UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN

Guía para desarrollar aplicaciones Web con Apache, PHP y MySQL

TRABAJO ESCRITO EN LA MODALIDAD DE
SEMINARIO O DIPLOMADO DE ACTUALIZACIÓN Y CAPACITACIÓN PROFESIONAL

QUE PARA OBTENER EL TITULO DE INGENIERO EN COMPUTACIÓN
PRESENTA

José Manuel Aguilar Argandar

ASESORA:
Biol. Lizbeth Heras Lara

Octubre de 2006, Edo. Méx.

Agradecimientos.

Debo agradecer a muchas personas, principalmente,

a mis padres,

a mi hermana,

a mis profesores

y a mis amigos.

Índice

Introducción y objetivos.

El diplomado "Desarrollo e Implementación de Sistemas con Software Libre en Linux" se realiza trimestralmente en el Centro de Enseñanza de Lenguas Extranjeras, sede Mascarones, perteneciente a la UNAM. La realización y planeación ha sido una tarea de la Dirección General de Servicios de Cómputo Académico. Éste diplomado cubre la demanda que en los últimos años cientos de profesionistas tienen en materia de tecnologías de software libre.

Las tecnologías, arquitecturas y metodologías para desarrollar software cambian constantemente; la creciente demanda de sistemas Web por parte de las empresas hace que la industria del software, en conjunto con las universidades, redefinan la manera de hacer software en un lapso corto de tiempo.

La duración total del diplomado es de 130 horas, sin embargo, lo he complementado con una serie de cursos (que suman 120 horas) impartidos en otras sedes de la DGSCA, sumando así 250 horas en total.

Éste trabajo pretende sintetizar los módulos que conforman al diplomado y a cada uno de los cursos extras, presentándolos a manera de guía y referencia para todos aquellos interesado en el desarrollo de aplicaciones Web.

La realización de éste trabajo deberá facilitar a los programadores recién llegados al mundo de las aplicaciones Web una introducción al desarrollo de aplicaciones Web

El capítulo 1 está dedicado a identificar el software libre y la instalación de una distribución de GNU/Linux. La administración de un servidor a través de comandos se trata en el capítulo 2.
El capítulo 3 se dedica al estudio del lenguaje HTML y a la realización de páginas Web dinámicas con el lenguaje PHP. Los elementos básicos de SQL y la interacción de PHP con el servidor de bases de datos MySQL se contemplan en el capítulo 4. El capítulo 5 retoma algunos elementos del capítulo anterior para mostrar la interacción de PHP con un servidor de bases de datos PostgreSQL. La forma de crear páginas Web dinámicas con Perl se estudia en el capítulo 6. Finalmente, el capítulo 7 se ha dedicado al análisis del lenguaje Java como una forma más de crear páginas Web dinámicas.

Objetivo general.

- Realizar una guía sobre como desarrollar aplicaciones Web con software libre para que sirva como referencia a los desarrolladores Web.

Objetivos específicos.

- Analizar cada modulo que conformo al diplomado "Desarrollo e Implementación de Sistemas con Software Libre en Linux" para extraer los temas mas importantes que conformarán a esta guía.

- Investigar las metodologías y arquitecturas de vanguardia para el desarrollo de aplicaciones Web y seleccionar la mas conveniente.

- Analizar las ventajas y desventajas del uso de un *web application framework* para decidir si es necesario su uso en el desarrollo de una aplicación Web.

Capítulo 1.

Instalación y administración básica de un sistema GNU/Linux.

"Free software" is a matter of liberty, not price. To understand the concept, you should think of "free" as in "free speech", not as in "free beer".

GNU Project

1.1 El software libre.

En los años 60 y 70 del siglo pasado, el software no era considerado un producto, era sólo un añadido que los vendedores de computadoras hacían a sus clientes para que pudieran usarlas. En aquellos años era muy común que los programadores y desarrollares compartieran el código fuente de sus programas unos con otros. Al finales de los 70, las compañías comenzaron a poner restricciones[1] para llevar a cabo esta costumbre.

Entre los años de 1983 y 1985, Richard Stellman, comenzó con el proyecto GNU[2] y fundó la FSF[3] con el fin de proteger la libertad de los usuarios del software y crear un sistema operativo libre.

El software libre se refiere a la libertad que tienen los usuarios para ejecutar, modificar, copiar, distribuir, estudiar y reutilizar el software. El proyecto GNU se ha referido a cuatro tipos de libertades:

- La libertad para correr el programa, para cualquier propósito (libertad 0).
- La libertad de estudiar como trabaja el programa y adaptarlo a las necesidades (libertad 1).
- La libertad de distribuir copias ayudando al vecino (libertad 2).
- La libertad de mejorar el programa y hacer publicas las mejoras, de modo que la comunidad se beneficie (libertad 3). El acceso al código fuente es necesario para realizar esto.

