la base de datos. En otras palabras, se pueden usar para agregar o modificar la información almacenada en la base de datos, pero no para definir o construir una nueva base de datos. Para modificar la estructura, o agregar tablas y base de datos se usan las sentencias DROP, ALTER y CREATE:

- ✓ alter table articulos add secciones int after id;
- ✓ alter table articulos modify secciones int not null;

Cambiar Privilegios

El comando anteriormente citado para crear las tablas fue ejecutado desde el usuario privilegiado de la instalación del MySQL. Es muy mala idea trabajar con ese usuario desde las páginas PHP. Lo que se hace normalmente, a efectos de no develar las claves en el código PHP, es permitir el acceso a las base de datos al usuario con el que es ejecutado el servidor web (Apache en este caso).

Si el Servidor Apache está ejecutándose como el usuario "***webmaster***", se le puede dar la mayoría de los privilegios (seleccionar, insertar, modificar, borrar, crear y eliminar tablas) con el comando GRANT:

```
>GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP
>ON simple
>TO webmaster@localhost;
```

Luego se ejecuta el comando ***flush*** para asegurarse que los demás servidores MySQL actualizan esta información.

```
flush privileges;
```

Select

SELECT es el comando principal para obtener información de una base de datos. La sintáxis es muy sencilla:

```
SELECT campo1, campo2 FROM tabla WHERE condición;
```

En algunos casos, aunque no está recomendado, se pueden seleccionar todos las columnas de la tabla usando la siguiente sintaxis.

```
SELECT * FROM tabla WHERE secciones = 1;
```

Uniones

Una de las cosas que complica un poco el SELECT son las *uniones*, aunque es una de las propiedades más útiles de los *selects*. Un select sobre una base de datos que no soporte uniones, puede ser imaginada como una fila de una hoja de cálculo. Pero una base de datos SQL es por definición relacional. Las uniones son definidas mediante una cláusula WHERE:

```
>SELECT seccion, nombre, titulo FROM secciones, articulos
>WHERE secciones.id = articulos.id;
 +---------+---------+-----------------+ | seccion | nombre  | titulo            | +---------+---------+-----------------+ |    1      | Primera |
Primer articulo  | |   2     | Segunda | Segundo articulo | +---------+---------+-----------------+
```

Insert

El comando para insertar nuevos datos a una base de datos es el *insert*.

```
INSERT INTO tabla (col1, col2, col3) VALUES (val1, val2, val3);
```

o

```
INSERT INTO tabla VALUES (val1, val2, val3);
```

o

```
insert into secciones values(1, 'Primera');
insert into secciones values(2, 'Segunda');
```

```
>insert into articulos values (1, 'Primer articulo',
>  'Es el cuerpo del primer articulo y deberia estar bien');
>insert into articulos (seccion, titulo, texto) values (2, 'Segundo articulo',
>  'Es el cuerpo del segundo articulo.');
```

Con *insert*, como se ve en el ejemplo anterior, se empieza a poblar la tabla de una manera rápida y ordenada.

Update

Se utiliza para modificar datos que previamente han sido almacenados en la base de datos. Se selecciona el o los campos que se quieren alterar de una tabla determinada.

```
UPDATE table SET campo1='valor1, campo2='valor2'
WHERE condición
update articulos set seccion=1
  where id=2;
```

La clausula *where* es similar a la usada en el *select* para las uniones.

Delete

Con *delete* se elemina un elmento de una tabla, o una tabla de una dase de datos, borrando así la información referente a dicho objeto.

```
DELETE elemento FROM tabla WHERE condición;
delete from articulos where seccion=1;
```

Es muy importante especificar la cláusula *where*, de otra forma se borrarán todos los datos almacenados en la tabla especificada.

7.4 Funciones Básicas de PHP/MySQL

Si se quiere accesar a una base de datos desde PHP, primero debe conectarse al servidor MySQL:

```
$enlace = mysql_connect($hostname, $user, $password);
```

si se usan variables, o

```
$enlace = mysql_connect('localhost', 'master', 'laclave');
```

en el caso de constantes.

Los argumentos son opcionales, sino se especifican se supone "localhost" para el servidor, el mismo usuario que el servidor web para el usuario y clave vacía (en caso de no tener asignada una clave de usuario para la conexión a la base de datos).

Conexión a la base de datos

La función ***mysql_connect*** retorna una conexión o enlace (*link*) a la base de datos que luego será usado para ejecutar los comandos SQL. El uso y almacenamiento de dicho valor es opcional, ya que la mayoría de las funciones de MySQL usan por defecto la última conexión. En caso de que se tengan varios enlaces o conexiones, hay que especificar cuál usar.

Luego de establecida la conexión hay que seleccionar una base de datos:

```
mysql_select_db($database);
```

si se usa variable o

```
mysql_select_db('simple');
```

si se usan constantes.

```
<?php
    $con=mysql_connect();
    mysql_select_db("simple", $con);
    mysql_close($con);
?>
```

Conexiones Persistentes

Cada vez que un cliente se conecta y solicita una página PHP que se conectará a una base de datos, el proceso del Servidor Apache que lo ejecuta debe establecer la conexión al momento que se ejecuta el ***mysql_connect***. Esta operación normalmente involucra arrancar un nuevo proceso del servidor MySQL, por lo que se produce un retardo (latencia) en la ejecución y visualización de la página.

Para evitar que se arranque un nuevo proceso por cada conexión a la base de datos, el MySQL del PHP permite especificar conexiones persistentes. En este tipo de conexiones, la conexión se mantiene abierta y podrá ser reutilizada más tarde en otra ejecución. Por lo tanto el servidor MySQL no acaba, sino que espera por nuevas peticiones, lo que acelera muchísimo la ejecución de los programas.

Para ello no hay nada más que usar la función equivalente **mysql_pconnect**:

```
$enlace = mysql_pconnect($hostname, $user, $password);
```

Consultas e inserciones a la base de datos

A continuación se muestran unos ejemplos muy sencillos pero representativos de las operaciones más comunes con bases de datos y páginas PHP:

- ✓ Recuperar datos desde una base de datos y listarlos en la página web.
- ✓ Permitir que un usuario inserte nuevos datos en la base de datos mediante un formulario HTML.

Con estos pequeños ejemplos se muestran las funciones más importantes y ya serán capaces de realizar la mayoría de las operaciones comunes. Las operaciones de modificación son muy similares al INSERT, salvo que se usa el comando SQL UPDATE con una cláusula WHERE similar a las usadas en los SELECT.

En los ejemplos se trabajará con las tablas previamente creadas de artículos y secciones.

Obtener la Lista de Bases de Datos

Lo primero que se hace es conectarse a la base de datos y seleccionar las base de datos existentes en el servidor. Ello se logra con la función **mysql_list_dbs** que retorna un resultado consistente en la lista de base de datos. Luego sólo queda recorrer toda la lista con la función ***mysql_fetch_object*** e imprimir el nombre de cada base de datos.

```
<?php
   $con=mysql_pconnect()
      or die ("No pude conectarme");
   print "<B>Connectado</B><P>";

   $db_list = mysql_list_dbs($con);
   while ($row = mysql_fetch_object($db_list))
   {
    print $row->Database . "<br>";
   }
   mysql_close($con);
?>
```

Listar el contenido de una tabla

En el siguiente ejemplo se lista todo el contenido de la tabla secciones y también el número secciones que existen. Las funciones que se usan son:

- ✓ **mysql_select_db** → Esta función selecciona una base de datos para ser usada en las siguientes funciones. Si la función se ejecuta correctamente (el enlace es correcto y la base de datos existe) devuelve TRUE, caso contrario FALSE.

- ✓ **mysql_query** → Ejecuta un comando SQL y retorna un resultado (cursor) que mantiene la información necesaria para "recorrer" el conjunto de filas que cumplen con la condición especificada en el SELECT. Si el comando se ejecuta correctamente (la sintaxis es correcta, las tablas y columnas existen y se tiene permiso para hacer un SELECT) devuelve TRUE, caso contrario devuelve FALSE.

- ✓ **mysql_affected_rows** → Esta función devuelve la cantidad de filas afectadas o seleccionadas con el comando anterior. Es similar a la función **mysql_num_rows**, pero además puede ser usada para UPDATE o DELETE.

- ✓ **mysql_fetch_array** → Esta es una función muy potente y eficiente, ya que recupera de la base de datos la siguiente fila que interesa y pone todos los datos en un array que puede ser accesado de dos formas: con un índice numérico como un array normal, o como un array asociativo usando los nombres de las columnas especificados en el SELECT: En el ejemplo se usa como un array asociativo con los nombres de las columnas "id" y "nombre".

```
<?php

   $con=mysql_pconnect()
      or die ("No hay conexión ");

   mysql_select_db("simple", $con)
      or die("No se puede accesar a la base de datos");
   print "Seleccionando de simple<br><hr>";

   $resultado=mysql_query("select * from secciones", $con);
   $items = mysql_affected_rows($con);
   print "Número de filas: $items <p>";

   while( ($fila=mysql_fetch_array($resultado)) )
   {
    print $fila["id"] . ": " . $fila["nombre"] . "<p>";
    }

   mysql_close($con);
?>
```

Insertar filas desde un formulario

En el siguiente ejemplo se muestra como definir un formulario sencillo y luego insertar los valores del formulario en una tabla. En el ejemplo lo que se hace es permitir insertar campos a la tabla "secciones". Como se puede observar, no se emplea ninguna función nueva, sino que se hace uso de la instrucción SQL INSERT en la función **mysql_query** ya usada en el ejemplo anterior.

El funcionamiento del programa es muy sencillo, define un formulario de un sólo campo (nombre) y la acción del formulario es el mismo *script* PHP. Al principio del programa se verifica si la llamada es desde el formulario (mediante la verificación de la variable $submit), si es así, se inserta el valor de la variable "nombre" (definida en el formulario) en una nueva fila de la tabla "secciones". Hay que notar que no se especifica el valor de "id" ya que este valor es generado automáticamente por el MySQL (ver el *autoincrement* del create).

```
<?php
if ($submit)
  {
  $db = mysql_pconnect("localhost");
  mysql_select_db("simple",$db);
  $sql = "INSERT INTO secciones (nombre) VALUES ('$nombre')";
  $result = mysql_query($sql);
  print "Información introducida.\n";
}
else
 {
  ?>
  <form method="post" action="<?php echo $PHP_SELF?>">
  Nombre:<input type="Text" name="nombre"><br>
  <input type="Submit" name="submit" value="Enter information">
  </form>
<?php
  } // end if
?>
```

Unión e inserción más compleja

En el siguiente ejemplo lo que se hace, es permitir insertar nuevos artículos, pero hay un requerimiento nuevo: cada artículo debe tener una sección asociada (en el campo "***seccion***"), por lo que es importante que el usuario especifique un valor correcto. Es decir, que el código de sección (y su nombre) exista en la tabla "secciones". La solución es sencilla sino hay demasiados valores posibles: poner un campo de entrada del tipo "SELECT" donde se listan los códigos y nombres de las secciones existentes.

```
<0php
$db = mysql_pconnect("localhost");
mysql_select_db("simple",$db);
if ($submit)
{
  $sql = "INSERT INTO articulos (seccion, titulo, texto)
         VALUES ($seccion, '$titulo', '$texto')";
  $result = mysql_query($sql);
  if($result)  {  print "<h3>Información introducida.</h3>\n"; }
  else {   print "<h3>No se pudo introducir el artículo</h3>\n";  }
}

// Se listan todos los artículos y sus secciones (UNION)
$result = mysql_query("select nombre, titulo from articulos, secciones
             where articulos.seccion=secciones.id", $db);
while ( ($datos = mysql_fetch_array($result)) )
{    print $datos["titulo"] . " (<i>" . $datos["nombre"] . "</i>)<br>\n";  }

// Damos la opcion de recargar la página
print "<CENTER><A HREF=\"$PHP_SELF\">Actualizar</A></CENTER>";
?>
  <H1>Insertar nuevo artículo</H1>
  <form method="post" action="<?php echo $PHP_SELF?>">
  Sección: <select name=seccion>
<?php   // Ahora se seleccionan de la BD las secciones existentes
  $result = mysql_query("select id, nombre from secciones
         order by nombre asc", $db);
  while ( ($datos = mysql_fetch_array($result)) )
{
   printf("<option value=%d>%s</option>\n", $datos["id"], $datos["nombre"]);
  }
?>
  </select><br>
  Título:<textarea cols="40" rows="2" name="titulo"><? echo $titulo ?></textarea><br>
  Texto:<textarea cols="40" rows="10" name="texto"><? echo $texto ?></textarea><br>
  <input type="Submit" name="submit" value="Enter information">
  </form>
```

También se muestra un ejemplo de UNION. Interesa mostrar en la página un listado de todos los artículos y la sección a la que pertenecen. Para poder obtener el nombre de la sección y los títulos de los artículos se debe hacer una unión entre la tabla de artículos y la de secciones y seleccionar las columnas que interesan (título del artículo y nombre de la sección). Se ve en el siguiente ejemplo:

Conociendo el manejo de las bases de datos, la adquisición, inserción y manejo de datos, se puede trabajar con un volumen de información de tamaño considerable, que puede contener miles o incluso millones de registros; se puede acceder de una manera rápida a algún elemento de dicha colección de datos. Todo esto lo permite el manejador de datos MySQL, aunque no es el único existente en el mercado de la información.

Agregando a esto, la interacción con la programación y el entorno Web, hacen que el uso de información se haga de una manera mucho más dinámica para el usuario, permitiendo con ello manipularla de manera remota, por medio de Internet.

Así, en la actualidad, como se está trabajando ya a nivel mundial, y con información de diferente tipo, al manejar MySQL e interactuar con el ambiente Web y la programación, hacen que el trabajo con la información se haga de manera sencilla y fácil para cualquier usuario, sin necesidad, incluso, de desplazarse de un lugar a otro.

CAPITULO VIII

PROGRAMACIÓN CON VISUAL BASIC .NET

Cuando se habla de Visual Basic .NET, se trata de un sistema potente que hace que sea posible la creación de Interfaces potetes, extensibles y versátiles, que ayudan a que el desempeño de las tareas, en un aplio espectro de trabajos. Es necesario empezar a entender el Entorno de Desarrollo. Para ello hay que comprender la Estructura de una Solución:

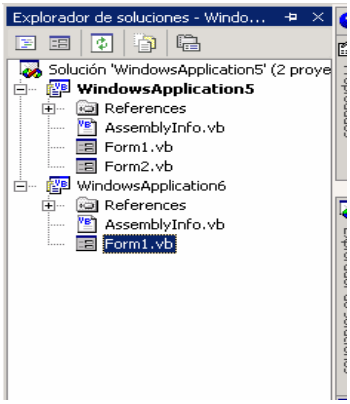
- ✓ Una solución puede constar de varios proyectos, escritos en lenguajes diferentes y creados con el uso de distintas tecnologías.

8.1 INTERFAZ DEL USUARIO

La interfaz del usuario, es medio por el cual se va a interactuar con el programa por medio del uso de la PC. Es decir, que ayuda al usuario a realizar tareas por medio de un ambiente amigable, que facilite la realización de determinadas tareas. A continuación de describen brevemente algunos elementos que se encuentran dentro de la interfaz del usuario.

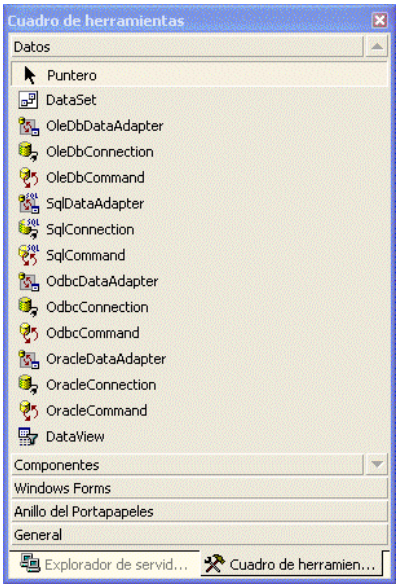
EXPLORADOR DE SOLUCIONES

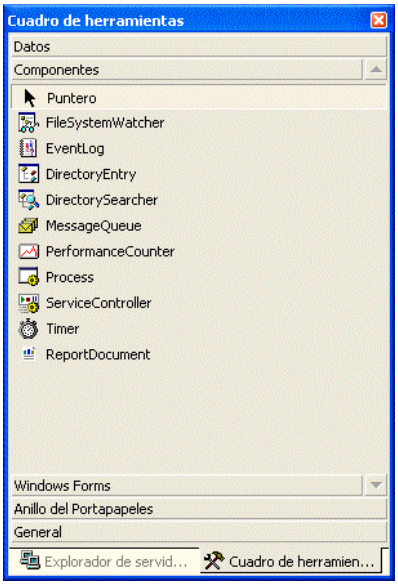
- ✓ Se pueden crear varios proyectos en una misma solución, a diferencia de las anteriores versiones de Visual Basic en las que sólo se podía generar un proyecto.

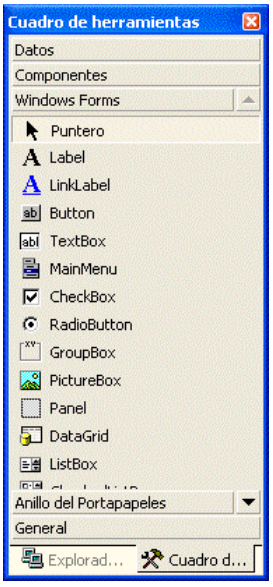


EL CUADRO DE HERRAMIENTAS

Contiene los objetos y controles que se pueden agregar a los formularios de Windows y formularios Web.