Cualquier software que cumpla con estas libertades puede ser nombrado software libre, así los usuarios u organizaciones de éste, pueden distribuir las copias, con ciertas modificaciones o no, ya sea gratis o cobrando una cantidad por las distribuciones a cualquiera o en cualquier lugar.

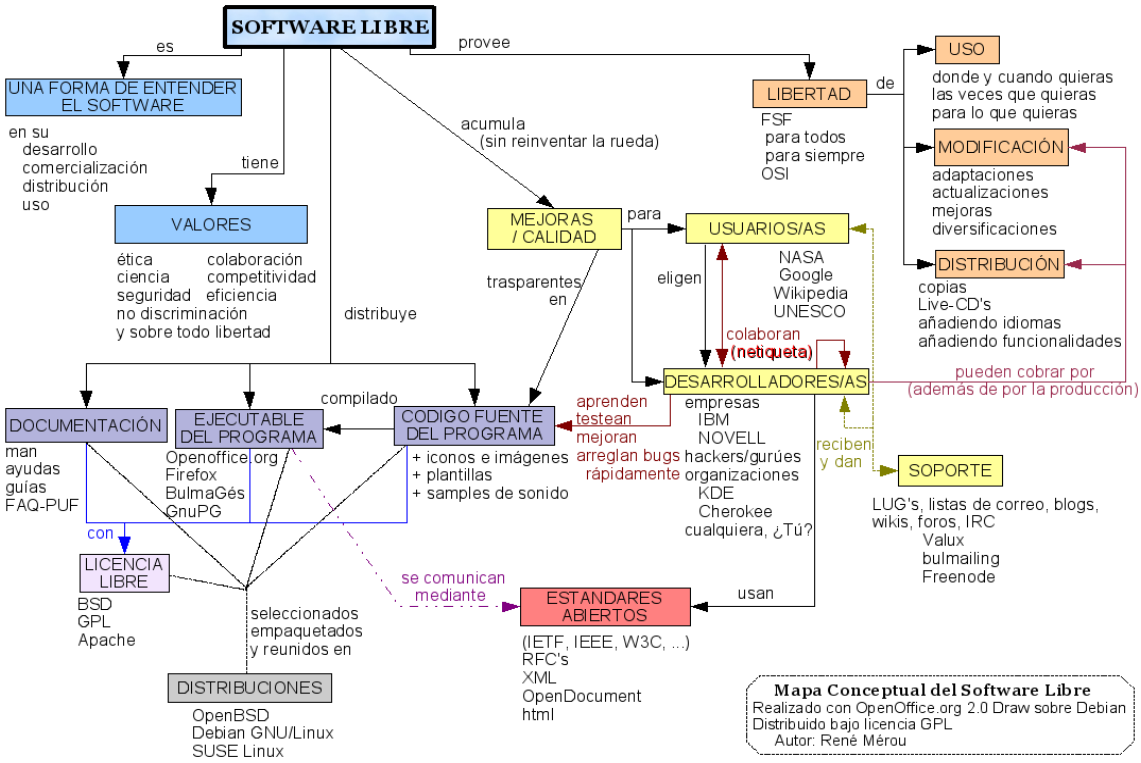
Un requisito indispensable es disponer del código fuente del programa, para que la libertad de modificar y mejorar tengan sentido.

También debe quedar entendido que software libre no significa "no comerciar", por supuesto que se puede comerciar con el software libre, de hecho se debe hacer software libre para su uso comercial, desarrollo comercial y distribución comercial.

Quizás se haya pagado por copias del software libre, o tal vez no, pero independientemente de como se haya obtenido, siempre se tiene la libertad de copiar, modificar y distribuir, e incluso cobrar por él.

En la siguiente figura podemos ver un mapa conceptual del software libre, realizado por René Mérou[4].

1 http://www.bachue.com/colibri/faifes/chapter-1.html
2 GNU is not Unix, http://www.gnu.org
3 Free Software Foundation, fundación de software libre
4 http://www.es.gnu.org/~reneme/mapa-conceptual-software-libre.png



1.2 Historia de UNIX y GNU/Linux.

El origen de UNIX fue al final de la década de los sesentas cuando el MIT[5], los laboratorios Bell de AT&T y General Electric trabajaban en un sistema operativo denominado MULTICS (Multiplexed Information and Computing System) que debía ejecutarse en un mainframe modelo GE-645. Cuando los Laboratorios Bell decidieron abandonar el proyecto uno de sus programadores, Ken Thompson, decidió seguir trabajando en dicha computadora y desarrolló un juego de video llamado Space Travel. Cuando Thompson descubrió que el juego se desenvolvía muy lento en aquella plataforma y que jugarlo costaba algo así como 75 dólares por partida, decidió escribirlo en lenguaje ensamblador para una computadora DEC PDP-7, esta vez con ayuda de Dennis Ritchie.

Habiendo acumulado la experiencia de éste proyecto y la de MULTICS, decidieron escribir un sistema operativo para la DEC PDP-7. Desarrollaron un sistema multitarea con un sistema de archivos, un intérprete de comandos y un conjunto de programas. El proyecto fue denominado UNICS (Uniplexed Information and Computing System), haciendo broma a MULTICS pues sólo prestaba servicio a dos usuarios. Después de un tiempo fue cambiado el nombre por el de UNIX.

En 1973 decidieron escribir nuevamente UNIX pero esta vez en lenguaje C y con apoyo económico de los Laboratorios Bell. Éste cambio permitiría hacer portable, compacto y conciso el código de UNIX. AT&T puso a UNIX a disposición de universidades y empresas de los Estados Unidos a través de licencias no muy costosas en aquellos tiempos.

Para 1975 ya se habían desarrollado las versiones 4, 5 y 6 que incluían tuberías o pipes, esto hizo que el desarrollo de UNIX fuera modular. En 1978 había más de 600 computadoras corriendo UNIX en alguna de sus versiones.

5 Massachusetts Institute of Technology

El kernel Linux fue un proyecto universitario que inició en 1991 Linus Torvalds, basándose en el sistema operativo Minix desarrollado por Andrew S. Tanenbaum. En aquel tiempo el kernel Linux debía estar instalado en una máquina con Minix para poder ser utilizado. El 5 de octubre de 1991, Linus anunció en un canal de usernet, comp.os.minix, la primera versión oficial, la 0.02. Esta versión podía ejecutar satisfactoriamente el `bash` (GNU Bourne Again Shell), el compilador `gcc`, `gnu-make`, `gnu-sed`, `compress`, entre otros. La puso a disposición de todo el mundo para que cada quien pudiera utilizarla, modificarla y mejorarla.

Cuando se llegó a la versión 0.03, Linus cambió éste número por el 0.10. La aceptación por parte de los usuarios en todo el mundo se llevó en poco tiempo, marzo de 1992, a la versión 0.95. En diciembre de 1993, estaba lista la versión 0.99 y quedaba claro que pronto llegaría la versión oficial con la denominación 1.00.

Éric Lévénez actualiza constantemente un cronograma muy extenso de las distribuciones de UNIX y el kernel Linux, el sitio web es http://www.levenez.com/unix/.

1.2.1 Sistema de archivos

Con mucha frecuencia los resultados de los procesos u operaciones que la CPU lleva a cabo se deben almacenar de manera permanente en algún dispositivo de almacenamiento no volátil para un uso futuro, de modo que el sistema operativo debe de contar con una estructura que permita almacenar y accesar a estos resultados, esto se logra ubicando la información en archivos. La parte del sistema operativo que se encarga de la gestión de los archivos recibe el nombre de sistema de archivos.

Los procesos pueden leer, escribir o crear nuevos archivos. La información almacenada en los archivos es persistente, es decir, no se altera después de la desaparición de un proceso.

1.2.2 Distribuciones

GNU/Linux llega hasta los usuarios mediante distribuciones, éstas son creadas por organizaciones de voluntarios, empresas privadas y públicas, universidades y hasta por particulares. Algunas tiene un precio y otras se mantiene por donaciones. No todas las distribuciones son iguales, algunas incluyen sofisticados asistentes de instalación, administradores de instalación y actualización de software, entre muchas otras herramientas.
Las distribuciones se han ido creando por las distintas necesidades y propósitos, arquitecturas de hardware, regiones e idioma, o bien como pasatiempo de algunos usuarios.

Se calcula que existen aproximadamente trescientas distintas distribuciones en desarrollo, sin embargo las más populares y con mayor soporte son las siguientes:

- Debian GNU/Linux
- Fedora
- Gentoo Linux
- Mandriva Linux
- Red Hat Enterprise
- Slackware Linux
- SuSE Linux
- Ubuntu

La siguiente lista de sitios web nos será útil cuando deseemos conocer más distribuciones y las características que ofrecen:

- http://www.linuxiso.org/
- http://www.distromania.com/
- http://www.linuxdistro.info/
- http://lwn.net/Distributions/

1.3 Slackware

La primera versión de Slackware fue lanzada el 16 de julio de 1993 por Patrick Volkerding, fundador y desarrollador. Fue basada en la distribución SLS Linux que se entregaba en discos de 3½". Su desarrollo y mantenimiento continúa hasta la fecha. Sus objetivos siempre han sido la estabilidad, simplicidad, fácil instalación y administración, a demás de cumplir con los estándares de GNU/Linux.

Slackware será la distribución que utilizaremos a lo largo de éste trabajo, específicamente la versión 10.2.

Para mayor información y descargas de Slackware se pueden consultar los siguientes sitios:

- http://www.slackware.com/index.php
- http://www.slackbook.org/
- http://slackwiki.org/Main_Page
- http://slackworld.berlios.de/
- http://www.tuxjm.net/index.html

1.3.1 Requerimientos de Hardware

Los requerimientos básicos para la instalación de Slackware son:

- Procesador 586
- RAM 32 MB
- Espacio en disco duro1GB
- CD-ROM 4x

Cuando la computadora no cuenta con una unidad de CD-ROM se puede recurrir a una instalación por NFS (se requiere una tarjeta de red) y también es necesario contar con una unidad de 3½" (floppy).