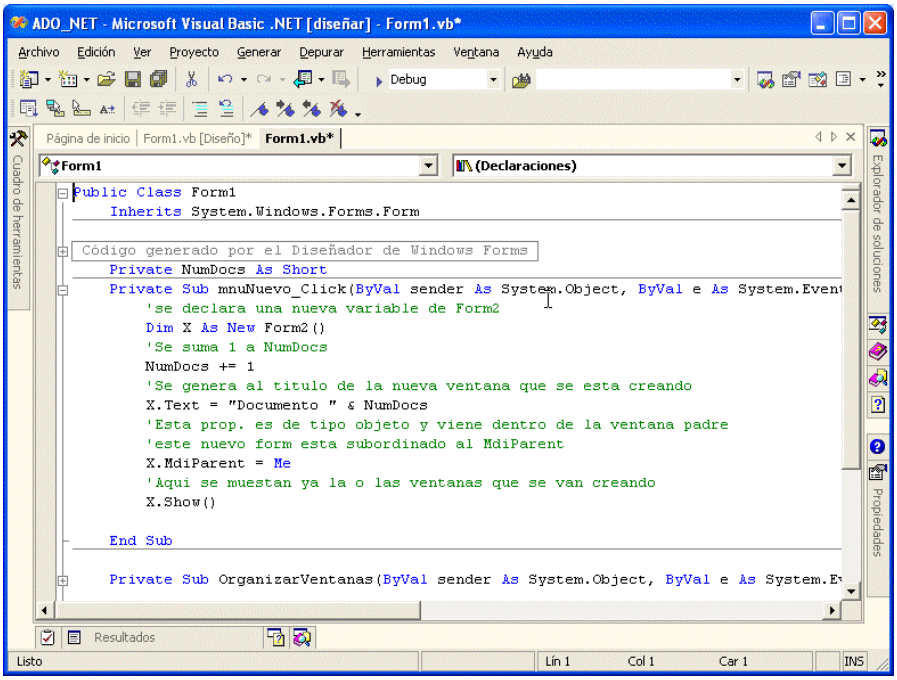


EL EDITOR DE CÓDIGO

Es el lugar donde se pasará una buena parte del tiempo escribiendo funciones y código.

Para poderlo desplegar, se hace lo siguiente:

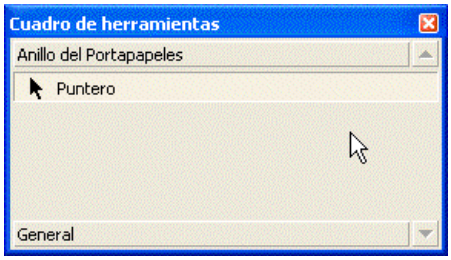
- ✓ En el explorador de soluciones, se da clic derecho sobre el formulario.
- ✓ Otra forma es ir al menú ***Ver*** → ***Código***. O simplemente pulsando la tecla **F7**



EL ANILLO DEL PORTAPAPELES

Esta ficha almacena los últimos doce elementos agregados al Anillo del Portapapeles del sistema mediante los comandos Cortar o Copiar. Estos elementos pueden arrastrarse desde la ficha Anillo del Portapapeles y colocarse en la superficie activa de edición o diseño. El Cuadro de herramientas está disponible desde el menú Ver.

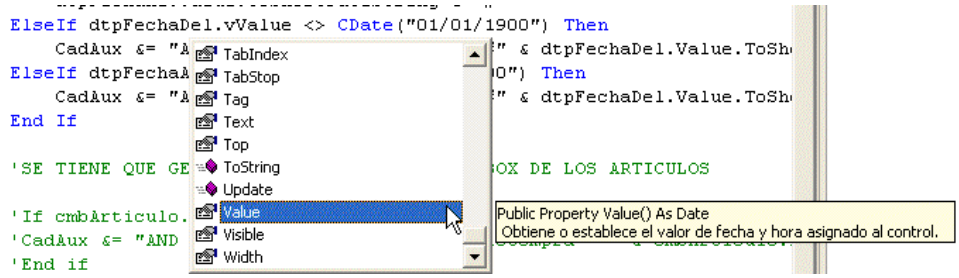
INTELLISENSE



IntelliSense ofrece una serie de opciones que facilitan el acceso a referencias del lenguaje. Puede guardar el contexto, buscar la información que necesite, incluso, dejar que IntelliSense termine de escribir el texto automáticamente.

IntelliSense se compone de las siguientes opciones:

- ✓ Lista de miembros
- ✓ Información de parámetros
- ✓ Información rápida
- ✓ Palabra completa
- ✓ Relleno automático de llaves



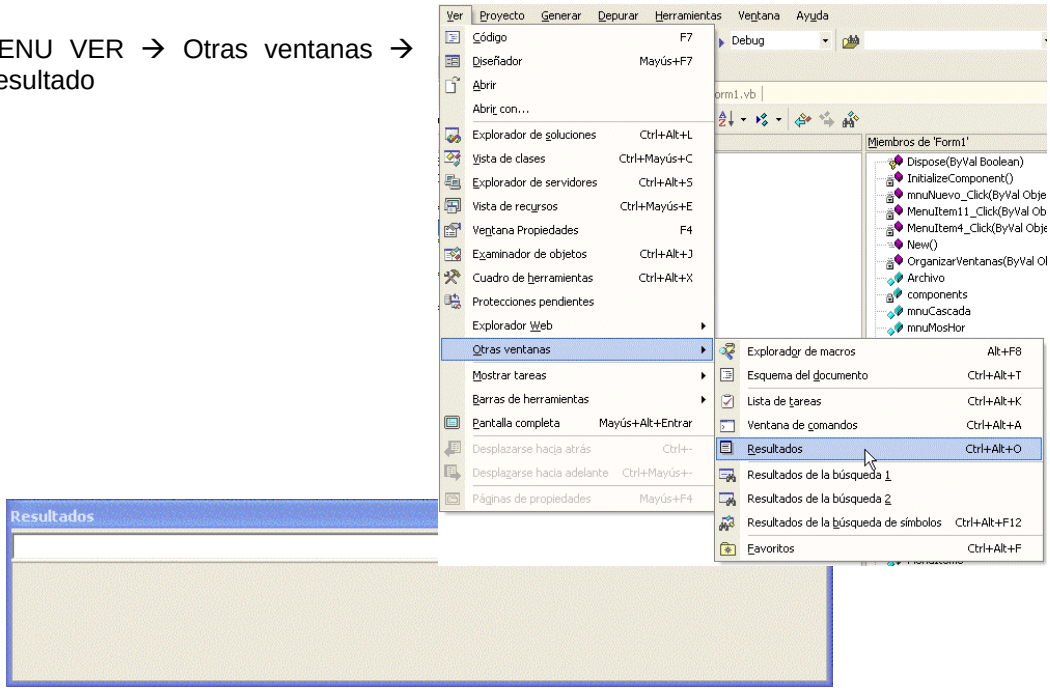
LA LISTA DE TAREAS

- ✓ Proporciona detalles sobre errores al compilar el código o al escribirlo.
- ✓ Ayuda a documentar los proyectos
- ✓ Gestiona tareas dentro de una solución. Con el menú Ver -> Otras Ventanas

LA VENTANA DE RESULTADOS

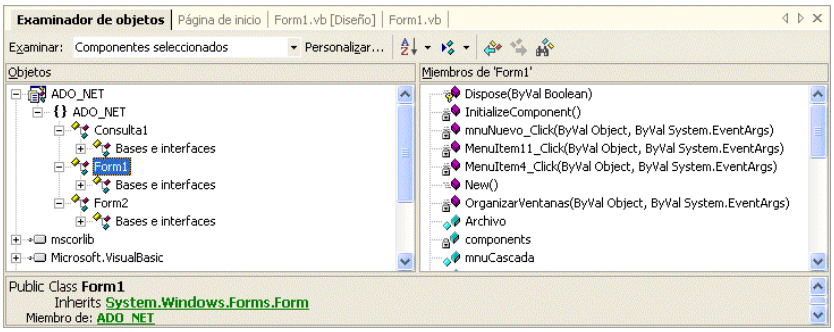
Se utiliza para mostrar información relativa al compilador al generar el proyecto o la solución.

- ✓ MENU VER → Otras ventanas → Resultado



EL EXAMINADOR DE OBJETOS

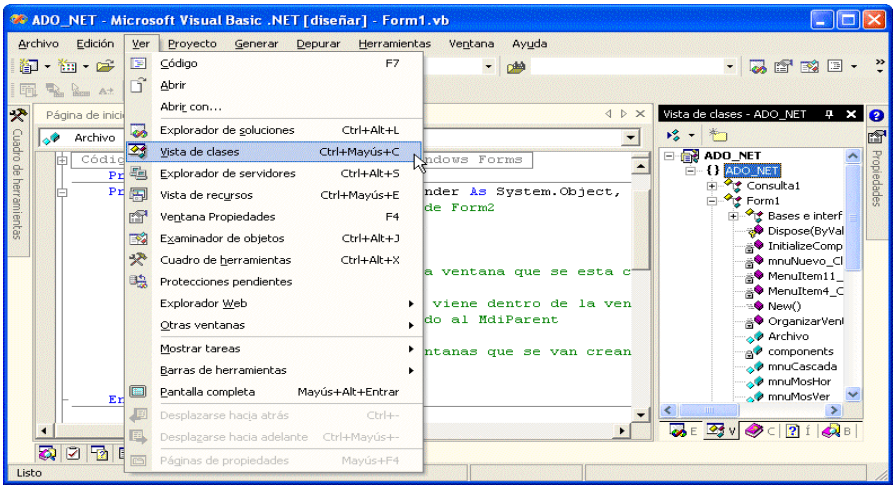
Permite ver todos los objetos en la solución así como sus propiedades y métodos. Para poder acceder a él, con el menú Ver → Explorador de Objetos



LA VISTA DE CLASES

Es una vista de alto nivel de todas las clases y objetos de la solucion.

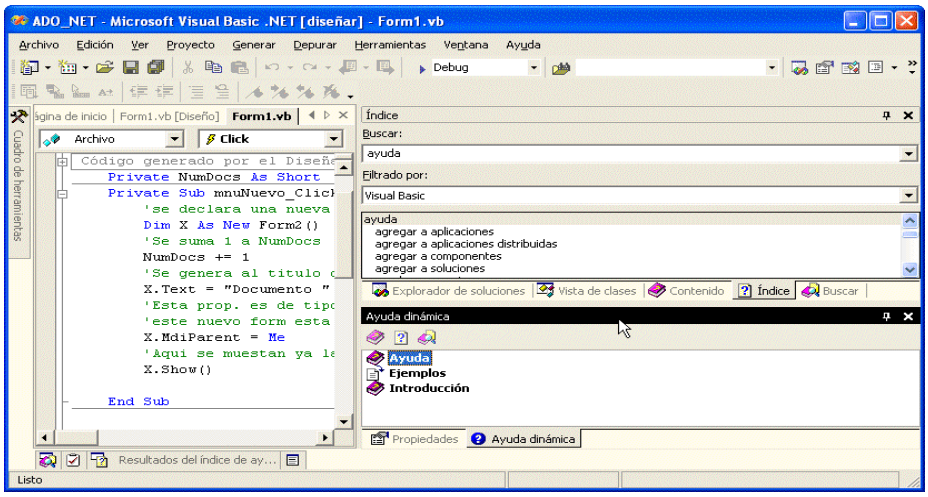
- ✓ Menu Ver → Vista de Clases



EL SISTEMA DE AYUDA

El sistema de Ayuda, sirve de apoyo al usuario en el manejo y funcionamiento de todos los elementos que verá con Visual Basic .NET.

- ✓ Pulsando la tecla **F1**



8.2 .NET FRAMEWORK

Proporciona los servicios necesarios para desarrollar e implantar las aplicaciones dentro del entorno de internet.

Los componentes fundamentales de esta familia son:

- ✓ **CLR** (Entorno común de ejecución)
- ✓ **BCL** (Biblioteca de clases Básicas unificadas)

CLR (COMMON LANGUAGE RUNTIME)
Entorno común de ejecución

Las funciones del CLR son:

- ✓ Un entorno de ejecución seguro y sólido:
  - ➢ Proporciona un entorno que gestiona al código al ejecutarlo.
  - ➢ Tratamiento de excepciones
  - ➢ Recogida de basura. (GC, Garbage Collection) Control automático de la caducidad de objetos.
- ✓ Procesos simplificados de desarrollo
  - ➢ Funciones orientadas a objetos → Simplifica la reutilización y la interoperabilidad entre componentes
- ✓ Compatibilidad multilenguaje
  - ➢ **CLS**. (Common Language Specification) especificación Común del Lenguaje
  - ➢ **CTS**.(Common Type System) Sistema Común de Clase.
- ✓ Gestión e implantación simplificados
  - ➢ ENSAMBLADOS

BCL (Biblioteca de clases Básicas unificadas)

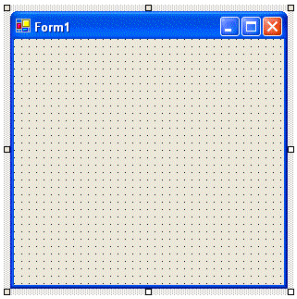
La biblioteca de clases de .NET Framework es una biblioteca de clases, interfaces y tipos de valor que se incluye en Microsoft .NET Framework SDK. Esta biblioteca brinda acceso a la funcionalidad del sistema y es la base sobre la que se crean las aplicaciones, los componentes y los controles de .NET Framework. Proporciona un conjunto de objetos jerarquizados clasificados por su funcionalidad.

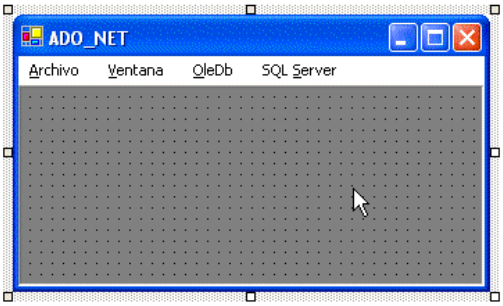
Para poder entender que es una Biblioteca de Clases, antes, es necesario saber que utilizan Espacios de Nombres. Un Espacio de Nombres, en el caso que ocupa a este tema, contiene clases para crear aplicaciones para Windows que aprovechan todas las ventajas de las características de la interfaz de usuario disponibles en el sistema operativo Microsoft Windows de la Biblioteca de Clases System.Windows.Forms (Espacio de Nombres)

BIBLIOTECA DE CLASES SYSTEM.WINDOWS.FORMS DE .NET FRAMEWORK

System.Windows.forms.form

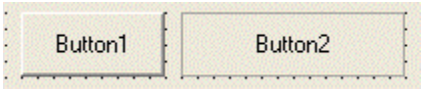
Form es una representación de cualquiera de las ventanas que se muestran en una aplicación. La clase Form se puede usar para crear ventanas estándar, de herramientas, sin bordes y flotantes. También se puede utilizar la clase Form para crear ventanas modales, como cuadros de diálogo. El formulario MDI (interfaz de múltiples documentos), que es un tipo especial de formulario, puede contener otros formularios, denominados formularios MDI secundarios. Un formulario MDI se crea estableciendo la propiedad IsMdiContainer en true. Los formularios MDI secundarios se crean estableciendo la propiedad MdiParent en el formulario MDI principal que va a contener el formulario secundario.





System.Windows.forms.button

Se puede hacer clic en **Button** utilizando el mouse (ratón), la tecla ENTRAR o la BARRA ESPACIADORA si el botón tiene foco. Se establece la propiedad AcceptButton o CancelButton de un objeto Form para permitir a los usuarios hacer clic en un botón presionando ENTRAR o ESCAPE incluso si el botón no tiene foco. Esto proporciona al formulario el comportamiento de un cuadro de diálogo. Al mostrar un formulario mediante el método ShowDialog, se puede utilizar la propiedad DialogResult de un botón para especificar el valor devuelto de **ShowDialog**.



System.Windows.Forms.Checkbox

Se utiliza un control **CheckBox** para dar al usuario una opción del tipo verdadero/falso o sí/no. El control de la casilla de verificación puede mostrar una imagen o texto o ambos.

System.Windows.Forms.Combobox

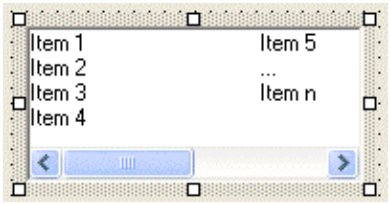
Un **ComboBox** muestra un campo de edición combinado con un ListBox y permite al usuario seleccionar una opción de la lista o escribir texto nuevo. El comportamiento predeterminado de **ComboBox** es mostrar un campo de edición con una lista desplegable oculta. Puede escribir un valor que permita lo siguiente: una lista desplegable simple, en que la lista se muestra siempre; un cuadro de lista desplegable, en que la parte de texto no se puede editar y es necesario presionar una flecha para ver el cuadro de lista desplegable; o el cuadro de lista desplegable predeterminado, en que la parte de texto se puede editar y el usuario debe presionar la tecla de flecha para ver la lista. Para que siempre se muestre una lista que el usuario no puede editar, use un control **ListBox**.

System.Windows.Forms.Label

Los controles **Label** se utilizan normalmente para proporcionar texto descriptivo de un control. Por ejemplo, se puede utilizar un objeto **Label** para agregar texto descriptivo para un control TextBox e informar al usuario del tipo de datos que se espera tener en el control. Los controles **Label** se pueden utilizar también para agregar texto descriptivo a un Form para proporcionar al usuario información útil. Los controles **Label** se pueden utilizar también para mostrar información en tiempo de ejecución sobre el estado de una aplicación. Por ejemplo, se puede agregar un control **Label** a un formulario para mostrar el estado de cada archivo cuando se procesa una lista de archivos.

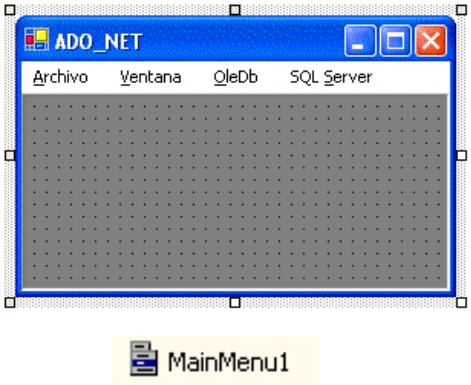
System.Windows.Forms.Listbox

El control **ListBox** permite mostrar una lista de elementos para que el usuario los seleccione haciendo clic en ellos. Un control **ListBox** puede proporcionar una o varias selecciones mediante la propiedad SelectionMode. **ListBox** también proporciona la propiedad MultiColumn para poder mostrar los elementos en columnas en lugar de mostrarlos en una lista vertical. De este modo, el control puede mostrar más elementos a la vez, y el usuario no tiene que buscar y desplazarse hasta un elemento.



System.Windows.Forms.MainMenu

El control **MainMenu** representa el contenedor para la estructura de menú de un formulario. Un menú está formado por objetos MenuItem que representan los comandos de menú individuales de la estructura de menú. Cada objeto **MenuItem** puede ser un comando de la aplicación o un menú primario para otros elementos de submenú. Para enlazar **MainMenu** con el objeto Form que lo mostrará, asigne **MainMenu** a la propiedad Menu de **Form**.



System.Windows.Forms.Messagebox

No se puede crear una nueva instancia de la clase **MessageBox**. Para mostrar un cuadro de mensaje, llame al método estático (**Shared** en Visual Basic) MessageBox.Show. El título, el mensaje, los botones y los iconos que aparecen en el cuadro de mensaje están determinados por los parámetros que se pasan a este método. Por ejemplo con el siguiente código:

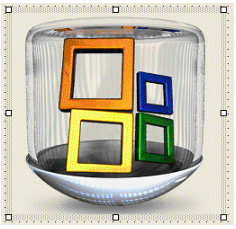
```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click

        MessageBox.Show("Mensaje de Texto")
        End
    End Sub
```

Al pulsar el boton (Button1) aparecerá una cuadro de mensaje.

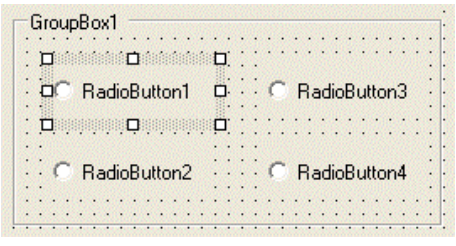
System.Windows.Forms.PictureBox

Se suele utilizar **PictureBox** para mostrar gráficos de un archivo de mapa de bits, metarchivo, icono, JPEG, GIF o PNG.



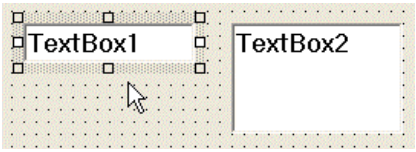
System.Windows.Forms.RadioButton

Cuando el usuario selecciona un botón de opción en un grupo, los demás se desactivan automáticamente. Todos los controles **RadioButton** de un contenedor determinado, como Form, constituyen un grupo. Para crear varios grupos en un formulario, coloque cada grupo en su propio contenedor, como un control GroupBox o Panel.



System.Windows.Forms.TextBox y StatusBar

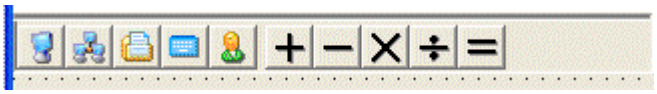
El control TextBox permite al usuario escribir texto en una aplicación. Este control tiene funcionalidad adicional que no se encuentra en el control de cuadro de texto de Windows estándar, como el enmascaramiento de caracteres de contraseña y la edición de múltiples líneas.



Normalmente, un control **StatusBar** está formado por objetos StatusBarPanel, cada uno de los cuales muestra texto o un icono. También puede proporcionar paneles personalizados dibujados por el usuario como una barra de progreso o una serie de imágenes que muestren el estado de la aplicación.

System.Windows.Forms.ToolBar

Los controles **ToolBar** se utilizan para mostrar los controles ToolBarButton que pueden aparecer como un botón estándar, un botón de alternar o un botón desplegable. Se pueden asignar imágenes a los botones creando un ImageList, asignándolo a la propiedad ImageList de la barra de herramientas y asignando el valor de índice de la imagen a la propiedad ImageIndex para cada **ToolBarButton**. A continuación, se puede asignar texto que aparecerá debajo o a la derecha de la imagen estableciendo la propiedad Text de **ToolBarButton**.



8.3 Programación Orientada a Objetos

Todo .NET Framework está basado en clases (u objetos). A diferencia de las versiones anteriores de Visual Basic, la versión .NET de este lenguaje basa su funcionamiento casi exclusivamente en las clases contenidas en .NET Framework, además casi sin ningún tipo de limitaciones. Debido a esta dependencia en las clases del .NET Framework y sobre todo a la forma "hereditaria" de usarlas, Visual Basic .NET tenía que ofrecer esta característica sin ningún tipo de restricciones.

En la Programación Orientada a Objetos (POO) se tienen los siguientes elementos:

- ✓ Clases
- ✓ Objetos
- ✓ Métodos
- ✓ Constructores y destructores
- ✓ Fundamentos herencia
- ✓ Polimorfismo
- ✓ Encapsulamiento
- ✓ Nombres de espacios

Clases

Las Clases representan simbólicamente los objetos de una aplicación. Describen las propiedades, métodos y eventos que conforman cada objeto. Se puede definir entonces que una Clase es un conjunto de objetos que comparten características similares (propiedades y métodos).

Por ejemplificar de alguna manera a las Clases, son como una receta que puede utilizarse para preparar varias veces el mismo alimento. En las Clases se puede:

- ✓ Reutilizar parte de su procedimiento para producir un alimento diferente
- ✓ Los ingredientes de un plato determinado (atributos)
- ✓ La forma en como deben prepararse (método)

Instancia de una clase

- ✓ Crear una copia de una clase.
- ✓ En el caso de 'instanciar' una receta, se tendría que el objeto que se crea es el alimento mismo.

Encapsulamiento

El Encapsulamiento hace posible ocultar el código utilizado para implementar el objeto, dejando visible solamente su funcionalidad:

- ✓ Capacidad de los objetos de contener y controlar el acceso a sus elementos.
- ✓ Esto es posible agrupando estos elementos en clases

Por ejemplo, supóngase la Clase ***cliente.*** Los métodos que tendría esta clase serían:

- ✓ Altas
- ✓ Bajas
- ✓ Consultas

Métodos de una clase

Los métodos de una clase son los procedimientos o funciones declarados dentro del cuerpo de una clase.

Propiedades de una clase

Las propiedades de una Clase constituyen una extensibilidad de las variables. Tanto las variables como las propiedades se denominan mediante tipos asociados. La diferencia que existe es que las propiedades no ubican almacenamiento, las variables si.

Propiedades

Las propiedades poseen descriptores de acceso que permiten la lectura y escritura de sus valores. Las declaraciones de propiedad incluyen dos partes: una que es similar a la declaración de un campo y otra que incluye un descriptor de acceso ***get*** o ***set***.

Ejemplo de Propiedades:

Declara una propiedad llamada **pNombre**, de tipo cadena (*string*). La propiedad utiliza una variable de tipo pública denominada Nombre y que fue declarada previamente en un una clase.

- ✓ **Get** → Permite que la propiedad pNombre retorne a un procedimiento llamado algún valor pasado por el mismo.
- ✓ **Set** → Se utiliza para devolver el valor de una propiedad.

Objeto

Un Objeto es un elemento que posee características básicas: estado y comportamiento.

Por ejemplo se puede tener como objeto en Visual Basic .Net una etiqueta o un botón de comando. Es posible comprender el "***qué***" es lo que hace el objeto (funcionalidad) y no el "***cómo***" lo hace (implementación)

Herencia

La Herencia es la cualidad de crear clases que estén basadas en otras clases. Entonces, La nueva clase tendrá todas las propiedades y métodos de la clase de la que esta derivada, además de poder modificar el comportamiento de los procedimientos que ha heredado, así como añadir nuevos.

Fundamentos de Herencia

- ✓ Todas las clases creadas con Visual Basic .net son heredables por defecto.
- ✓ Lo anterior no se aplica si se utiliza la palabra reservada NotInheritable
- ✓ La instrucción Inherits se utiliza para declarar una nueva clase, derivada de una clase base existente
- ✓ Las clases derivadas o instanciadas sólo pueden heredar de una clase base.
  - ➢ Una clase derivada hereda métodos de su clase base.

Modificadores para métodos y propiedades

- ✓ **Overridable** → Se utiliza para reemplazar un método o propiedad en una clase derivada
- ✓ **Overrides** → Se utiliza para reemplazar un método o propiedad **Overridable** definido en la clase base
- ✓ **NotOverridable** → No permite acción de Overridable.

Polimorfismo

El Poliformismo implica definir métodos en una clase base y que se reemplazan con nuevas implementaciones en las clases derivadas. Polimorfismo es la posibilidad de utilizar diferentes propiedades o métodos con el mismo nombre, de forma que cuando se utilicen, ya no hay que preocuparse de identificar la clase a la cual pertenezca.

Espacio de Nombres (*NAMESPACE*)

Un espacio de Nombre esta formado por un conjunto de clases, funciones y tipos de datos que permiten la funcionalidad de los objetos que se construyan basados en el. Los espacios de nombre son capaces de almacenar todas las clases, funciones y estructuras de datos necesarios para brindar funcionalidad a cualquier objeto.

Se tiene entonces que los Espacios de nombres:

- ✓ Son por defecto públicos y no se puede modificar su “visibilidad”
- ✓ Permite identificar clases, funciones y estructuras de datos que hagan referencia a un mismo tema.

8.4 Variables, constantes y otros conceptos relacionados

Una constante es algo que permanece inalterable, por eso se llama constante, porque siempre es constante: inalterable, siempre es lo mismo. Sin embargo una variable puede alterarse, es decir, se le puede cambiar el valor que tiene, por eso se llama variable.

Entonces se dice que las variables son "nombres" que pueden contener un valor, ya sea de tipo numérico como de cualquier otro tipo. Esos nombres son convenciones que se usan para facilitar el entendimiento de las cosas, ya que para las computadoras, una variable es una dirección de memoria en la que se guarda un valor o un objeto, como lo que sucede en .NET, todo es un objeto.

Existen distintos tipos de valores que se pueden asignar a una variable, por ejemplo, se puede tener un valor numérico o se puede tener un valor de tipo alfanumérico o de cadena (*string*), pero en cualquier caso, la forma de hacerlo siempre es de la misma forma, al menos en .NET ya no hay las distinciones que antes había en las versiones anteriores de Visual Basic. Por ejemplo si se quiere guardar el número 10 en una variable, se hace como sigue:

i = 10

En este caso **i** es la variable, mientras que **10** sería una constante, (10 siempre vale 10), la cual se asigna a esa "posición" de memoria a la que llamamos **i**, realmente no interesa saber dónde se guarda ese valor, lo único que interesa es saber que se guarda en algún lado para que en cualquier ocasión poder volver a usarlo.

También se pueden aplicar expresiones[1] al asignar una variable, una expresión es algo así como un cálculo que se quiere hacer, por ejemplo: **i = x * 25**, en este caso **x * 25** se dice que es una expresión, cuyo resultado, (el resultante de multiplicar lo que vale la variable x por la constante 25), se almacenará en la variable i. Si x vale 3, (es decir el valor de la variable x es tres), el resultado de multiplicarlo por 25, se guardará en la variable i, es decir i valdrá 75.

Pero no es suficiente saber qué es una variable, lo importante es saber cómo decirle al VB.NET que se quiere usar un espacio de memoria para almacenar un valor, ya sea numérico, de cadena o de cualquier otro tipo.

Las cadenas de caracteres (o valores alfanuméricos) se representan por algo que está contenido dentro de comillas dobles: "hola" sería una constante de cadena, ya que "hola" será siempre "hola", lo mismo que el número 10 siempre vale 10. Para asignar esa constante de caracteres a una variable, se haría algo como esto:

s = "Hola"

De esta forma, la variable **s** contiene el valor *constante* **"Hola"**. Se puede cambiar el valor de **s**, asignándole un nuevo valor: **s = "adiós"**, pero no se puede cambiar el valor de "Hola", ya que si éste valor se cambia dejará de ser "Hola" y se convertirá en otra cosa.

[1] Una expresión es una secuencia de operadores y operandos que describe un cálculo. Normalmente una expresión se evalúa en tiempo de ejecución. Existen expresiones numéricas y alfanuméricas o de caracteres.

Declaración de Variables

DIM

Como se ha dicho, existen distintos tipos de datos que VB.NET maneja, para que se pueda usar una variable para almacenar cualquiera de esos tipos, se tiene que decir al programa de VB que "reserve" espacio en la memoria para poder guardarlo. Esto se consigue mediante la "***declaración de variables***", es necesario, aunque no obligatorio, declarar las variables según el tipo de datos que va a almacenar. Por ejemplo, en el caso anterior, la variable **i** era de tipo numérico y la variable **s** era de tipo cadena. Esas variables habría que declararlas de la siguiente forma: (después se verán otras formas de declarar las variables numéricas)

```
Dim i As Integer
Dim s As String
```

Con esto se le está diciendo a VB.NET que reserve espacio en su memoria para guardar un valor de tipo ***Integer*** (***numérico***), en la variable **i** y que en la variable **s** se van a guardar valores de ***cadena de caracteres*** (***String***).

A continuación se van a ver cuales son los tipos de datos que .NET soporta. La siguiente tabla muestra algunos de ellos y los valores mínimos y máximos que puede contener, así como el tamaño que ocupa en memoria; también se comentan algunas otras cosas, que aunque ahora no parezcan "aclaratorias", en un futuro si lo serán:

Tipos de datos de Visual Basic.NET y su equivalente en el Common Language Runtime (CLR)

Tipo de Visual Basic	Tipo en CLR (Framework)	Espacio de memoria que ocupa	Valores que se pueden almacenar y comentarios
Boolean	System.Boolean	2 bytes	Un valor verdadero o falso. Valores: True o False. En VB se pueden representar por -1 o 0, en CLR serán 1 y 0, aunque no es recomendable usar valores numéricos, es preferible usar siempre True o False. Dim b As Boolean = True
Byte	System.Byte	1 byte	Un valor positivo, sin signo, para contener datos binarios. Valores: de 0 a 255 Puede convertirse a: Short, Integer, Long, Single, Double o Decimal sin recibir overflow Dim b As Byte = 129
Char	System.Char	2 bytes	Un carácter Unicode. Valores: de 0 a 65535 (sin signo). No se puede convertir directamente a tipo numérico. Para indicar que una constante de cadena, realmente es un Char, usar la letra C después de la cadena: Dim c As Char = "N"c
Date	System.DateTime	8 bytes	Una fecha. Valores: desde las 0:00:00 del 1 de Enero del 0001 hasta las 23:59:59 del 31 de Diciembre del 9999. Las fechas deben representarse entre almohadillas # y por lo habitual usando el formato norteamericano: #mm-dd-yyyy# Dim d As Date = #10-27-2001#
Decimal	System.Decimal	16 bytes	Un número decimal. Valores: de 0 a +/-79,228,162,514,264,337,593,543,950,335 sin decimales; de 0 a +/-7.9228162514264337593543950335 con 28 lugares a la derecha del decimal; el número más pequeño es:

			+/-0.00000000000000000000000001 (+/-1E-28). En los literales se puede usar la letra D o el signo @ para indicar que el valor es Decimal. Dim unDecimal As Decimal = 922337203685477580 8D Dim unDecimal2 As Decimal = 987654321.125@
Double	System.Double	8 bytes	Un número de coma flotante de doble precisión. Valores: de -1.7976931348623157E+308 a -4.94065645841246544E-324 para valores negativos; de 4.94065645841246544E-324 a 1.7976931348623157E+308 para valores positivos. Se puede convertir a Decimal sin recibir un overflow. Se puede usar como sufijo el signo almohadilla # o la letra R para representar un valor de doble precisión: Dim unDoble As Double = 125897.0235R Dim unDoble2 As Double = 987456.0125#
Integer	System.Int32	4 bytes	Un número entero (sin decimales) Valores: de -2,147,483,648 a 2,147,483,647. Se puede convertir a Long, Single, Double o Decimal sin producir overflow. Se puede usar la letra I o el signo % para indicar que es un número entero: Dim unEntero As Integer = 250009I Dim unEntero2 As Integer = 652000%
Long (entero largo)	System.Int64	8 bytes	Un entero largo (o grande) Valores: de -9,223,372,036,854,775,808 a 9,223,372,036,854,775,807. Se puede convertir a Single, Double o Decimal sin producir overflow. Se puede usar la letra L o el signo & para indicar que es un número Long: Dim unLong As Long = 12345678L Dim unLong2 As Long = 1234567890&
Object	System.Object (class)	4 bytes	Cualquier tipo se puede almacenar en una variable de tipo Object. Todos los datos que se manejan en .NET están basados en el tipo Object.
Short (entero corto)	System.Int16	2 bytes	Un entero corto (sin decimales) Valores: de -32,768 a 32,767. Se puede convertir a: Integer, Long, Single, Double o Decimal sin producir un overflow. Se puede usar la letra S para indicar que es un número entero corto: Dim unShort As Short = 32000S
Single	System.Single	4 bytes	Número de coma flotante de precisión simple. Valores: de -3.4028235E+38 a -1.401298E-45 para valores negativos; de 1.401298E-45 a 3.4028235E+38 para valores positivos. Se puede convertir a: Double o Decimal sin producir overflow. Se pueden usar la letra F y el símbolo ! para indicar que es un número Single: Dim unSingle As Single = 987.125F Dim unSingle2 As Single = 65478.6547!
String (cadenas de longitud variable)	System.String (clase)	Depende de la plataforma	Una cadena de caracteres Unicode. Valores: de 0 to aproximadamente 2 billones (2^31) de caracteres Unicode. Se puede usar el símbolo $ para indicar que una variable es un String.
Tipos definidos por el usuario (estructuras)	(Heredada de System.ValueType)	Depende de la plataforma	Cada miembro de la estructura tiene su rango, dependiendo del tipo de dato que representa.

En la tabla anterior se tienen los tipos de datos que se pueden usar en VB.NET y por tanto, de los que se pueden declarar variables.

Existe una instrucción, (Option Explicit), ahora viene puesta por defecto; la cual si que obliga a que se declaren las variables, pero si se quita esa instrucción, entonces se pueden cometer errores graves. La cuestión es que siempre se deben declarar las variables, e incluso más: **siempre se deben declarar las variables del tipo que quieres que dicha variable contenga**. Ya que, se pueden declarar variables sin tipo específico: **Dim unaVariable,** que en realidad es como si se la hubiese declarado del tipo ***Object***, (***As Object***), por tanto aceptará cualquier tipo de datos, pero esto no es una buena práctica. Entonces hay que declarar siempre el tipo de las variables.