GNU/Linux es tan flexible que puede ser instalado en el disco duro en tan sólo 100MB, sin embargo, la instalación completa de Slackware requiere de 2GB más unos cuantos megabytes para los archivos de los usuarios.

Las características del equipo de computo empleado en la realización de éste trabajo son las siguientes:

- Procesador Pentium IV 3.0 GHz
- RAM 512 MB
- Espacio en disco duro40GB
- CD-ROM/DVD 50x/12x

1.3.2 Organización de paquetes en Slackware

Hace un par de años Slackware se distribuía en discos de 3½", y la forma de organizar cada disco era por series ("*disk set*"). Esta organización se sigue manteniendo sólo que ahora varias series están juntas en un CD.

Esta es una descripción de cada serie:

Serie	Descripción
A	El sistema base. Contiene el software necesario para levantar y correr el sistema, así como editores de texto y programas sencillos de comunicación.
AP	Varias aplicaciones que no requieren de un ambiente gráfico (X Window).
D	Software de desarrollo. Compiladores, debuggers, intérpretes y las páginas de manual.
E	GNU Emacs. Herramienta para edición de textos, multifuncional.
F	Documentación extra. *FAQs*, *HOWTOs.*
GNOME	El escritorio Gnome (Administrador de ventanas).
K	El código fuente del kernel Linux.
KDE	El escritorio KDE (Administrador de ventanas).
KDEI	Paquetes de internacionalización de KDE.
L	Librerías dinámicas que algunos programas pueden emplear.
N	Software de red. Demonios, servidores web, programas de mail, telnet, etc.
T	Sistema tipográfico teTex.
TCL	Tool Command Language. Tk, TclX, and TkDesk.
X	El sistema de ventanas X (X Window System)
XAP	Aplicaciones que requieren un ambiente gráfico (X Window System).
Y	Juegos para modo texto (BSD console games)

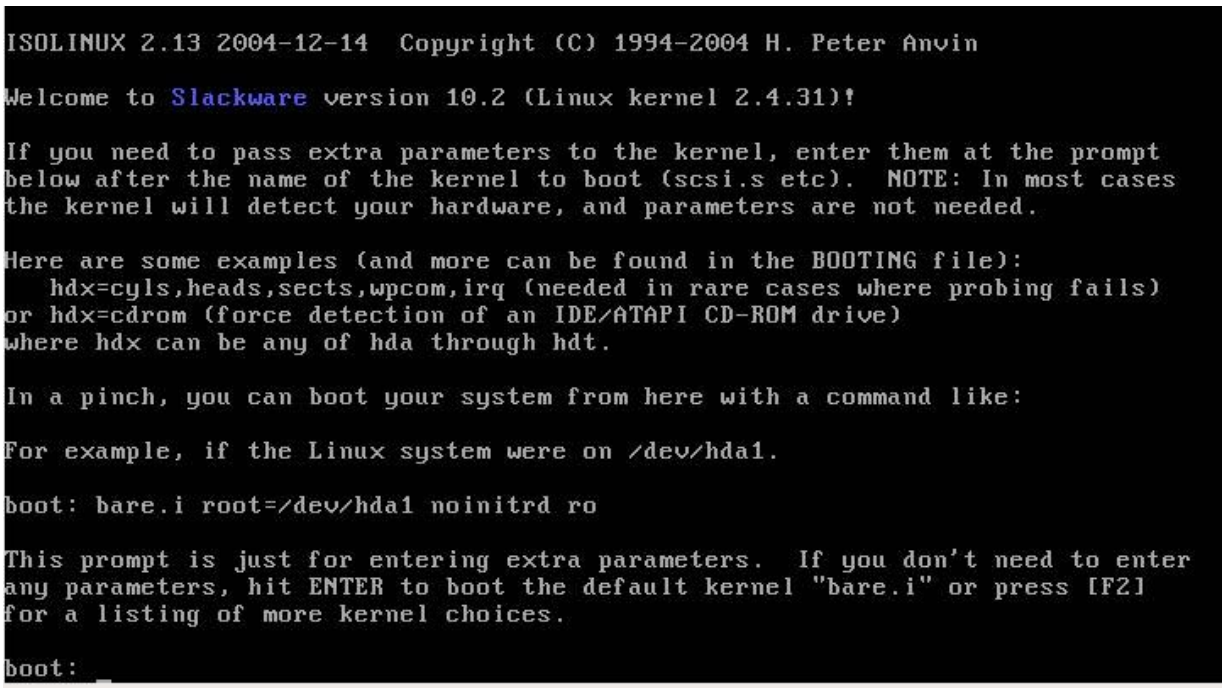
1.3.3 Métodos de instalación

Slackware Inc. ofrece la distribución oficial, consta de 4 CDs, el primero contiene el software necesario y básico para montar un servidor, el segundo es un "live CD" que permite correr Slackware sin necesidad instalarlo en el disco duro así como paquetes extras, el tercero y cuarto CD contienen el código fuente de todo Slackware y un útil libro electrónico que trata sobre Slackware a fondo.
Es necesario pagar $40 dólares por esta distribución oficial, sin embargo por ser GNU/Linux y estar bajo la licencia GNU es posible descargar Slackware de Internet, consta de dos CD con toda la funcionalidad de la versión oficial.