Las variables se pueden declarar de dos formas, aunque básicamente es lo mismo:

1. Declarando la variable y dejando que VB asigne el valor por defecto.
2. Declarando la variable y asignándole el valor inicial que se quiera que tenga.

Por defecto, cuando no se asigna un valor a una variable, éstas contendrán los siguientes valores, dependiendo del tipo de datos que sea:

- ✓ Las variables numéricas tendrán un valor CERO.
- ✓ Las cadenas de caracteres una cadena vacía: ""
- ✓ Las variables Boolean un valor False (recuerda que False y CERO es lo mismo)
- ✓ Las variable de tipo Objeto tendrán un valor Nothing, es decir nada, un valor nulo.

Por ejemplo: **Dim i As Integer** tendrá un valor inicial de 0. Pero si se quiere que inicialmente valga 15, se puede hacer de cualquiera de estas dos formas:

1.) **Dim i As Integer**
i = 15

2.) **Dim i As Integer = 15**

CONST

Esta segunda forma es exclusiva de la versión **.NET** de Visual Basic, (también de otros lenguajes, pero es algo nuevo para los que ya han trabajado con antiguas versiones de VB). Mientras que la forma mostrada en el punto 1.) es la forma clásica, (la única que se puede usar en las versiones anteriores de VB). Las constantes se declaran de la misma forma que la indicada en el inciso 2.), ya que no se podrían declarar como lo mostrado en el punto 1.), por la sencilla razón de que a una constante no se le puede volver a asignar ningún otro valor, ya que sino, no serían constantes, sino variables.

Por ejemplo: Const n As Integer = 15

Para declarar una constante de tipo String, se hace de la siguiente forma:

Const s As String = "Hola"

De igual manera, para declarar una variable de tipo String y que contenga un valor, se hace de la siguiente forma:

Dim Nombre As String = "Javier"

Es decir, en las variables se usará la palabra ***DIM***, mientras que en las constantes se usará ***CONST***. Se pueden usar cualquier constante o variable en las expresiones, e incluso, se puede usar el resultado de esa expresión para asignar un valor a una variable. Por ejemplo:

```
Dim x As Integer = 25
Dim i As Integer
i = x * 2
```

En este caso, se evalúa el resultado de la expresión, (lo que hay a la derecha del signo igual), y el resultado de la misma, se asigna a la variable que estará a la izquierda del signo igual. Incluso se pueden hacer expresiones como la que sigue:

```
i = i + 15
```

Con esto, se está indicando a VB que: calcula lo que actualmente vale la variable i, se suma el valor 15 y el resultado de esa suma se guarda en la variable i. Por tanto, suponiendo que i valiese 50, después de esta asignación, su valor será 65, (es decir 50 que valía antes más 15 que le ha sumado). Esto último se llama incrementar una variable, y VB.NET tiene su propio operador para estos casos, es decir cuando lo que se asigna a una variable es lo que ya había antes más el resultado de una expresión:

```
i += 15
```

Aunque también se pueden usar: *=, /=, -=, etcétera, dependiendo de la operación que se dice hacer con el valor que ya tuviera la variable.

Por tanto **i = i * 2**, es lo mismo que **i *= 2**

Se puede usar cualquier tipo de expresión, siempre y cuando el resultado esté dentro de los soportados por esa variable:

```
i += 25 + (n * 2)
```

Es decir, no se puede asignar a una variable de tipo numérico el resultado de una expresión alfanumérica:

```
i += "10 * 25"
```

Ya que "10 * 25" es una constante de tipo cadena (string), no una expresión que multiplica 10 por 25. Al estar entre comillas dobles se convierte *automáticamente* en una constante de cadena y deja de ser una expresión numérica. Y si se tiene activo **Option Stric On**, tampoco se podrían usar números que no fuesen del tipo Integer:

```
i += 25 * 3.1416
```

Ya que VB mostraría un mensaje de error, aunque para solventar estos inconvenientes existen unas funciones de conversión, que sirven para pasar datos de un tipo a otro. No se va a profundizar, pero para que se entienda que se está haciendo, lo que esté escrito dentro de comillas dobles o esté contenido en una variable de cadena no se evalúa, lo más que se podría hacer sería convertir esa cadena en un valor numérico, en el caso de "10 * 25", el resultado de convertirlo en valor numérico será 10, ya que todo lo que hay después del 10, no se evalúa, simplemente porque no es un número son letras, que tienen el "aspecto" de operadores, pero que no es el operador de multiplicar, sino el símbolo *.

Por tanto, esto: **i = Val("10 * 25"),** es lo mismo que esto otro: **i = Val("10")**

En este caso, se usa la función **Val** para convertir una cadena en un número, pero ese número es del tipo ***Double*** y si se tiene ***Option Strict On***, no dejará convertirlo en un ***Integer***, así de "estricto" es el Option Strict.

Para solucionarlo, se usará la función CType:

i = CType(Val("10 * 25"), Integer)

Con esto se está diciendo VB que primero convierta la cadena en un número mediante la función ***Val***, (que devuelve un número de tipo Double), después se le dice que ese número Double lo convierta en un valor Integer.

También se pdría hacer de esta otra forma:

i = CInt(Val("10 * 25"))

Pero cuidado con los valores que se evalúan, ya que si el valor que se quiere asignar no "cabe" en la variable a la que se asigna, dará un error de overflow, es decir que el número que se quiere asignar es más grande de los que ese tipo de datos puede soportar, para solucionar esto, habrá que usar un tipo de datos que soporte valores mayores. Por ejemplo:

i = CInt(Val("25987278547875"))

Dará error, porque el número ese que está dentro de las comillas es demasiado grande para almacenarlo en una variable de tipo *Intege*. En el Anexo 5 se muestra un resúmen de las distintas funciones de conversión de tipos y algunos ejemplos.

Cabe hacer hincapié en las dos últimas funciones, sobre todo si ya se ha usado anteriormente el Visual Basic e incluso el Basic de MS-DOS, ya que por tradición esas dos funciones devolvían valores enteros de tipo *Integer*. Ahora, a pesar de lo que la ayuda de VB.NET pueda decir, ya que puede depender de la interpretación que le den los autores, Int y Fix devuelve un valor del *mismo tipo* que el que se indica en el parámetro o expresión, pero sin decimales.

Declarar varias variables en una misma línea

Para poder declarar una variable en VB .Net, se utiliza la instrucción **Dim**. Por ejemplo, esta línea declara dos variables del tipo Integer:

Dim** a, b **As Integer O de la siguiente manera: ***Dim** c **As Integer**, d **As Integer***

Evidentemente en la segunda línea se indica dos veces el tipo de datos de cada una de las variables, mientras que en la primera sólo se especifica una vez. Cabe mencionar que se puede declarar más de una variable en una instrucción **DIM**, si las variables se separan con una coma. Es decir, hay que poner una coma entre cada variable que se declare en la misma línea y al final indicar el tipo de datos de las variables.

Declarar varios tipos de variables en una misma línea

Ésta no es la única forma de declarar varias variables en una misma línea, ya que puede ser que se quiera declarar variables de distintos tipos. Si ese es el caso, hay que indicar junto a cada variable el tipo de datos que se desea que tenga. Por ejemplo:

Dim** i **As Integer**, s **As String

En este caso, se tienen dos variables de dos tipos distintos, cada una con su ***As tipo*** correspondiente, pero separadas por una coma. Se pueden hacer declaraciones de variables de diferente tipo como se muestra a continuación:

Dim j, k As Integer, s1, Nombre As String, d1 As Decimal

En esta ocasión, las variables **j** y **k** son del tipo **Integer**, las variables **s1** y **Nombre** del tipo **String** y por último la variable **d1** es de tipo **Decimal**.

Tipo de dato por defecto de las variables

En el caso de las versiones de MS-DOS y el VB1, el tipo de datos era Single, en las versiones anteriores a VB.NET el tipo de datos predeterminado era Variant. En VB.NET, el tipo de datos sería Object. Por ejemplo:

***Dim** z*

El tipo de datos de la variable **z** sería **Object**, es decir, seria lo mismo que hacer los siguiente:

Dim** z **As Object

Una cosa que no está permitida al declarar varias variables usando sólo un *As Tipo*, es la asignación de un valor predeterminado. Como se vió anteriormente se puede hacer lo siguiente para asignar el valor 15 a la variable N:

***Dim** N **As Integer** = 15*

La visibilidad (o alcance) de las variables

Si se ha creado ya en Visual Studio algún proyecto, por ejemplo una aplicación de consola. El código que se crea es el siguiente:

```
Module Module1
    Sub Main()

    End Sub
End Module
```

Es decir, se crea un módulo llamado Module1 y un procedimiento llamado Sub Main, que por otro lado es el que sirve como punto de entrada al programa. Los procedimientos Sub son instrucciones y cuando se usan en otras partes del programa, se ejecuta el código que haya en su interior.

Lo que interesa es saber que hay dos "espacios" diferentes en los que se puede insertar las declaraciones de las variables:

Después de Module Module1 o después de Sub Main.

El código insertado entre **Module** y **End Module**, se dice que es código a nivel de módulo, en este caso el único código posible es el de declaraciones de variables o procedimientos. El código insertado dentro del **Sub** y **End Sub**, se dice que es código a nivel de procedimiento, en que se inserta código, además de declaraciones de variables.

Si se declara una variable a nivel de módulo, dicha variable estará disponible en "todo" el módulo, incluido los procedimientos que pudiera haber dentro del módulo. Por otro lado, las

variables declaradas dentro de un procedimiento, sólo serán **visibles** dentro de ese procedimiento, es decir que fuera del procedimiento no se tiene conocimiento de la existencia de dichas variables. Por ejemplo:

```
Module Module1
    ' Variable declarada a nivel de módulo
    Dim n As Integer = 15

    Sub Main()
        ' Variable declarada a nivel de procedimiento
        Dim i As Long = 10
        '
         ' Esto mostrará que n vale 15
        Console.WriteLine("El valor de n es: {0}", n)
        Console.WriteLine("El valor de i es: {0}", i)

        Console.ReadLine()
    End Sub

    Sub Prueba()
         ' Esto mostrará que n vale 15
        Console.WriteLine("El valor de n es: {0}", n)

        ' Esta línea dará error, ya que la variable i no está declarada
        Console.WriteLine("El valor de i es: {0}", i)

        Console.ReadLine()
    End Sub
End Module
```

Este módulo contiene dos procedimientos de tipo Sub, (los SUBs son como las instrucciones, realizan una tarea, pero no devuelven un valor como lo harían las funciones y las propiedades) La variable se ha declarado a nivel de módulo, por tanto estará visible en todo el módulo y por todos los procedimientos de ese módulo. Dentro del procedimiento **Main**, se declara la variable **i**, ésta sólo estará disponible dentro de ese procedimiento. Por eso al intentar usarla en el procedimiento **Prueba**, el VB.NET manda un error diciendo que la variable no está declarada. Y a pesar de que está declarada, lo está pero sólo para lo que haya dentro del procedimiento Main, por tanto en Prueba no se sabe que existe una variable llamada **i**.

8.5 Prioridad de los operadores

Los operadores pueden ser aritméticos, de comparación y lógicos. A continuación se muestra de manera ordenada la prioridad de los diferentes operadores:

Operadores Aritméticos y de Concatenación:

Exponenciación (^)
Negación (-)
Multiplicación y división (*, /)
División de números enteros (\)
Módulo aritmético (Mod)
Suma y resta (+, -)
Concatenación de cadenas (&)

Operadores de comparación:

Igualdad (=)
Desigualdad (<>)
Menor o mayor que (<, >)
Mayor o igual que (>=)
Menor o igual que (<=)

Operadores lógicos

Negación (Not)
Conjunción (And, AndAlso)
Disyunción (Or, OrElse, Xor)

Para terminar, cabe decir que las expresiones que se pueden usar con el IF pueden ser tanto numéricas como alfanuméricas o de cadena, ya que, por ejemplo, también puede ser conveniente saber si el contendido de una cadena ***s*** es mayor que la palabra "hola", aunque en este tipo de expresiones, se evalúa tal y como si se fuese a clasificar, es decir "ahora" será menor que "hola", ya que si se clasificáramos, tendríamos que la letra A está antes que la H.

8.6 Expresiones Lógicas

Con las expresiones lógicas se van a evaluar expresiones cuyo resultado pueda ser un valor verdadero o falso. Hay ocasiones en las que se necesitará decidir que hacer dependiendo de algún condicionante, por ejemplo, si se tiene que comprobar si la tecla que se había pulsado era la tecla Suprimir y si es así, por ejemplo, eliminar elementos de un ListBox.

If…. Then

If *<expresión a evaluar>* ***Then*** *<Lo que haya que hacer si la expresión evaluada devuelve Verdadero>*

Esta es la forma más simple, ya que aquí lo que se hace es evaluar la expresión que se indica después de ***IF*** y si esa expresión devuelve un valor verdadero (es decir es verdad), se ejecutan los comandos que haya después de ***THEN*** y si esa expresión no es cierta, se ejecuta lo que haya en la siguiente línea.

Eso mismo también se suele usar de esta otra forma:

If *<expresión a evaluar>* ***Then***
<Lo que haya que hacer si la expresión evaluada devuelve Verdadero>
End If

Que para el caso es lo mismo, con la diferencia de que resulta más claro de leer y que podemos usar más de una línea de código, con lo cual resulta más evidente el que se puedan hacer más cosas.

Else

Pero si también se quiere hacer hacer algo cuando la expresión ***NO*** se cumpla, se puede usar la palabra ***ELSE*** y a continuación el código que queremos usar cuando la expresión no se cumpla.

If *<expresión a evaluar>* ***Then*** *<Lo que haya que hacer si la expresión evaluada devuelve Verdadero>* ***Else*** *<Lo que haya que hacer si no se cumple> (todo en una misma línea)*

O una mejor forma, que además queda más claro y evidente lo que se quiere hacer:

```
If <expresión a evaluar> Then
    <Lo que haya que hacer si la expresión evaluada devuelve Verdadero>
Else
    <Lo que haya que hacer si no se cumple>
End If
```

Después de ***Else*** se puede usar otro ***IF*** si así se cree conveniente, esto es útil cuando se queire comprobar más de una cosa y dependiendo del valor, hacer una cosa u otra:

```
If a = 10 Then
    ' Lo que sea que haya que hacer cuando a vale 10
ElseIf a = 15 Then
    ' Lo que haya que hacer cuando a vale 15
Else
    ' Lo que haya que hacer en caso de que a no valga ni 10 ni 15
End If
' Esto se ejecuta siempre después de haberse comprobado todo lo anterior.
```

Fijarse uno que enseguida de cada ***If*** /***Then*** se ha usado lo que se llama un ***comentario***. Los comentarios empiezan por una comilla simple (apóstrofe), en los comentarios se puede poner lo que se quiera, con la seguridad de que no será tenido en cuenta por el programa de Visual Basic. Los comentarios sólo pueden ocupar una línea, salvo que dicha línea al final tenga el signo _ (subrayado bajo), lo cual indica al IDE que se quiere continuar en la siguiente línea. Ese símbolo se puede llamar "continuador de línea" y se puede usar siempre que se quiera, no sólo para los comentarios. Los comentarios también se pueden hacer con la palabra reservada Rem, pero eso es algo que ya nadie usa.

8.7 Bucles en Visual Basic .NET

Cada vez que se quiera "reiterar" o repetir un mismo código un número determinado de veces, e incluso un número indeterminado de veces, se tiene que usar los denominados ‘bucles’.

En Visual Basic .NET existen diferente instrucciones para hacer bucles (o reiteraciones), veamos esas instrucciones y después se verán con más detalle:

For / Next

For / Next, con este tipo de bucle se puede repetir un código un número determinado de veces.

La forma de usarlo sería:

```
For <variable numérica> = <valor inicial> To <valor final> [Step <incremento>]
    ' contenido del bucle, lo que se va a repetir
Next
```

La *variable numérica* tomará valores que van desde el *valor inicial* hasta el *valor final*, sino se especifica el valor del *incremento*, éste será 1. Pero si la intención es que el valor del incremento sea diferente a 1, habrá que indicar un valor de incremento; lo mismo se tiene que hacer si se desea que el valor inicial sea mayor que el final, con idea de que "cuente" de mayor a

menor, aunque en este caso el incremento en realidad será un "decremento" ya que el valor de incremento será negativo. Se entederá mejor con algunos ejemplos:

```
Dim i As Integer
'
For i = 1 To 10
    ' contará de 1 hasta 10
    ' la variable i tomará los valores 1, 2, 3, etc.
Next
'
For i = 1 To 100 Step 2
    ' contará desde 1 hasta 100 (realmente 99) de 2 en 2
    ' la variable i tomará los valores 1, 3, 5, etc.
Next
'
For i = 10 To 1 Step -1
    ' contará desde 10 hasta 1
    ' la variable i tomará los valores 10, 9, 8, etc.
Next
'
For i = 100 To 1 Step -10
    ' contará desde 100 hasta 1, (realmente hasta 10)
    ' la variable i tomará los valores 100, 90, 80, etc.
Next
'
For i = 10 To 1
    ' este bucle no se repetirá ninguna vez
Next
'
For i = 1 To 20 Step 50
    ' esto sólo se repetirá una vez
Next
```

En algunos casos, hay que tener en cuenta que el valor final del bucle puede que no sea el indicado, todo dependerá del incremento que se haya especificado. Por ejemplo, en el tercer bucle, se le indica que cuente desde 1 hasta 100 de dos en dos, el valor final será 99. En otros casos puede incluso que no se repita ninguna vez, éste es el caso del penúltimo bucle, ya que se le indica que cuente de 10 a 1, pero al no indicar Step con un valor diferente, Visual Basic "supone" que será 1 y en cuanto le suma uno a 10, se da cuenta de que 11 es mayor que 1 y como se dijo que se quiere contar desde 10 hasta 1, pues sabe que no debe continuar.