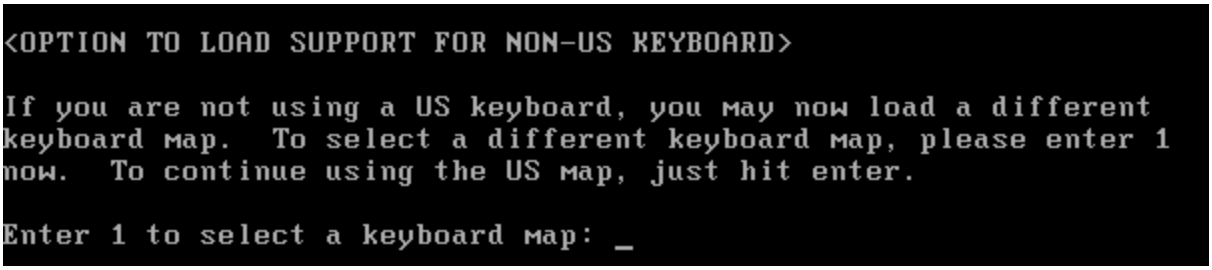
Los equipos de computo actuales permiten arrancar ("*boot*") un sistema operativo desde el CDROM, esto es posible si en el BIOS se ha indicado que el CDROM será el primero en arrancar, así que primero debemos asegurarnos que esto sea correcto.

Si por alguna razón el CDROM no permite el arranque, es necesario crear desde Windows dos discos (de 3½") de arranque: *bootdisk* y *rootdisk*. Esto se logra mediante la herramienta RAWRITE que se ejecuta desde línea de comandos: `D:\>RAWRITE [nombre_de_la_imagen] [destino]`.
Para más información se puede consultar el archivo `README.txt` que se localiza en el disco 1 en el directorio `<CDROM>:\isolinux\sbootmgr`.

La primer pantalla que veremos al iniciar el arranque será una pantalla de bienvenida y un *prompt* en donde se nos solicita parámetros extras para continuar con el arranque de GNU/Linux. Por lo general se presiona la tecla `<enter>` para arrancar con la imagen del kernel por default (`bare.i`).



Después de una serie de mensajes informativos que el kernel genera, se solicitara el tipo de teclado conectado a la computadora y finalmente se mostrará un *prompt* de *login*.



Para comenzar a particionar el disco duro y continuar con la instalación de GNU/Linux debemos entrar como superusuario (`root`).

1.3.4 Particionamiento del disco duro

En la arquitectura de las computadoras personales Intel y compatibles es necesario seccionar el disco duro; los discos duros se dividen en cilindros y los cilindros en sectores. Cada sector del disco duro tiene un tamaño de 512 bytes, el primer sector tiene dos importantes funciones: contiene la *MBR* (Master Boot Record) y la tabla de particiones. La MBR contiene una secuencia de comandos necesarios para cargar un sistema operativo, mientras que la tabla de particiones contiene la información necesaria para acceder al resto del disco duro (las particiones).

Debido al reducido tamaño de la MBR sólo es posible crear 4 particiones primarias, sin embrago, una partición primaria puede ser usada como partición extendida y contener un número ilimitado de particiones lógicas (por default el kernel limita a 59 particiones lógicas, pero se puede recompilar para aceptar más).

Para particionar el disco duro se debe invocar desde la línea de comandos la herramienta `fdisk disco_duro`.
Por lo general la instrucción para formatear el primer disco duro IDE es `fdisk /dev/hda`, si se tratara del segundo disco duro IDE sería `hdb`, etc., cuando se trata de discos SCSI el primer disco duro se llama `sda`, el segundo `sdb`, etc.. Por otra parte, la primer partición primaria de un disco IDE se nombra como `/dev/hda1`, mientras que la primer partición lógica se denomina `/dev/hda5`.

Es importante mencionar que cuando modificamos la MBR, los datos que se encuentran en el disco duro se perderán. Así que es recomendable respaldar toda la información contenida en el disco duro antes de modificar la MBR.

Cuando iniciamos la herramienta de `fdisk`, se mostrara un prompt propio de la herramienta.

Si insertamos el comando `m` apareceré un menú con las opciones disponibles. Con el comando `p` se mostraran las particiones existentes en el disco duro.

Lo primero que debemos hacer es eliminar todas las particiones primarias o extendidas existentes, insertamos el comando `d` y el número de partición para realizar esta tarea y lo repetimos por cada partición que tengamos.

Dependiendo de las necesidades y los requerimientos que se tengan, se puede planificar de muchas formas el particionamiento del disco duro para permitir el alojamiento de distintos sistemas operativos. Sin embargo, asumimos que el equipo será un servidor dedicado y sólo manejara GNU/Linux como único sistema operativo.

Creamos una partición extendida de aproximadamente 20GB y después creamos 5 particiones lógicas, esto se consigue tecleando el comando `n` y seleccionando la opción adecuada.

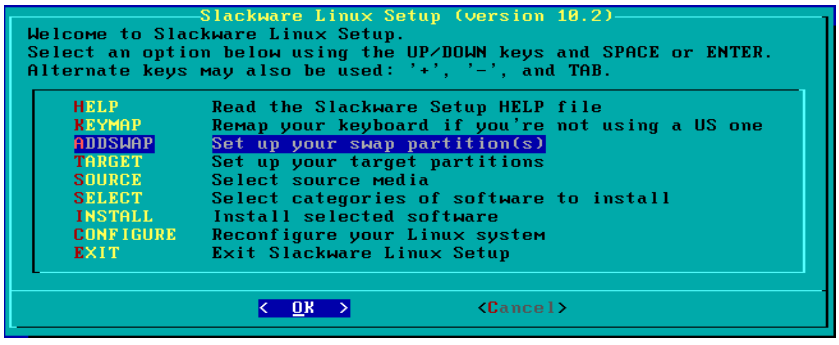
Partición	Tamaño
`swap`	512MB
`/`	4GB
`/home`	5GB
`/usr`	5GB
`/opt`	5GB

Cuando creamos una partición por default se le asigna el ID en hexadecimal de 83 (*linux native*), es necesario modificar el ID de la partición destinada para ser swap. Tecleando el comando `t`, seleccionando el número de la partición e insertando el nuevo ID, se modifica el ID de la partición, para el caso de las particiones swap el ID es el 82.

Para guardar los cambios realizados a la MBR y abandonar la herramienta se debe teclear el comando `w`.

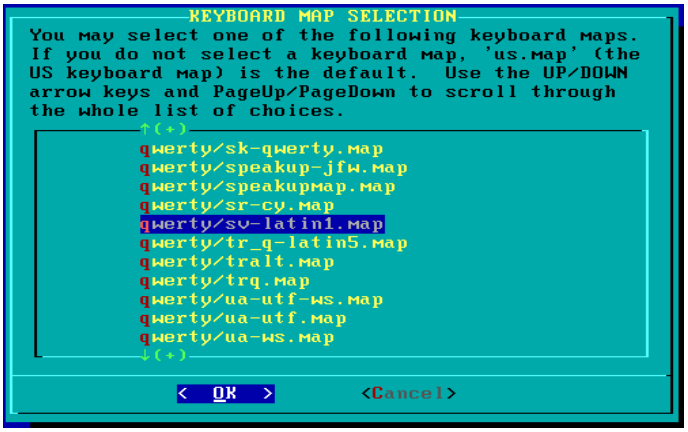
1.3.5 Asistente de instalación

Después de haber particionado el disco duro, debemos ejecutar el asistente de instalación de Slackware tecleando en la línea de comandos `setup`.



Éste asistente nos permitirá formatear la partición swap y el resto de las particiones, seleccionar la fuente de instalación (en nuestro caso será desde el CD-ROM), las series que deseamos instalar y finalmente configurar el sistema, red y el ambiente gráfico.

Una de las primeras tareas que el asistente de instalación realiza es la configuración del teclado, mediante las flechas de navegación recorremos una lista de opciones que se nos presentan y con la tecla `<enter>` seleccionamos el teclado correspondiente:



Si el particionamiento del disco duro se realizó de manera correcta, el asistente identificará la partición destinada para ser *swap* y la formateara.

La siguiente etapa, *TARGET*, se refiere a localizar las particiones donde deseamos alojar el sistema de archivos de GNU/Linux. En nuestro caso, cada partición alojara distintas partes del sistema de archivos por eficiencia y seguridad. El tipo de formato que se le pueden dar a estas particiones son los siguientes: reiserfs (por default), ext3, ext2, jfs, y xfs.

Para nuestro caso esto deberá corresponder a la siguiente tabla:

Punto de montaje	**Partición**	**Tipo de formato**
/	/dev/hda5	reiserfs
/home	/dev/hda6	reiserfs
/usr	/dev/hda7	reiserfs
/opt	/dev/hda8	reiserfs

La siguiente fase se denomina *SOURCE*, corresponde a la fuente que proveerá la serie de paquetes de instalación, seleccionaremos la primer opción: "*Install from a Slackware CD o DVD*".

El penúltimo paso antes de comenzar el proceso de instalación se denomina *SELECT*, debemos seleccionar las series de paquetes que requerimos. Por políticas de seguridad los servidores de producción sólo deben contar con un número mínimo de paquetes y servicios, esto excluye compiladores, software de desarrollo, juegos, entornos y herramientas gráficas.
En nuestro caso, instalaremos todos los paquetes disponibles por cada serie y en el futuro los retiraremos del sistema.
El último paso, *INTALL*, nos permite seleccionar uno a uno cada paquete y nos informa sobre el funcionamiento y uso de dicho paquete antes de instalarlo. Seleccionaremos la opción *full*.