For Each

For Each, este bucle repetirá o iterará por cada uno de los elementos contenidos en una colección. La forma de usarlo es:

For Each *<variable>* ***In*** *<colección del tipo de la variable>*
    ' lo que se hará mientras se repita el bucle
Next

Se puede usar este tipo de bucle para recorrer cada uno de los caracteres de una cadena, que a continuacion se muestra:

```
Dim s As String
'
For Each s In "Hola Mundo"
    Console.WriteLine(s)
Next
Console.ReadLine()
```

While / End While

While / End While, se repetirá mientras se cumpla la expresión lógica que se indicará después de While. La forma de usarlo es:

While <expresión>
    ' lo que haya que hacer mientras se cumpla la expresión
End While

Con este tipo de bucles, se evalúa la expresión y si el resultado es un valor verdadero, se ejecutará el código que esté dentro del bucle, es decir, entre While y End While. La expresión será una expresión lógica que se evaluará para conseguir un valor verdadero o falso. Por ejemplo:

```
Dim i As Integer
'
While i < 10
    Console.WriteLine(i)
    i = i + 1
End While
Console.ReadLine()
'
'
Dim n As Integer = 3
i = 1
While i = 10 * n
    ' no se repetirá ninguna vez
End While
```

En el primer caso, el bucle se repetirá mientras i sea menor que 10, fíjese que el valor de i se incrementa después de mostrarlo, por tanto se mostrarán los valores desde 0 hasta 9, ya que cuando i vale 10, no se cumple la condición y se sale del bucle. En el segundo ejemplo no se repetirá ninguna vez, ya que la condición es que i sea igual a 10 multiplicado por n, cosa que no ocurre, ya que i vale 1 y n vale 3 y como sabemos 1 no es igual a 30.

Do / Loop

Do / Loop, este tipo de bucle es muy parecido al anterior, aunque algo más flexible. Si se utiliza sólo con esas dos instrucciones, este tipo de bucle no acabará nunca y repetirá todo lo que haya entre Do y Loop.

Pero la flexibilidad a la se refiere con este tipo de bucle se puede, es que se puede usar con dos instrucciones que permitirán evaluar expresiones lógicas: ***While*** y ***Until***.

Pero no hay que confundir este While con el While/End While que se acaba de ver anteriormente, aunque la forma de usarlo es prácticamente como se acaba de ver. La ventaja de usar While o Until con los bucles Do/Loop es que estas dos instrucciones se pueden usar tanto junto a Do como junto a Loop, la diferencia está en que si se usan con Do, la evaluación se hará antes de empezar el bucle, mientras que si se usan con Loop, la evaluación se hará después de que el bucle se repita al menos una vez. A continuación se muestra cómo usar este tipo de bucle con las dos "variantes":

Do While *<expresión>*
    '
Loop

Do
    '
Loop While *<expresión>*

```
Do Until <expresión>
    '
Loop

Do
    '
Loop Until <expresión>
```

Usando Do con While no se diferencia en nada con lo que se vió anteriormente, con la excepción de que se use junto a Loop, pero como se ha comentado antes, en ese caso la evaluación de la expresión se hace después de que se repita como mínimo una vez.

Until, a diferencia de While, la expresión se evalúa cuando no se cumple la condición, es como si se negara la expresión con While (Not <expresión>). Una expresión usadas con Until se podría leer de la siguiente forma: hasta que la expresión se cumpla:

Do Until X > 10 (repite hasta que X sea mayor que 10)

Por ejemplo:

```
i = 0
Do Until i > 9
   Console.WriteLine(i)
   i = i + 1
Loop
```

Este bucle se repetirá para valores de i desde 0 hasta 9 (ambos inclusive). Y este también:

```
i = 0
Do While Not (i > 9)
   Console.WriteLine(i)
   i = i + 1
Loop
```

Con esto se entederá mejor cuando de refiere uno a negar la expresión con While. En el caso de que Until se use junto a Loop, la expresión se comprobará después de repetir al menos una vez el bucle.

Cuando se está dentro de un bucle, se puede abandonarlo de dos formas:

1- Esperando a que el bucle termine.
2- Saliendo "abruptamente" o abandonándolo antes de que termine de forma lógica o por los valores que se le han indicado (al menos en el caso de los bucles For).

Para poder abandonar un bucle, hay que usar la instrucción Exit seguida del tipo de bucle que se quiere abandonar:

- ✓ Exit For
- ✓ Exit While
- ✓ Exit Do

Esto es útil si se necesita abandonar el bucle por medio de una condición, normalmente se utiliza con un If / Then.

Select Case

Hasta ahora se han usado las instrucciones IF / Then / Else, pero Visual Basic pone a disposición la instrucción **Select Case** con la que se pueude elegir entre varias opciones y en la versión .NET facilita un poco las cosas, al menos con respecto a como se usaba en las versiones anteriores de Visual Basic.

Select Case, se usa de la siguiente forma:

```
Select Case <expresión a evaluar>
Case <lista de expresiones>
    ' ...
Case <otra lista de expresiones>
    ' ...
Case Else
    ' si no se cumple ninguna de las listas de expresiones
End Select
```

Después de ***Select Case*** se pondrá una expresión a evaluar, es decir lo que se quiere comprobar si se cumple o no, esto es lo mismo que se hace con el **If** <expresión a evaluar> **Then**; lo que diferencia al **Select Case** del **If**... **Then** es que con el **If... Then** si se quieren hacer varias comprobaciones, se tendrán que usar diferentes **If... Then** con varios **ElseIf...**, mientras que con el ***Select Case***, cada una de las cosas que se desean comprobar se ponen en los distintos **Case...** Poe ejemplo, supongase que se quiere comprobar si el contenido de la variable **i** tiene distintos valores y según esos valores tendremos que hacer una cosa u otra. Si se hiciéra usando **If... Then**, se tendría que escribir algo como esto:

```
If i = 3 Then
    '
ElseIf i > 5 AndAlso i < 12 Then
    '
ElseIf i = 14 OrElse i = 17 Then
    '
ElseIf i > 25 Then
    '
Else
    '
End If
```

Esto mismo, con Select Case se haría de esta forma:

```
Select Case i
    Case 3
        '
    Case 6 To 11
        '
    Case 14, 17
        '
    Case Is > 25
        '
    Case Else
        '
End Select
```

Como se puede comprobar, después de Select Case se escribe lo que se quiere tener en cuenta, en este caso es el contenido de la variable i, cuando se quiere comprobar si es igual a 3, simplemente se pone el 3, lo mismo se hace cuando se quiere hacer algo para el caso de que el valor sea 14 o 17, pero en este caso simplemente se especifican los valores que se quieren comprobar separados por comas, se pueden poner tantos como necesitemos.

Como se observa, funciona casi igual que con If... Then, pero de una forma algo más ordenada, la única limitante es que sólo se puede evaluar una expresión, mientras que con If se pueden usar tantas como se quieran, precisamente por eso existen estas dos posibilidades, cuando una no es válida, podemos usar la otra.

Además de expresiones simples, en Select Case se puede indicar cualquier expresión válida, siempre que el resultado de esa expresión sea comparable con un valor, el cual, a su vez, producirá un valor verdadero o falso. Estas expresiones, pueden ser tanto numéricas como de cadenas de caracteres.

8.8 ERRORES

Cuando en el código de una aplicación se produce un error sintáctico, es decir, porque se haya escrito mal alguna instrucción de Visual Basic .NET, será el propio entorno de desarrollo el que avise de que hay algo que no es correcto; a este tipo de errores se suele llamar ***errores sintáctico o en tiempo de diseño***. Pero si lo que ocurre es que se ha asignado un valor erróneo a una variable o se ha realizado una división por cero o se está intentado accesar a un archivo que no existe, entonces, se producirá un error en tiempo de ejecución, es decir sólo se sabrá que hay algo mal cuando el archivo ejecutable esté funcionando.

Control Estructurado de Errores

Try

El método recomendado de capturar errores en Visual Basic .NET, es usando la estructura Try Catch Finally. La forma de usar esta estructura será de la siguiente forma:

```
Try
    ' el código que puede producir error
Catch [tipo de error a capturar]
    ' código cuando se produzca un error
Finally
    ' código se produzca o no un error
End Try
```

En el bloque **Try** se pondrá el código que puede que produzca un error. Los bloques **Catch y Finally** no son los dos obligatorios, pero al menos hay que usar uno de ellos, es decir o se usa ***Catch*** o ***Finally*** o ambos, pero como mínimo uno de ellos.

Cath

Si se usa ***Catch***, esa parte se ejecutará si se produce un error, es la parte que "capturará" el error.

Después de Catch podemos indicar el tipo de error que queremos capturar, incluso podemos usar más de un bloque Catch, si es que nuestra intención es detectar diferentes tipos de errores.

En el caso de que sólo se use ***Finally***, se tiene que tener en cuenta de que si se produce un error, el programa se detendrá de la misma forma que si no hubiésemos usado la detección de errores, por tanto, aunque se use Finally (y no se esté obligado a usar Catch), es más que recomendable que siempre se use una cláusula Catch, aunque en ese bloque no se haga nada, pero aunque no "se trate" correctamente el error, al menos no se detendrá porque se haya producido el error. Si se decide "prevenir" que se produzca un error, pero simplemente se quiere

que el programa continúe su ejecución, se puede usar un bloque Catch que esté vacío, con lo cual el error simplemente se ignorará.

Si se produce una excepción o error en un bloque Try, el código que siga a la línea que ha producido el error, deja de ejecutarse, para pasar a ejecutar el código que haya en el bloque Catch, se seguiría (si lo hubiera) por el bloque Finally y por último con lo que haya después de End Try.

Si no se produce ningún error, se continuaría con todo el código que haya en el bloque Try y después se seguiría, (si lo hubiera), con el bloque Finally y por último con lo que haya después de End Try.

8.9 Tipos de Datos por Valor

Cuando se dimensiona una variable, por ejemplo, de tipo Integer, el CLR reserva el espacio de memoria que necesita para almacenar un valor del tipo indicado; cuando a esa variable se le asigna un valor, el CLR almacena dicho valor en la posición de memoria que reservó para esa variable. Si posteriormente se le asigna un nuevo valor a dicha variable, ese valor se almacena también en la memoria, sustituyendo el anterior; así es como funcionan las variables. Si a continuación se crea una segunda variable y se le asigna el valor que contenía la primera, se tendrán dos posiciones de memoria con dos valores, que por pura casualidad, resulta que son iguales, entre otras cosas porque a la segunda variable se le ha asignado un valor "igual" al que tenía la primera.

```
' Ejemplo de Ipos de Datos por Valor
Sub Main()
    ' Se crea una variable de tipo Integer
    Dim i As Integer
    ' Se le asigna un valor
    i = 15
    ' Se muestra el valor de i
    Console.WriteLine("i vale {0}", i)
    '
    ' se crea otra variable
    Dim j As Integer
    ' Se le asigna el valor de i
    j = i
    Console.WriteLine("se hace esta asignación: j = i")
    '
    ' Se muestra cuanto contienen las variables
    Console.WriteLine("i vale {0} y j vale {1}", i, j)
    '
    ' Ahora se cambia el valor de i
    i = 25
    Console.WriteLine("se hace esta asignación: i = 25")
    ' se muestran nuevamente los valores
    Console.WriteLine("i vale {0} y j vale {1}", i, j)
    '
    Console.WriteLine("Pulse Enter para finalizar")
    Console.ReadLine()
End Sub
```

Como se puede comprobar, cada variable tiene un valor independiente del otro.

8.10 Tipos de Datos por Referencia

En Visual Basic .NET también se pueden crear objetos, los objetos se crean a partir de una clase. Cuando se dimensiona una variable cuyo tipo es una clase, simplemente se está creando una variable que es capaz de manipular un objeto de ese tipo, en el momento de la declaración, el CLR no reserva espacio para el objeto que contendrá esa variable, ya que esto sólo lo hace cuando se usa la instrucción ***New***. En el momento en que se crea el objeto, (mediante New), es cuando el CLR reserva la memoria para dicho objeto y le dice a la variable en que parte de la memoria está almacenado, de forma que la variable pueda accesar al objeto que se ha creado. Si posteriormente se declara otra variable del mismo tipo que la primera, se tendrán dos variables que saben "manejar" datos de ese tipo, pero si a la segunda variable se le asigna el contenido de la primera, en la memoria no existirán dos copias de ese objeto, sólo existirá un objeto que estará referenciado por dos variables. Por lo tanto, cualquier cambio que se haga en dicho objeto se reflejará en ambas variables. Para el siguiente ejemplo, se va a crear una clase con una sola propiedad, ya que las clases a diferencia de los tipos por valor, deben tener propiedades a las que asignarles algún valor:

```
' Ejemplo de Tipos de Datos por Referencia
Class prueba
    Public Nombre As String
End Class

Sub Main()
    ' Se crea una variable de tipo prueba
    Dim a As prueba
    ' Se crea (o instancía) el objeto en memoria
    a = New prueba()
    ' Se le asigna un valor
    a.Nombre = "hola"
    ' Se muestra el contenido de a
    Console.WriteLine("a vale {0}", a.Nombre)
    '
    ' Se dimensiona otra variable
    Dim b As prueba
    ' Se asigna a la nueva variable el valor de a
    b = a
    Console.WriteLine("Se hace esta asignación: b = a")
    '
    ' Se muestra el contenido de las dos
    Console.WriteLine("a vale {0} y b vale {1}", a.Nombre, b.Nombre)
    ' Ahora se cambia el valor de la anterior
    a.Nombre = "adios"
    '
    Console.WriteLine("hacemos una nueva asignación a a.Nombre")
    '
    ' Se muestran nuevamente los valores
    Console.WriteLine("a vale {0} y b vale {1}", a.Nombre, b.Nombre)
    '
    Console.WriteLine("Pulsa Enter para finalizar")
    Console.ReadLine()
End Sub
```

La clase **prueba** es una clase muy simple; primero se dimensiona una variable de ese tipo y después se crea un nuevo objeto del tipo **prueba**, a la cual se le asigna a la variable **a**. Una vez que se tiene "instanciado" (o creado) el objeto al que hace referencia la variable **a**, se le asigna a la propiedad **Nombre** de dicho objeto un valor.Lo siguiente que se hace es declarar otra variable del tipo **prueba** y se le asigna lo que contiene la primera variable. La única diferencia es la forma de manipular las clases, ya que no se pueden usar "directamente", porque se tienen que crear (mediante New) y asignar el valor a una de las propiedades que dicha clase contenga.

Ahora, cuando se muestra el contenido de la propiedad **Nombre** de ambas variables, las dos muestran lo mismo, que es lo esperado, pero cuando se asigna un nuevo valor a la variable **a**, al volver a mostrar los valores de las dos variables, las dos siguen mostrando lo mismo; y esto no es lo que ocurría en el primer ejemplo. Esto sucende porque las dos variables, apuntan al mismo objeto que está creado en la memoria. Por lo tanto **las dos variables hacen referencia al mismo objeto** y cuando se realiza un cambio mediante una variable, ese cambio afecta también a la otra, por la sencilla razón que se está modificando el único objeto de ese tipo que hay creado en la memoria.

Para simplificar, los tipos por valor son los tipos de datos que para usarlos no se tiene que usar New, mientras que los tipos por referencia son tipos (clases) que para crear un nuevo objeto hay que usar la instrucción New. Los tipos por valor son los tipos "básicos" (o elementales), tales como Integer, Double, Decimal, Boolean, etcétera. Los tipos por referencia son todos los demás tipos.

8.11 Arreglos (Arrays)

Las variables que se han estado usando hasta ahora, eran de tipo escalar: sólo pueden contener un valor a la vez. Pero hay ocasiones en que surge la necesidad de querer tener en una misma variable, valores que de alguna forma están relacionados. Por ejemplo, si se tiene una colección de discos, se podría interesar tener esa discografía incluida en una misma variable para poder accesar a cualquiera de esos discos sin necesidad de tener que crear una variable distinta para cada uno de los discos, ya que sería totalmente ineficiente si, por ejemplo, se quisiéra imprimir una relación de los mismos.

Una de las formas en las que se pueden agrupar varios datos es mediante los arrays (o matrices). Usando un ***array***, se puede accesar a cualquiera de los valores que se tengan almacenados mediante un índice numérico. Por ejemplo, si se tiene la variable **discografía** y se queire accesar al tercer disco, se podría hacer de la siguiente forma: **discografía(3)**. Sabiendo esto, se puede comprobar que sería fácil recorrer el contenido de los arrays mediante un bucle For.

Declarar Variables como Arrays

Para poder indicarle a VB que se quiere crear un array se puede hacer de dos formas distintas, para este ejemplo se creará un array de tipo Integer:

1- La clásica (la usada en versiones anteriores)

Dim** a() **As Integer

2- La nueva forma introducida en .NET:

Dim** a **As Integer()

De cualquiera de estas dos formas se estaría creando un array de tipo Integer llamada a. Cuando se declara una variable de esta forma, sólo se le está indicando a VB que se quiere que la variable a crear a sea un array de tipo Integer, pero ese array no tiene reservado ningún espacio de memoria.

Asignar Valores a un Array

Para asignar un valor a un elemento de un array, se hace de la misma forma que con las variables normales, pero indicando el ***índice*** (o posición) en el que guardará el valor.

Por ejemplo, para almacenar el valor 15 en la posición 3 del array a, se haría lo siguiente:

a(3) = 15

Accesar a un Elemento de un Array

Si se quiere utilizar un elemento que está en la posición 3 para una operación, se puede hacer como con el resto de las variables, pero siempre usando el paréntesis y el número de elemento al que se quiere accesar:

*i = b * a(3)*

El índice para poder accesar al elemento del array puede ser cualquier expresión numérica, una variable o como se ha visto en los ejemplos, una constante. La única condición es que el resultado sea un número entero. Por lo tanto, se podría accesar a la posición indicada entre paréntesis, siempre que el resultado sea un valor entero y, por supuesto, esté dentro del rango que se haya dado al declarar el array:

x = a(i + k)

Nota aclaratoria:
Cuando se dice que el elemento está en la posición 3, no se hace referencia al tercer elemento del array, ya que si un array empieza por el elemento 0, el tercer elemento será el que esté en la posición 2, esto se debe a que el primer elemento será el que ocupe la posición cero.