1.3.6 Configuración del sistema

Al finalizar la instalación de los paquetes de cada serie, la siguiente tarea que el asistente realizará, será la configuración del sistema. Es preciso mencionar que la configuración del sistema que realizará el asistente no es la más adecuada ni la más avanzada, esta es una tarea más laboriosa y delicada, que requiere de la experiencia de un administrador de sistemas UNIX y una serie de políticas de seguridad. Sin embargo, es funcional y nos permitirá trabajar con GNU/Linux al terminar.

1.3.6.1 Instalación del kernel

En éste apartado se nos consultará si deseamos instalar el kernel en el disco duro o si preferimos cargarlo cada vez que encendamos la computadora desde un disco de 3½", dependiendo de las políticas de seguridad esto puede variar. En nuestro caso, seleccionaremos la opción "*Use a kernel from the Slackware*" para que el kernel que acompaña al CD de instalación se instale en el disco duro.

Después se nos pedirá que insertemos un disco de 3½" para hacer un disco de rescate, esto es de mucha utilidad si en algún momento el servidor falla y necesita iniciar desde un disco de rescate.

1.3.6.2 Zona horaria y reloj del sistema

Para muchas aplicaciones, herramientas y demonios, tener bien configurada la zona horaria del sistema y la fecha es una tarea crítica y necesaria.
Debemos buscar en una lista el país y la población en la que se encontrar trabajando nuestro servidor y colocar la fecha y hora adecuada.

1.3.6.3 Configurar el Mouse

En éste aparatado debemos buscar y seleccionar el tipo de mouse con el que cuenta la computadora, por lo general los equipos de cómputo actuales conectan el mouse al puerto PS/2; es posible que algunas funcionalidades no queden configuradas pero el mouse queda listo para usarse.

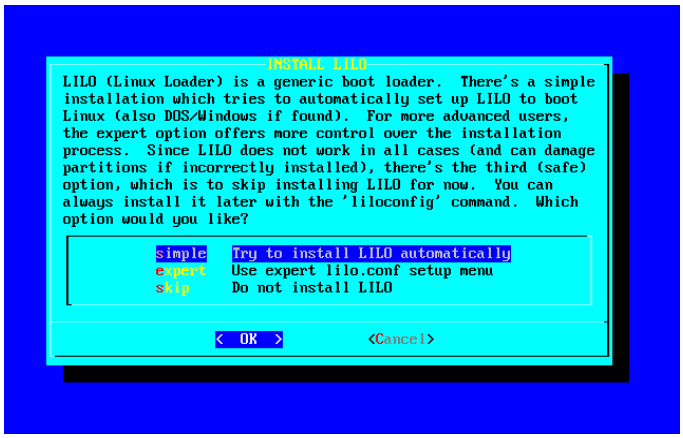
Toda la comunidad GNU/Linux sabe que el hardware más reciente no es soportado inmediatamente por GNU/Linux, sin embargo con el paso de los meses o días alguien logra desarrollar un *driver* para poder hacer uso de él. Un buen sitio en Internet donde se puede averiguar si determinado dispositivo es o no compatible con GNU/Linux es el siguiente: http://www.linuxcompatible.org/compatlist3.html

1.3.6.4 Configuración de LILO

Slackware ha optado por incluir LILO (LInux LOader) como herramienta predeterminada para configurar el arranque de GNU/Linux al encender el equipo. El asistente de instalación llamara a `liloconfig` para realizar esta tarea.

Se puede seleccionar la instalación automática, sin embrago el modo experto no es muy difícil de entender. En modo experto lo primero que debemos hacer es seleccionar la opción *Begin* para crear un nuevo archivo de configuración llamado `lilo.conf`, después agregamos la partición donde se encuentra ubicado el kernel linux y finalmente asignamos un nombre de etiqueta para identificarlo. Si existiera una partición DOS/Windows seleccionamos la opción DOS y hacemos los mismos pasos .

Finalmente, escribimos en la MBR a LILO.