Saber el Tamaño de un Array

Cuando se tiene un array declarado y asignado, se puede accesar a los elementos de ese array mediante un índice, como ya se ha visto anteriormente; pero si no se quiere "pasarse" cuando se quiera accesar a esos elementos, será de utilidad saber cuantos elementos tiene el array, para ello se puede usar la propiedad Length, la cual devuelve el número total de elementos, por tanto, esos elementos estarán comprendidos entre 0 y Length - 1. Por ejemplo, ésto es útil si se quiere accesar mediante un bucle For, en el siguiente código se mostrarían todos los elementos del array a:

```
For i = 0 To a.Length - 1
    Console.WriteLine(a(i))
Next
```

Inicializar un Array al Declararlo

Al igual que las variables normales se pueden declarar y al mismo tiempo asignarle un valor inicial, con los arrays también se puede hacer, pero de una forma diferente, ya que no es lo mismo asignar un valor que varios. Aunque hay que tener presente que si se inicializa un array al declararlo, no se puede indicar el número de elementos que tendrá, ya que el número de elementos estará supeditado a los valores asignados.

Para inicializar un array se deben declarar ese array sin indicar el número de elementos que contendrá, seguida de un signo igual y a continuación los valores encerrados en llaves. Por ejemplo:

```
Dim a() As Integer = {1, 42, 15, 90, 2}
```

También se puede hacerlo de esta otra forma:

```
Dim a As Integer() = {1, 42, 15, 90, 2}
```

Usando cualquiera de estas dos formas mostradas, el número de elementos será 5, por tanto los índices irán desde 0 hasta 4.

Los arrays pueden ser de cualquier tipo

En los ejemplos se están usando valores de tipo Integer, pero se puede hacer lo mismo si fuesen de tipo String o cualquier otro. En el caso de que sean datos de tipo String, los valores a asignar deberán estar entre comillas dobles o ser variables de tipo String. Por ejemplo:

```
Dim s As String() = {"Hola", "Mundo, ", "te", "saludo"}

s(3) = "saludamos"

Dim i As Integer
For i = 0 To s.Length - 1
    Console.WriteLine(s(i))
Next
```

Como se ve, hasta aquí reobserva un panorama general de la forma de trabajar con Visual Basic .Net, y con ello se puede tener como una alternativa de programación de manera individual y global de uso y manejo de información por medio de una interfaz que se adapte y por ende funcione para el o los usuarios a quien este dedicado el desarrollo del programa en este lenguaje.

CONCLUSIONES

Al comprender el Uso y Manejo del sistema Operativo Linux, la forma en que procesa la información, la estructura de la misma, el trabajo e interacción de los usuarios, sesiones, entre otras, hace posible que la administración de la información de cada uno de ellos se haga por separado permitiendo que haya una seguridad y confianza de tener en optimas condiciones tanto la info. de cada usuario. como de las aplicaciones que tengan. Sin afectar a los demás usuarios. como sucede con otros sistemas.

En la Administración de Servidores Web, en conjunto con la Seguridad en Cómputo, hacen que, al ofrecer un servicio por medio de Internet, se haga de manera sencilla, dinámica y sobre todo segura, para que el usuario pueda disfrutar de los beneficios que pueda ofrecer un Servidor para Web. Dicha seguridad se lograría al ofrecer al usuario mecanismos que permitan que el y solo él sea el que pueda acceder a un determinado tipo de información; a menos que se brinde una clave a determinados usuarios para que puedan acceder y utilizar a la información que el servidor brinde.

En la Creación de Paginas Web, para mostrar una interfaz amigable, funcional y dinámica al usuario., con la programación (en este caso, PHP) que sirve para dar mayor funcionalidad a la manera de trabajar la información que se tenga en las paginas Web y también por medio de este lenguaje poder interactuar con el manejador de Bases de Datos (MySQL), que dichas bases de datos son colecciones de información segmentadas y ordenadas en tablas, para que al buscar alguna información en ellas sea mas fácil su localización y gestión de la información.

En todo ello conjunto lleva al Desarrollo de Sistemas, Administración de los mismos, así como la robustez, soporte y extensibilidad que brindan los Sistemas basados y trabajados bajo ambiente Linux.

Es necesario recordar que no todo sistema que se desarrolla es del tipo de software libre o de código abierto (*open source*). Un ejemplo de ello se muestra con el software de tipo propietario, como es el caso de Visual Basic .Net, que es parte de una Suite de programas de Microsoft Visual Studio, y que sirve para elaborar interfaces de Aplicación en ambiente gráfico, ya sea para trabajar con éstas de manera local o de manera remota por medio de Internet.

Aplicando el enfoque Didáctico, se espera, con la experiencia obtenida a través de la adquisición y actualización de nuevas tecnologías, dar a conocer al usuario e interesarle acerca de estos tópicos, para que él por su cuenta se empiece a desarrollar y administrar también sistemas de computo.

BIBLIOGRAFÍA

Configuración de sistemas Linux
Daniel L. Morrill
Anaya Multimedia

Instalación y configuración de Linux
Patrick Voolkerding, Kevin Reichard, Eric Foster-Johnson
Prentice Hall

Linux. Guía de referencia y aprendizaje
Matt Welsh, Matthias Kalle Dalheimer, Lar Kaufman
O Reilly

Administración de Sistemas Operativos
Gómez J. Padilla
Editorial Ra-Ma

MySQL Reference Manual: Documentation from the Source
Autor Michael "Monty" Widenius, David Axmark
Editorial: O'Reilly

Php and Mysql for Dynamic Web Sites
Autor Larry E. (Larry Edward) Ullman
Editorial: O'Reilly

PHP and MySQL Web Development
Autor Luke Welling, Laura Thomson
Editorial: Sams Publishing

Web Database Applications with PHP & MySQL
Autor Hugh E. Williams, David (David John) Lane
Editorial: O'Reilly

MESOGRAFÍA

“Applied Cryptography: Protocols, Algorithms, and Source Code in C”
Bruce Schneier
“TCP/IP Illustrated Volume1: The protocols”
W. Richard Stevens
“The Shellcoder’s Handbook”
Dave Aitel, Sinan Eren, Jack Koziel et al
“Network Security Perimeter”
Stephen Northcutt, Lenny Zeltser et al
“Network Intrusion Detection: An Analyst’s Handbook”
Stephen Northcutt, Judy Novak
“Intrusion Signatures and Analysis”
Stephen Northcutt, Mark Cooper et al
“Mastering Network Security”
Chris Brenton
“Building Open Source Security Network Security Tools”
Mike D. Schiffman
Algunas ligas de interés
http://www.securityfocus.com
http://www.phrack.org
http://www.atstake.com
http://www.thc.org
http://www.coresecurity.com
http://www.securiteam.com
http://www.canal-ayuda.org/Seguridad/tipos_ataques.htm

ANEXOS

ANEXO 1

☞💻- Continuación de Redireccionamiento.

> → **Redireccionamiento Destructivo:** Es cuando se tiene un archivo con información, y al redireccionar la salida estándar a este archivo, la información previa se borra, quedando así sólo la información que se ha agregado recientemente.
>> → **Redireccionamiento No Destructivo:** Es cuando a un archivo existente se le direcciona una salida estándar, pero a diferencia del anterior, este archivo no pierde la información previa, sino que la acumula (almacena), de tal forma que la información siempre está presente. Hay que tomar en cuenta que la información que se contiene de ésta manera no sea de tamaño muy grande.
< → **Redireccionamiento de Entrada:** Es cuando existe un Archivo y se quiere ejecutar algún comando o acción con dicho archivo, por ejemplo:

```
javier@inventario:~$ wc < p11
 28  30 984
```

Se observa aquí que el contenido de ***p11*** lo analiza el comando ***wc*** (***Word Counter***). Así regresa el número de palabras que contiene dicho archivo.

>& → **Redireccionamiento de Error:** Es cuando se envia el contenido de una salida estándar junto con los mensajes de Error que pudiera generaR dicha salida.
| → **Redireccionamiento de un programa a otro programa:** El carácter pipe (**|**) funciona como un filtro mandando la salida de un programa hacia la entrada de otro programa, enlazando la salida de un programa con la entrada de otro programa, por ejemplo:

```
ls –l | more
```

Aquí se está enlazando la salida del comando ***ls*** con el programa ***more***, el cual nos mostrará el resultado del comando ***ls*** por pantalla cada vez que presionemos la tecla espaciadora.

☞💻- Ejemplos del comando ***hostname:***

- ✓ ***hostname*** → Muestra el nombre que se tiene asignado a la máquina.
- ✓ ***hostname -i*** → Muestra la Dirección IP que se tiene asignada a la computadora.
- ✓ ***hostname -d*** → Muestra el Dominio en el que está trabajando la computadora.
- ✓ ***hostname -f*** → Muestra el Nombre de Dominio que se le tiene asignado a la computadora (FQDN, Fully Qualified Domain Name). Consiste del nombre asignado del hostname y el nombre de dominio DNS.

```
javier@inventario:~$ hostname
inventario
javier@inventario:~$ hostname -f
inventario.mascarones.unam.mx
javier@inventario:~$ hostname -d
mascarones.unam.mx
javier@inventario:~$ hostname -i
```

```
132.248.75.249
```

Para ***desplegar las 10 primeras líneas de un archivo*** se tiene el comando ***head***, seguido de éste se le agrega un atributo de ***–n***, donde ***n*** es el número de líneas que se quiere desplegar, en vez de las 10 líneas que muestra por ***default,*** del archivo en cuestión. Por ejemplo:

```
head –10 p12
```

Para ***desplegar las 10 últimas líneas de un archivo*** se tiene el comando ***tail,*** seguido de éste se le agrega un atributo de ***–n***, donde ***n*** es el número de líneas que se quiere desplegar, en vez de las 10 líneas que muestra por ***default,*** del archivo en cuestión. Por ejemplo:

```
tail –10 p12
```

☞💻**-** Ejemplos del comando ***uname***:

- ✓ ***uname*** → Muestra el Tipo de Sistema Operativo, que en éste caso es Linux.
- ✓ ***uname -r*** → Muestra la versión con la que esta trabajando actualmente el Kernell del Sistema.
- ✓ ***uname -m*** → Muestra el tipo de Arquitectura que maneja el sistema.
- ✓ ***uname -n*** → Muestra el nombre de Nodo que se tiene para conectarse a una red (*node network*)
- ✓ ***uname -a*** → Imprime toda la información referente al sistema.

☞💻**-** Continuación de comando **find.** Por ejemplo:

```
javier@inventario:~$ find -name sh -print
```

Donde:

-name → Indica la Opción de búsqueda de Nombre.

sh → Es el Archivo a Buscar.

-print → Para Desplegar la información del archivo.

☞💻**-** Continuación de Asignación de permisos, por método simbólico:

Para modificar los permisos se pone el indicador de los permisos a afectar ya sea para el usuario, el grupo o los otros, seguido del signo + si se quiere agregar el permiso o de – si se quiere quitar el permiso separando con una coma los bloques de permisos, al final se indica el nombre del archivo o directorio a afectar.

☞💻**-** Continuación de Asignación de permisos, por método octal:

De manera tal que si se desean poner todos los permisos para el usuario, permisos de lectura y ejecución para el grupo y ninguno para los otros el comando quedaría de la siguiente forma:

```
chmod 750 p12
```

Sustituyendo el número correspondiente para cada bloque de permisos al usuario, al grupo y a los otros.

☞💻**-** Cuando se necesita dividir o cortar un archivo se tiene el comando ***cut***, hay que definir el patrón de división, como pueden ser (**, . : ; / # | ** o cualquier signo de puntuación), y las columnas que se quieran afectar de dicho archivo.

```
cut -f 1,5 -d : /etc/passwd
```

En el ejemplo anterior se están cortando las filas 1 y 5 con el atributo **–f**; con **–d** se está especificando cuál será el patrón que deberá seguir la instrucción, que en este caso serán dos puntos (**:**); por último la ubicación y el nombre del archivo que se quiere afectar. Para dividir por caracteres, se utiliza el atributo **–c.** La sintaxis sería la siguiente:

```
cut -c 1,5 /etc/passwd
```

Véase, que no se está dando un patrón de división, solamente el de caracteres. O sea que, no se agregó **–d :**, que es el patrón de división del ejemplo anterior.

☞💻**-** Continuación del editor ***pico.*** Se describen brevemente los comandos básicos para su manejo:

- ✓ **Ctrl (^) + O** → Guardar archivo, si el archivo ya tiene nombre lo guarda automáticamente, si no, pedirá el nombre con el cual se desea guardar.
- ✓ **Ctrl + X** → Salir del editor automáticamente si no se han hecho modificaciones al archivo. En caso de haber hecho modificaciones se pide la confirmación para guardar el archivo, si se desea guardar el archivo, se permite la opción de guardarlo con otro nombre.
- ✓ **Ctrl + C** → Indica el número de línea del archivo en donde se encuentra posicionado el cursor.
- ✓ **Ctrl + Y** → Avanzar una página dentro del archivo
- ✓ **Ctrl + V** → Regresar una página dentro del archivo
- ✓ **Ctrl + W** → Hacer una búsqueda de cadenas de caracteres, por letra o por palabra, dentro del archivo.
- ✓ **Ctrl + K** → Cortar la línea donde se encuentra posicionado el cursor actualmente, la línea cortada la envía a un buffer de memoria, es posible cortar más de una línea para después pegarlas en otra parte del documento.
- ✓ **Ctrl + U** → Pegar las líneas que fueron cortadas con anterioridad.
- ✓ **Ctrl + R** → Insertar el contenido de otro archivo al final del archivo que se esta editando.

☞💻**-** Formas y los comandos del editor ***vi***:

- ✓ **a** → 1.- Se activa el modo de Edición en el archivo, 2.- En el modo de edición Insertar texto a la derecha de la posición del cursor.
- ✓ **ESC** → Con la tecla de *escape* se sale de la edición y se pasa a Línea de Comando.
- ✓ **: w** → Escribir texto.
- ✓ **: w *archivo*** → Poner nombre al archivo que se está editando.
- ✓ **: wq** → Guardar los cambios hechos al archivo y salir a Modo Línea de Comando.
- ✓ **: q** → Salir del archivo.
- ✓ **: q!** → Salir del archivo sin guardar cambios.
- ✓ **: i** → Insertar texto en la posición actual de Cursor.
- ✓ **:s /palabra1/palabra2/g** → Sustituir una palabra, pero el cursor debe estar el posicionado en la palabra.
- ✓ **o** u **O** → Insertar una línea (de texto).
- ✓ **0 (cero)** → Insertar un número.
- ✓ **x** → Eliminar un caracter a la vez.

- ✓ **dd** → Eliminar una línea completa.
- ✓ **#x** → Eliminar # (número) de caracteres simultáneamente.
- ✓ **#dd** → Eliminar # (número) de líneas simultáneamente.
- ✓ **:u** → Deshacer todas las modificaciones (registros de acciones) hechas al archivo.
- ✓ **Y** → Copiar una línea a partir de la posición actual del cursor.
- ✓ **#Y** → Copiar # (número) de líneas a partir de la posición actual del cursor.
- ✓ **p** → Pegar una línea, previamente hecha una copia.
- ✓ **/palabra** → Buscar una palabra.
- ✓ **:set number** → Mostrar las líneas numeradas.
- ✓ **:set nonumber** → No mostrar o esconder las números de líneas.

Para desplazarse a través del archivo con este editor se tienen las siguiente teclas:

- ✓ **k** → Desplazarse una línea atrás (Arriba).
- ✓ **j** → Desplazarse una línea después (Abajo).
- ✓ **l** → Desplazarse una posición a la Derecha.
- ✓ **h** → Desplazarse una posición a la Izquierda.
- ✓ **Ctrl + d** → Avanzar media página.
- ✓ **Ctrl + f** → Avanzar una página completa.
- ✓ **Ctrl + u** → Retroceder media página.
- ✓ **Ctrl + b** → Retroceder una página completa.
- ✓ **ZZ** → Guardar y salir del editor a la línea de comando automáticamente.

☞💻**-** Ejemplos de ***grep***:

```
grep ^O nombres.txt
```
→ Lista las Líneas que empiezan con O en el archivo *nombres.txt*. El circunflejo (**^**) da la opción a **grep** de búsqueda al inicio de una cadena.

```
grep z$ nombres.txt
```
→ **$** Indíca a **grep** que liste las cadenas de texto que terminen en z.

```
grep A nombres.txt
```
→ Lista las cadenas de Texto que contengan una letra A en cualquier parte de la línea.

```
grep '^A|z$' nombres.txt
```
→ La expresión no genera ningún resultado. Esto se debe a que es una **Expresión Regular Extendida.** Para ello es necesario usar el comando ***egrep***.

```
egrep '^A|z$' nombres.txt
```
→ Lista las líneas que empiezan en A y terminan en z.

Pero también hay ocasiones, en que se pueden usar ambos comandos, **grep** y **egrep**. Por ejemplo:

```
grep '^A..l' nombres.txt
egrep '^A..l' nombres.txt
```
→ Lista las líneas que empiezan con A y en el cuarto carácter se tiene una letra **l.**

☞💻**-** Comandos Básicos de Red:

- ✓ ***who*** → Muestra un listado sencillo de quien se encuentra conectado en el servidor, a que hora se conecto y que esta haciendo.
- ✓ **w** → Muestra un listado con detalles de los usuarios conectados en el servidor.
- ✓ **whoami** → Indica con que login (cuenta de usuario) se está registrado dentro del servidor, este comando es útil ya que es posible que un usuario se convierta en otro diferente utilizando el comando su.

- ✓ **finger** → Muestra información de los usuarios conectados en el servidor, entre ella el nombre real del usuario.
- ✓ **finger login** o **finger nombre** → Muestra información en específico del usuario que se está indicando, entre la que se encuentra el nombre real del usuario, directorio $HOME, última fecha de acceso al servidor, etc.
- ✓ **users** → Muestra sólo los nombres de los usuarios conectados al servidor.
- ✓ **ping** → Es una aplicación usada en Internet para determinar si está o no activa la conexión a una máquina específica, o para averiguar la factibilidad de alcanzar a otra máquina. Básicamente, Ping envía una pequeña serie de sencillos ***packets*** -paquetes de datos-, y si la máquina apuntada retorna dichos packets, entonces esa máquina es considerada activa y disponible. La expresión fue tomada del sonar, que manda un sonido similar a un 'ping', y el eco que regresa es monitoreado y analizado. Algunos programas Ping también muestran la ruta seguida por los packets, y el tiempo empleado en ella, lo que resulta útil para seguimiento de problemas y evaluación de velocidades de conexión

Para generar un intercambio de paquetes de información, prueba de conexión y, mensaje de petición ←→ respuesta se pueden hacer con ***ping***. Por ejemplo:

- ➢ ping 127.0.0.1 → Genera una petición de Conexión interna del propio sistema. A esto se le dice que es una conexión de ***Loop Back*** (Circuito Cerrado).
- ➢ ping localhost → Hace una petición de conexión al nombre que tiene el sistema en *Loop Back*, o sea, internamente.
- ➢ ping inventario → Hace una petición de conexión a una PC, que tiene como hostname *inventario*.
- ➢ ping 132.248.75.120 → Hace una petición de conexión a una Dirección IP.
- ➢ ping atena.mascarones.unam.mx → Genera una petición de conexión al Servidor '*atena*'.

- ✓ **telnet** → Es un programa para Internet basado en texto, usado para enlazarse a una máquina remota. Una vez conectada, la máquina propia se comporta como si el usuario estuviera realmente sentado frente a la otra, aun cuando se hallen en diferentes partes del mundo. La conexión por este medio no es segura.
- ✓ **ssh** → (**S**ecure **SH**ell) es el nombre de un protocolo y del programa que lo implementa. Este protocolo sirve para acceder a máquinas remotas a través de una red, de forma similar a como se hace con *telnet*. La diferencia principal es que ***SSH*** usa técnicas de cifrado (encriptado) que hacen que la información que viaja por el medio de comunicación vaya de manera no legible y ninguna tercera persona pueda descubrir el usuario y contraseña de la conexión ni lo que se escribe durante toda la sesión; aunque es posible atacar este tipo de sistemas por medio de ataques de *REPLAY* y manipular así la información entre destinos. Al igual que telnet, sólo permite conexiones tipo terminal de texto, aunque puede redirigir el tráfico de X para poder ejecutar programas gráficos si se tiene un Servidor X arrancado.
- ✓ **write** → Este comando nos permite enviar un mensaje a un usuario, siempre y cuando el usuario se encuentre conectado en el servidor.
- ✓ **talk** → Este programa permite realizar una conversación interactiva en tiempo real, es posible mantener una conversación con más de dos usuarios al mismo tiempo, es necesario que los usuarios se encuentren conectados en el mismo servidor.
- ✓ **ytalk** → Este comando tiene funciones muy parecidas a talk, sólo que éste permite realizar una solicitud de conversación a más de un usuario, como único requisito es que se tengan un mínimo de tres líneas para cada usuario. En versiones más recientes es posible que **talk** tenga el mismo funcionamiento que **ytalk**.
- ✓ **wall** → Este comando nos permite enviar un mensaje a todos los usuarios que se encuentran conectados al servidor, el mensaje es escrito directamente en pantalla.

Desactivar mensajes en pantalla

- ✓ **mesg** → Este comando indica el estado en que se encuentra configurado el despliegue de mensajes para el usuario, sólo tiene dos estados, y ó n.

- ✓ **is y** (yes), indica que esta habilitada la opción de recepción de mensajes mientras que **is n** (no) indica que no esta habilitada la recepción de mensajes, en ocasiones es conveniente deshabilitar la recepción de mensajes como cuando se están realizando tareas que requieren toda la atención del usuario; por ejemplo cuando se edita un archivo con *vi* o cuando se elabora algún programa.

Para habilitar la recepción de mensajes debemos dar la instrucción ***mesg y.*** Para deshabilitar la recepción de mensajes debemos dar la instrucción ***mesg n.***

- ✓ **diff** → Este comando activa o desactiva la recepción de mensajes proporcionados por el sistema como son la recepción de correo electrónico, etc. Al igual que **mesg** tiene sólo dos estados: activo y desactivo.

☞💻**-** Principales opciones de tar:

- ✓ **-c** → Crea un nuevo archivo tarareado (concentrado).
- ✓ **-v** → *verbose*, imprime en pantalla todo aquellos archivos y/o directorios que se están afectando.
- ✓ **-f** → Indica al sistema que va a afectar (alterar el contenido) a un archivo.
- ✓ **-x** → Extrae el contenido de un archivo .tar
- ✓ **-t** → Lista todos los archivos contenidos en un archivo .tar en formato largo mostrando todos sus detalles.
- ✓ **-u** → **update**, actualiza el contenido del archivo tar realizando una búsqueda dentro de la estructura de directorios del sistema desde donde se inició el proceso de tarareo, añadiendo todos los archivos y directorios que hayan sido creados o sufrido modificaciones después de haber sido creado el archivo tarareado.
- ✓ **-w** → Esta opción pide confirmación para tararear o destararear cada elemento que va a ser afectado.
- ✓ **-d** → Opción no disponible en todas las versiones, hace una búsqueda de todos los archivos y directorios que hayan sufrido modificaciones después de haber sido creado el archivo .tar

☞💻**-** Opciones de de gzip:

- ✓ **-d** → Esta opción permite descomprimir un archivo gzipeado.
- ✓ **-f** → Fuerza la compresión o descompresión de la información afectada sin importar los permisos o propiedades que tenga el archivo o directorio al momento de ser afectado.
- ✓ **-t** → Esta opción hace una revisión del archivo tarareado para verificar su integridad, en caso de que el archivo no se encuentre íntegro el sistema mandara un mensaje indicando el tipo de incongruencia del archivo.
- ✓ **-l** → Ésta opción muestra las propiedades del archivo tarareado indicando el tamaño del archivo cuando esta comprimido, el tamaño de la información cuando se encuentra descomprimida y el porcentaje de compresión de la información.

☞💻**-** Ejemplos de algunas utilerias:

```
javier@inventario:~$ cal
     May 2006
Su Mo Tu We Th Fr Sa
   1  2  3  4  5  6
```

```
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30 31

javier@inventario:~$ cal
     May 2006
Su Mo Tu We Th Fr Sa
   1  2  3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30 31

javier@inventario:~$ df
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/hda6             4738480   1957016   2536876  44% /
/dev/hda7             4816220    100608   4467008   3% /home
javier@inventario:~$ date
Wed May  3 15:09:03 CDT 2006
```

☞💻**-** Revisión del correo.

Tecleando el comando ***mail*** sólo dará acceso a una lista del correo que se haya recibido, aquí se describen algunas opciones para la revisión del correo:

- ✓ **h** → Mostrará una lista numerada de los títulos de los correos recibidos.
- ✓ **#** → Tecleando el número que aparece en el listado de correos se verá el contenido del correo deseado.**r** → Manda una replica del correo que se está visualizando en ese momento.
- ✓ **q** → Sale del programa mail guardando todos los cambios que se hayan realizado como borrado de correos, marcado de correos ya leídos, etc.
- ✓ **x** → Sale del programa mail sin salvar cambios en la lista, es decir que no marcara los correos que ya se consultaron y los mantendrá marcados como nuevos
- ✓ **d#** → Borra el correo marcado en la lista con el # (número) que se le está indicando.

ANEXO 2

☞💻- Descripción de los Message Digest:

SHA[1] → La familia **SHA** (*Secure Hash Algorithm*, Algoritmo de Hash Seguro) es un sistema de funciones *hash* criptográficas relacionadas de la Agencia Nacional de Seguridad estadounidense y publicadas por el *National Institute of Standards and Technology* (NIST). El primer miembro de la familia fue publicado en 1993 es oficialmente llamado **SHA**. Sin embargo, hoy día, no oficialmente se le llama **SHA-0** para evitar confusiones con sus sucesores. Dos años más tarde el primer sucesor de SHA fue publicado con el nombre de **SHA-1**. Existen cuatro variantes más que se han publicado desde entonces cuyas diferencias se basan en un diseño algo modificado y rangos de salida incrementados: **SHA-224**, **SHA-256**, **SHA-384**, y **SHA-512** (todos ellos son referidos como **SHA-2**). En 1998, un ataque a SHA-0 fue encontrado pero no fue reconocido para SHA-1, se desconoce si fue la NSA quién lo descubrió pero aumentó la seguridad del SHA-1.

SHA-1[2] → Ha sido examinado muy de cerca por la comunidad criptográfica pública, y no se ha encontrado ningún ataque efectivo. No obstante, en el año 2004, un número de ataques significativos fueron divulgados sobre funciones criptográficas de *hash* con una estructura similar a SHA-1; esto ha planteado dudas sobre la seguridad a largo plazo de SHA-1.
SHA-0 y SHA-1 producen una salida resumen de 160 bits de un mensaje que puede tener un tamaño máximo de 264 bits, y se basa en principios similares a los usados por el profesor Ronald L. Rivest del MIT en el diseño de los algoritmos de resumen del mensaje MD4 y MD5. La codificación *hash* vacía para SHA-1 corresponde a:

SHA1("") = da39a3ee5e6b4b0d3255bfef95601890afd80709

MD5[3] → En criptografía, **MD5** (acrónimo de *Message-Digest Algorithm 5*, Algoritmo de Resumen del Mensaje 5) es un algoritmo de reducción criptográfico de 128 bits ampliamente usado. El código MD5 fue diseñado por Ronald Rivest en 1991. Durante el año 2004 fueron divulgados ciertos defectos de seguridad, lo que hará que en un futuro cercano se cambie de este sistema a otro más seguro. **MD5** es uno de los algoritmos de reducción criptográficos diseñados por el profesor Ronald Rivest del MIT (*Massachussets Institute of Technology*, Instituto Tecnológico de Masachusets). Cuando un análisis analítico indicó que el algoritmo MD4 era inseguro, se decidió a programar el **MD5** para sustituirlo en 1991. Las debilidades en MD4 fueron descubiertas por Hans Dobbertin.
En 1996 Dobbertin anunció una colisión de *hash* de la función de compresión del MD5. Esto no era una ataque contra la función de *hash* del MD5, pero hizo que los criptógrafos empezasen a recomendar el reemplazo de la codificación MD5 a otras como SHA-1 o RIPEMD-160. En Agosto de 2004 unos investigadores chinos encontraron también colisiones *hash* en el MD5. Actualmente el uso de MD5 es muy amplio y se desconoce cómo afectarán estos problemas a su uso y a su futuro.

[1] http://es.wikipedia.org/wiki/SHA
[2] Idem.
[3] http://es.wikipedia.org/wiki/MD5

ANEXO 3

☞💻**-** Para <table> se pueden presentar los siguientes atributos:

- ✓ **BORDER =n** → Si se especifica se dibujará un borde alrededor de la tabla y separanado los distintos campos que presenta. Se indica un número que especificará el tamaño del borde, por defecto será 0, es decir, no se dibujará ningún borde. Aunque no se dibuje el borde sí se mantendrá el espaciado los elementos de la tabla.
- ✓ **CELLSPACING=n** → Indica el espacio que debe existir entre las distintas celdas de la tabla. Por defecto será 2. Si se indica 0 no habrá ningún espacio entre las celdas.
- ✓ **CELLPADDING=n** → Es la cantidad de espacio entre el borde de la celda y el contenido de ésta, por defecto es 1. Si se indica 0 las celdas aparecerán sin separación.
- ✓ **WIDTH=valor o porcentaje%** → Será el ancho de la tabla, se puede indicar como valor absoluto o como porcentaje del ancho del documento. En el primer caso se definirá en puntos y en el segundo en función del tamaño del documento (tamaño de la ventana del visualizador). Se recomienda utilizar tamaños de tabla en porcentaje del documento, ya que de esta forma la tabla queda perfectamente ajustada para cualquier tamaño del documento.
- ✓ **HEIGHT=valor o porcentaje%** → Definirá el alto de la tabla, a igual que WIDTH, se puede indicar en valor absoluto o en porcentaje. En este caso es más recomendado en valor absoluto ya que el alto es más difícil que pueda coincidir con el tamaño de la ventana.

☞💻**-**Atributos de TR

<TR ALIGN= LEFT o CENTER o RIGHT VALIGN= TOP o MIDDLE ó BOTTOM > → Se podrá especificar por defecto la alineación que tendrán los elementos dentro de las celdas.

- ✓ **ALIGN = LEFT o CENTER o RIGHT** → Indica la alineación del elemento dentro de la celda, en este caso afectará a todos los valores situados en la fila actual, también se podrá indicar individualmente en los elementos TD. Puede tomar uno de los siguientes valores:
  - ➢ LEFT: Alineación a la izquierda de la celda. Este el valor que se toma por defecto
  - ➢ RIGHT: Alineación a la derecha.
  - ➢ CENTER: Indicará centrado.
- ✓ **VALIGN = TOP o MIDDLE o BOTTOM** → Indicará la alineación vertical del dato dentro de la celda. Se podrá especificar donde se colocarán los datos dentro de la celda. Afectarán a toda la fila. Los valores que puede tomar son:
- ✓ **TOP** → Parte superior de la celda.
- ✓ **MIDDLE** → Centrado verticalmente dentro de la celda.
- ✓ **BOTTOM** → En la parte baja de la celda.
- ✓ **BGCOLOR** → Indicará el color de fondo que tendrán todas las celda de la fila de la tabla. El formato para definir los colores es el mismo al que se usa para los atributos de BODY. Esta etiqueta sólo es válida para el *Internet Explorer de Microsoft* y el *Netscape 3.0* o superior.

☞💻**-** Atributos de TD y TH

<TH ALIGN= (LEFT | CENTER | RIGHT | JUSTIFY | DECIMAL) VALIGN= (TOP | MIDDLE | BOTTOM | BASELINE) WIDTH= *Tamaño* BGCOLOR=*color* ROWSPAN="*Filas que debe contener la celda*" COLSPAN= "*Columnas que ocupa la celda*" NOWRAP >

<TD ALIGN= (LEFT | CENTER | RIGHT | JUSTIFY) VALIGN= (TOP | MIDDLE | BOTTOM) WIDTH=*Tamaño* BGCOLOR=*color* ROWSPAN="*Filas que debe contener la columna*" COLSPAN= "*Columnas que ocupa la celda*" NOWRAP>

Pueden presentar los siguientes atributos:

- ✓ **ALIGN** → Indica la alineación horizontal del dato dentro de la celda, se especificará individualmente para cada una de las celdas. Su significado es igual que el expresado para la etiqueta TR.
- ✓ **VALIGN** → Indicará la alineación vertical del dato dentro de la celda. Se especifica individualmente para cada celda, el formato es el mismo que el expresado para TR.
- ✓ **WIDTH** → Especifica el ancho que tendra la columna de la tabla, se puede especificar eln valor absoluto, en puntos de la pantalla o en tanto por ciento del tamaño de la tabla.
- ✓ **BGCOLOR** → Indicará el color de fondo que tendrá la celda de la tabla. El color hay que indicarlo independientemente para cada una de las celdas de la columna. El formato para definir los colores es el mismo al que se usa para los atributos de BODY. Esta etiqueta sólo es válida para el *Internet Explorer de Microsoft* y el *Netscape 3.0* o superior.
- ✓ **ROWSPAN** → Indicará el número de filas que ocupará está celda en la misma fila. En este caso la celda se expandirá ocupando tantas celdas de la tabla inicial como se especifique. Esto permite definir por ejemplo celdas de cabecera que afecten a varias celdas de la tabla, o bien, crear una fila que ocupen toda la tabla.
- ✓ **COLSPAN** → Indicará el número de columnas que ocupará la celda actual, obtendremos una celda que ocupa varias columnas. Igual que el anterior pero para el caso de las columnas.
- ✓ **NOWRAP** → Se usará para que no se divida la línea por defecto. Si la usamos las líneas de texto no se dividirán dentro de la celda en varias líneas. Por tanto si la linea es muy larga la columna de la tabla será tan ancha como la línea, sólo se dividirá si se usa el elemento.

EJEMPLOS

A continuación se presentan algunos ejemplos de Tablas:

Una Tabla Simple:

1	2	3
4	5	6

```
<TABLE BORDER>
<TR>
<TD>1 <TD> 2 <TD> 3
<TR>
<TD>4 <TD> 5 <TD> 6
</TABLE>
```

Demostración de TH, COLSPAN y ROWSPAN:

	Numeros		
Num	1	2	3
	4	5	6

```
<TABLE BORDER>
<TR>
<TD><TH COLSPAN=3>Numeros
```

```
<TR>
<TH ROWSPAN=2>Num<TD>1 <TD> 2 <TD> 3
<TR>
<TD>4 <TD> 5 <TD> 6
</TABLE>
```

Demostración de Multiples Cabeceras:

TITULO # 1		TITULO # 2	
Datos1.1	**Datos1.2**	**Datos2.1**	**Datos2.2**
1	2	3	4
5	6	7	8

```
<TABLE BORDER>
<TR>
<TH COLSPAN=2>Datos1 </TH><TH COLSPAN=2>Datos2 </TH>
</TR>
<TR>
<TH>Datos1.1 </TH><TH>Datos1.2 </TH><TH>Datos2.1 </TH><TH>Datos2.2 </TH>
</TR>
<TR>
<TD>1</TD><TD>2</TD><TD>3</TD><TD>4</TD>
</TR>
<TR>
<TD>5</TD><TD>6</TD><TD>7</TD><TD>8</TD>
</TR>
</TABLE>
```

Tabla sin Borde:

1 2 3

4 5 6

```
<TABLE>
<TR>
<TD>1 <TD> 2 <TD> 3
<TR>
<TD>4 <TD> 5 <TD> 6
</TABLE>
```

Tabla con Borde=15:

1	2	3


```
<TABLE BORDER=15>
<TR>
<TD>1 <TD> 2 <TD> 3
<TR>
<TD>4 <TD> 5 <TD> 6
</TABLE>
```

Ejemplos de colores de fondo de las celdas BGCOLOR: (Sólo para el Netscape 3.0 e Internet Explorer en adelante)

Cabecera de Titulo 1	Cabecera de Titulo 2	Cabecera de Titulo 3
Celda 1	Celda 2	Celda 3
Celda 4	Celda 5	Celda 6

```
<TABLE BORDER>
<TR BGCOLOR=YELLOW>
<TH>Cabecera de Titulo 1 <TH> Cabecera de Titulo 2 <TH> Cabecera de Titulo 3
<TR BGCOLOR=WHITE>
<TD>Celda 1 <TD> Celda 2 <TD> Celda 3
<TR>
<TD BGCOLOR=RED>Celda 4 <TD BGCOLOR=WHITE> Celda 5 <TD BGCOLOR=GREEN>
Celda 6
</TABLE>
```

☞💻- Tipos de Instrucciones de Entrada:

<INPUT TYPE=TEXT...> Texto corto

Se utiliza para la entrada de cadenas de texto corto, como por ejemplo nombre de personas, números, fechas o diversos datos que se puedan expresar en una línea de texto. Se mostrará un recuadro que ocupa una línea y la que será posible especificar este texto. El formato de la instrucción es el siguiente:

< INPUT TYPE=TEXT NAME="*variable*" VALUE="*valor inicial*" SIZE="*tamaño*" MAXLENGTH="*longitud máxima*" >

El tamaño de la ventana de introducción de texto se fija con el atributo SIZE, que indica el tamaño de la ventana en caracteres. Aquí sólo se define la parte visible, pero el usuario podrá introducir más texto si lo desea. Para indicar el máximo número de caracteres que se permiten en la entrada usaremos: MAXLENGTH. El atributo NAME indica el nombre de la variable que toma el valor de la entrada y VALUE especifica el valor de inicialización del campo. De todos los atributos

sólo será obligatorio NAME, siendo el resto opcionales. En la entrada se podrán usar cualquier tipo de caracteres incluso los acentuados, en su formato normal.

<INPUT TYPE=PASSWORD...> Palabras secretas

Es similar al anterior pero en este caso no se imprimen los caracteres según se van introduciendo, se muestra un asterisco en vez de los caracteres. Sólo se puede ver el número de caracteres, pero no valor. Se usa para la introducción de claves de acceso (passwords) y datos que no deban ser vistos al introducirlos. El formato es:

< INPUT TYPE=PASSWORD NAME="*variable*" VALUE="*valor inicial*" SIZE="*tamaño*"

MAXLENGTH="li>longitud máxima" >

<INPUT TYPE=CHECKBOX> Botones de selección

El checkbox es un botón que puede presentar dos estados activado o desactivado. El formato es el siguiente:

< INPUT TYPE=CHECKBOX NAME="variable" CHECKED>

Se requiere el atributo **NAME**. Los valores que tomará la variable serán *on* ó *off*, dependiendo de su estado. Si el botón estaba activado cuando se envían los datos del formulario se enviaran el nombre de la variable y el valor que indique su estado. Si se incluye el atributo **CHECKED** el botón se encontrará activado en la inicialización. Si se indica el atributo **VALUE**, cuando se envian los datos con el botón activado se mandará la variable con el valor indicado y en caso contrario no se mandará ningún valor.

<INPUT TYPE=RADIO...> Selección entre múltiples opciones

Se usa cuando la opción puede tomar un valor simple de una serie de alternativas. En este caso se presentan unos valores opcionales de los que sólo puede tomar un valor. Para especificar cada uno de estos valores se incluirá una etiqueta RADIO por cada una de las posibles alternativas, su estructura general será:

< INPUT TYPE=RADIO NAME="*variable*" VALUE="*valor 1*" CHECKED >

< INPUT TYPE=RADIO NAME="*variable*" VALUE="*valor 2*" >

. . .

< INPUT TYPE=RADIO NAME="*variable*" VALUE="*valor n*" >

Cada una de las etiquetas RADIO tendrá el mismos atributo NAME, y con un distinto atributo VALUE que será el valor que tome si se selecciona esta opción. Para inicializarlo se usa el atributo CHECKED que se indicará sólo en la opción que se quiera especificar por defecto.

<INPUT TYPE=HIDDEN...> Parámetros ocultos

En este caso no se muestra ningún campo para la entrada de datos al usuario, pero el parámetro, variable o valor especificado es enviado junto con el formulario. Se suele usar para transmitir información de estado ó control ó para enviar algún tipo de información que no debe ser variada en el formulario, pero sí debe ser enviada junto a este. El formato es:

< INPUT TYPE=HIDDEN NAME="*variable*" VALUE="*valor*" >

Deberá incluir tanto la variable como el valor.

<INPUT TYPE=SUBMIT...> Enviar Datos

Este botón se usa para enviar los datos del formulario, al pulsar el usuario este botón, se acaba la introducción del formulario y pasa el control al programa indicado en ACTION. En todo

formulario debe existir al menos un botón de SUBMIT, si sólo incluye un recuadro del tipo TEXT no será necesario incluirlo. El formato es:

< INPUT TYPE=SUBMIT VALUE="*mensaje a mostrar*" >

El atributo VALUE especifica una etiqueta no editable que se mostrará en el botón de envío. Lo normal es que este botón no envíe datos, pero si se indica el atributo NAME con un nombre de variable será enviada la variable con el valor de VALUE. Esto puede ser útil si se incluyen distintos botones de SUBMIT para distinguir cual fue pulsado.

<INPUT TYPE=IMAGE...> Botón de Envío gráfico

Su funcionalidad es similar al botón de SUBMIT, se usa igualmente para enviar los datos de un formulario, pero en este caso se presenta una imagen como botón. Igualmente al pulsar sobre el botón se enviará el formulario. El formato es el siguiente:

< INPUT TYPE=IMAGE NAME="*variable*" SRC="*URL de la Imagen*" >

El punto de la imagen en el que pulsa el usuario también es pasado al programa interprete del formulario, de forma que la imagen igualmente podría ser un mapa sensible. Se pasarán dos parámetros x e y con las coordenadas del punto donde pulso, siendo el programa interprete el encargado de determinar la zona donde se pulsó, si se desea.

<INPUT TYPE=RESET...> Borrar los datos

Este botón se usa para volver a los valores por defecto todos los elementos del formulario, borrando todos los datos introducidos por el usuario. Su formato es el siguiente:

< INPUT TYPE=RESET VALUE="*Etiqueta a mostrar*" >

El atributo VALUE especifica la etiqueta que tendrá el botón.

☞💻- Elementos y atributos de FRAME

- ✓ **URL** → Como su nombre indica, especifica el documento HTML o fichero que se mostrará en la frame definida. Si no se especifica documento alguno se mostrará la frame vacía.
- ✓ **NAME** → Indica el nombre de la frame, este nombre es importante ya que se usará en los hiperenlaces (normalmente en los documentos de las otras frames) para indicar la frame de destino del documento. Si no se indica el nombre sólo se podrá mostrar el documento actual, sin que sea posible cambiarlo mediante hiperenlaces.
- ✓ **MARGINWIDTH** → Indica el ancho del margen, este atributo es opcional y normalmente el navegador ajusta todos los márgenes al mismo tamaño, si se especifica se valor será en puntos de la pantalla.
- ✓ **MARGINHEIGHT** → Igual que en el caso anterior pero para el alto de los márgenes. Lo normal es no especificar ninguno de estos dos atributos.
- ✓ **SCROLLING** → Indica si la frame tendrá o no una barra de scroll, la barra de scroll se muestra en el lateral y permite desplazarse por el documento, pulsando con el ratón en ella. Si toma el valor YES, siempre se mostrará esta barra, para el valor AUTO sólo se mostrará si el documento no cabe en la frame, si es necesaria. Y por último NO indica que en ningún caso se muestre la barra de scroll. Si no se indica nada se toma por defecto el valor AUTO.
- ✓ **NORESIZE** → Indica que la frame no debe ser variada de tamaño por el usuario, se puede variar el tamaño de una frame situando el cursor del ratón encima y arrastrando en la dirección deseada. Con este atributo el usuario no podrá cambiar el tamaño de la frame en ningún caso. Por defecto todas las frames pueden variar su tamaño.

☞💻- Etiquetas para TARGET

A → En los hiperenlaces indicará la frames donde se mostrará el documento, una vez que se siga el hiperenlace.

<A HREF="*url*" TARGET="*frame*">

BASE → Indicará la frame en la que se mostrará por defecto todos los hiperenlaces del documento actual. Se debe especificar en la cabecera del documento (HEAD).

<BASE TARGET="*frame*">

AREA → Se indica que el frame donde se verá el documento que se activa en la zona correspondiente de la imagen.

<AREA SHAPE=RECT COORDS="*x,y,...*" HREF="*url*" TARGET="*frame*">

FORM → Indicará la frame de destino del resultado del formulario.

<FORM ACTION="*url*" TARGET="*frame*">

Existen valores especiales de TARGET que permiten definir destinos distintos a las frames definidas. Estos valores son los siguientes:

- ✓ **TARGET="_blank"** → Indica que se muestre en una nueva ventana vacía, para seguir el enlace se lanza otra ventana distinta del navegador.
- ✓ **TARGET="_self"** → Se mostrará en la misma ventana o frame que lo referencia, se puede usar para modificar el valor dado por BASE.
- ✓ **TARGET="_parent"** → Se muestra en la frame o estructura de frames que llamó al documento actual.
- ✓ **TARGET="_top"** → Indica que se muestre en la ventana completa, eliminando la estructura de frames que tenga la ventana.

☞💻- Caracteres especiales:

Si se incluyen en el texto de un documento HTML el símbolo menor que (<) o mayor que (>) se interpretará siempre como la definición de una etiqueta y por tanto no se mostrarán al interpretar el documento. Para expresar estos símbolos y otros del lenguaje HTML se usan las siguientes secuencias de escape:

Sec. Escape	Símbolo
<	Signo **<** (menor que)
>	Signo **>** (mayor que)
&amp;	Signo **&** (ampersand)
&quot;	Se mostrará el signo de comillas **"**

Caracteres acentuados → Existen una serie de etiquetas que permiten mostrar los caracteres acentuados y caracteres latinos (ñ). Estos caracteres si se incluyen en un documento HTML sin utilizar las secuencias de escape, se mostrarán correctamente. El usuario en la PC con su navegador podrá leer estos caracteres correctamente siempre y cuando tenga el mismo código (español) que el autor del documento. Los primeros 127 caracteres del código ASCII son comunes para todos los países e incluyen todas las letras del alfabeto, a partir del 128 son específicos para cada lenguaje, entre estos se incluyen los caracteres acentuados y la letra ñ, por tanto si alguien desde otro país que tenga un código incompatible intenta leer su documento, probablemente encuentre caracteres extraños que no sepa interpretar y por tanto no será capaz de leer los caracteres acentuados.

Existen diversas secuencias que definen los distintos tipos de acentos: agudo, grave o circunflejo. Para el acento agudo usaremos el literal *acute*, tanto para las mayúsculas como para las minúsculas. Por tanto se incluirá el símbolo de ampersand (&), la vocal que se desea acentuar, la palabra *acute* y el símbolo punto y coma (;). Representándose los acentos de la siguiente forma:

Sec. Escape	Letra		Sec.Escape	Letra
&**a**acute;	á		&**A**acute;	Á
&**e**acute;	é		&**E**acute;	É
&**i**acute;	í		&**I**acute;	Í
&**o**acute;	ó		&**O**acute;	Ó
&**u**acute;	ú		&**U**acute;	Ú

Para la letra ñ se usa la secuencia *tilde*, del siguiente modo:

Sec. Escape	Letra		Sec.Escape	Letra
ñ	ñ		Ñ	Ñ

Para el acento grave se usa *grave*, siendo en este caso la representación:

Sec. Escape	Letra		Sec.Escape	Letra
&**a**grave;	à		&**A**grave;	À
&**e**grave;	è		&**E**grave;	È
&**i**grave;	ì		&**I**grave;	Ì
&**o**grave;	ò		&**O**grave;	Ò
&**u**grave;	ù		&**U**grave;	Ù

Otros Símbolos → Para expresar una carácter por su valor en el código ASCII, se usa el símbolo **#**, seguido de su equivalente numérico. Para el acento circunflejo (**^**) se utiliza el literal *circ* y para la diéresis (¨) se utiliza el literal *uml*. Para expresar los símbolos de apertura interrogación, apertura de exclamación y estos acentos se usan:

Sec. Escape	Letra		Sec.Escape	Letra

&#191;	¿	&#161;	¡
&uuml;	ü	&Uuml;	Ü
&icirc;	î	&Icirc;	Î

Algunos otros símbolos especiales serán los siguientes.

Sec. Escape	Simbolo	Sec.Escape	Simbolo
&**c**cedil;	ç	&**C**cedil;	Ç
&reg;	® Símbolo de registrado	&copy;	© Símbolo de Copyright.
&#nnn;	Donde nnn es un número decimal, el carácter nnn del código ISO-8859-1 (ASCII).		

ANEXO 4

☞💻- Clases más utilizadas de expresiones regulares:

- ✓ **alnum: "[[:alnum:]]"** → Concuerda con cualquier cadena que contenga caracteres alfanuméricos.
- ✓ **digit: "[[:digit:]]"** → Cualquier cadena que contenga caracteres numéricos.
- ✓ **space: "[[:space:]]"** → Cualquier cadena que contenga espacios.
- ✓ **alpha: "[[:alpha:]]"** → Cualquier cadena que contenga caracteres alfabéticos.
- ✓ **upper : "[[:upper:]]"** → Cualquier cadena que contenga caracteres alfabéticos en mayúsculas.
- ✓ **lower : "[[:lower:]]"** → Cualquier cadena que contenga caracteres alfabéticos en minúsculas.
- ✓ **punct : "[[:punct:]]"** → Cualquier cadena que contenga signos de puntuación.
- ✓ **xdigit : "[[:xdigit:]]"** → Cualquier cadena que contenga dígitos hexadecimales (es equivalente a [0-9a-fA-F]).

- ✓ **"a+"** → Concuerda con cualquier cadena que contenga al menos una "a".
- ✓ **"a*"** → Cero o más ocurrencias de "a".
- ✓ **"a?"** → Cero o más ocurrencias de "a".
- ✓ **"a{2}"** → Una secuencia de dos "a".
- ✓ **"a{2,4}"** → Una secuencia de dos a cuatro "a".

Ahora, ya se pueden construir algunas expresiones regulares útiles:

- ✓ **"^[[:alnum:]]{4}$"** → Una cadena conteniendo exactamente cuatro caracteres alfanuméricos.
- ✓ **"^[^@[:space:]]+@[^@[:space:]]+$"** → Esta expresión concuerda con cualquier cadena que contenga cualquier carácter excepto "@" y " " (Espacio) antes de una "@" (arroba) y lo mismo después de una "@".
- ✓ **[^.]*$"** → Concuerda con todos los caracteres al final de una cadena y antes de ".".

Nótese que la sintaxis de las expresiones regulares es muy promiscua. Esto quiere decir que intenta concordar tantas veces como sea posible:

```
print ereg_replace("<.*>", "", "Test");
```

Esto no eliminará todos las etiquetas HTML como cabría esperar, pero retornará una cadena vacía: Reemplaza "<" seguido por cualquier carácter (incluyendo ">") hasta el final de la cadena.

La forma correcta para hacerlo sería algo así:

```
print ereg_replace("<[^>]*>", "", "Test");
```

☞💻**-** Algunas de las funciones informativas de php son las siguientes:

Función	Descripción
file_exists(archivo)	Esta función devuelve TRUE si el archivo existe, en caso contrario devuelve FALSE.
filesize(archivo)	Devuelve el tamaño del archivo expresándolo en bytes. En caso de que no existiera el archivo nos dará un error.
filetype(archivo)	Devuelve una cadena indicando el tipo de archivo. En caso de que no existiera el archivo nos dará un error.
filemtime(archivo)	Devuelve la fecha de la última modificación del archivo expresada en tiempo Unix.
stat(archivo)	Devuelve un array con múltiple información sobre el archivo. En la tabla que aparece a continuación se detallan los contenidos asociados a cada uno de sus índices.

Valores del array devuelto por la función stat

Indice	Significado
0	Dispositivo
1	I node
2	Modo de protección de I node
3	Número de enlaces
4	Id de usuario del propietario
5	Id de grupo del propietario
6	tipo de dispositivo si es un inode device *
7	Tamaño en bytes
8	Fecha del último acceso
9	Fecha de la última modificación
10	Fecha del último cambio
11	Tamaño del bloque para el sistema I/O *
12	Número de bloques ocupados

ANEXO 5

☞💻- Funciones de Conversión de Tipos

(estos están tomados de la ayuda de Visual Basic .NET)

Nombre de la función	Tipo de datos que devuelve	Valores del argumento "expresion"
CBool(*expresion*)	Boolean	Cualquier valor de cadena o expresión numérica.
CByte(*expresion*)	Byte	de 0 a 255; las fracciones se redondean.
CChar(*expresion*)	Char	Cualquier expresión de cadena; los valores deben ser de 0 a 65535.
CDate(*expresion*)	Date	Cualquier representación válida de una fecha o una hora.
CDbl(*expresion*)	Double	Cualquier valor Duoble, ver la tabla anterior para los valores posibles.
CDec(*expresion*)	Decimal	Cualquier valor Decimal, ver la tabla anterior para los valores posibles.
CInt(*expresion*)	Integer	Cualquier valor Integer, ver la tabla anterior para los valores posibles, las fracciones se redondean.
CLng(*expresion*)	Long	Cualquier valor Long, ver la tabla anterior para los valores posibles, las fracciones se redondean.
CObj(*expresion*)	Object	Cualquier expresión válida.
CShort(*expresion*)	Short	Cualquier valor Short, ver la tabla anterior para los valores posibles, las fracciones se redondean.
CSng(*expresion*)	Single	Cualquier valor Single, ver la tabla anterior para los valores posibles.
CStr(*expresion*)	String	Depende del tipo de datos de la expresión.
		Nota: Todos los objetos de VB.NET tienen unos métodos para realizar conversiones a otros tipos, al menos de número a cadena, ya que tienen la propiedad .ToString que devuelve una representación en formato cadena del número en cuestión (igual que CStr).
CType(*expresion*, Tipo)	El indicado en el segundo parámetro	Cualquier tipo de datos
Val(*expresion*)	Double	Una cadena de caracteres.
Fix(*expresion*)	Depende del tipo de datos de la expresión	Cualquier tipo de datos
Int(*expresion*)	Depende del tipo de datos de la expresión	Cualquier tipo de datos

☞💻- Los símbolos que se pueden usar para efectuar comparaciones, son los siguentes:

=	igual
<	menor que
>	mayor que
<=	menor o igual
>=	mayor o igual
<>	distinto