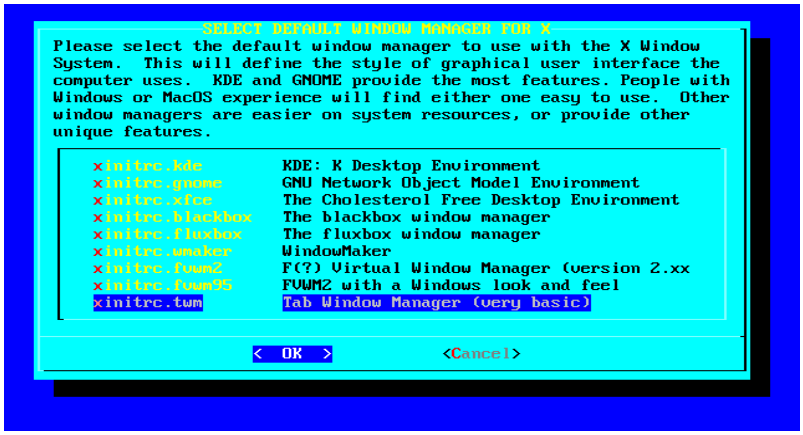


1.3.6.5 Configuración de la red

El asistente de instalación, solicitará algunos datos sobre la configuración de la red, como el puerto donde se localiza el modem o bien la tarjeta de red. Ademas solicitara el nombre del *host*, nombre de dominio, el servidor de nombres, el *gateway* y el tipo de conexión a Internet (DHCP, IP estática, etc). Esta información se puede modificar Después de haber completado la instalación con las herramientas adecuadas. Recomendamos no poner toda nuestra confianza en esta fase de la configuración de red pues si la tarjeta de red o modem no están soportados por el kernel, habrá que realizar una configuración más personalizada.

1.3.6.6 Configuración de X

Esta es la última fase de configuración que cubre el asistente de instalación de Slackware, en realidad nos permite seleccionar y habilitar el administrador de ventanas que deseamos lanzar al iniciar el sistema. Recomendamos seleccionar la opción `xinitrc.KDE`.
La configuración del ambiente gráfico, o los cambios que deseamos realizar, requiere de algunas herramientas como son: `xorgconfig`, `xorgsetup`, `xwmconfig`; eso se debe hacer como superusuario Después de haber concluido la instalación.



1.3.6.7 El superusuario: *root*

La última acción que realizará el asistente de instalación sera solicitar un *password* para el superusuario o `root`. El superusuario no tiene restricciones de ningún tipo, puede configurar el sistema, agregar más usuarios, instalar software, es el administrador del sistema operativo.

1.3.7 Acceso al sistema: *log in*

En un ambiente gráfico los elementos que nos permiten interactuar con el sistema operativo y los recursos son: ventanas, botones, barras de desplazamiento, iconos, etc.. En un ambiente de línea de comandos (*commandline environment*) la interacción se realiza mediante el *shell* o intérprete de comandos.

Para interactuar con el sistema operativo debemos tener una cuenta de acceso al sistema y un *password*, es responsabilidad del `root` otorgarnos estos dos requisitos.

Cuando encendemos la computadora y termina de cargarse GNU/Linux lo común es ver un *prompt* de *log in*, es aquí donde debemos introducir nuestro nombre de usuario y *password*.

Resumen

A lo largo de éste capítulo se abordo la importancia que tiene el software libre en los nuevos modelos de negocio y el papel que juega como alternativa de desarrollo e implementación cooperativa. El apoyo que ha tenido GNU/Linux por parte de grandes compañías de la industria del software y tecnologías de la información seguirá creciendo, ya sea a través de proyectos, fundaciones o patrocinios. Es un hecho que la forma actual de hacer negocios exige estándares abiertos, en los que cabe perfectamente GNU/Linux.

La primer actividad que se realizó durante el Diplomado de "Desarrollo e Implementación de Sistemas con Software Libre en Linux" fue la instalación, como ya se explicó, se realiza a través de asistentes de instalación. Los usuarios recién llegados al mundo de GNU/Linux encontraran un poco compleja la fase de instalación de Slackware, sin embargo, distribuciones como Fedora o SuSe están equipadas con asistentes gráficos y muy amigables.

En el siguiente capítulo nos enfocaremos a manipular el Sistema Operativo, mediante ordenes o comandos.

Capítulo 2.

Comandos básicos de Shell.

"En una máquina UNIX, root es el nombre del más divino de todos los usuarios, el que puede leer, borrar o alterar cualquier archivo, el que puede ejecutar cualquier programa, el que puede añadir cualquier usuario o eliminar a cualquiera de los existentes."

Stephenson, Neal. Criptonomicón: El código enigma

2.1 Comandos de ayuda: `man, info, whereis, which`

Existen una infinidad de comandos en GNU/Linux, difícilmente alguien podría memorizar todas y cada una de las opciones o parámetros que posee un comando. A continuación analizaremos algunas herramientas que nos ayudaran a recordar, o bien a descubrir, la utilidad de un comando así como los parámetros que son validos para dicho comando.

man, es un herramienta que nos proporciona la documentación de un comando a través de un manual de páginas de ayuda. Estas páginas fueron creadas por el autor del comando, o bien, por la gran comunidad de GNU/Linux.

La página de ayuda de un comando se divide en 9 secciones:

1. User commands
2. System calls
3. Libc calls
4. Devices
5. File formats and protocols
6. Games
7. Conventions, macro packages and so forth
8. System administration
9. Kernel

Algunas veces por razones de espacio en disco duro o seguridad, el administrador de un sistema no instala todas las páginas de ayuda, y sólo deja las páginas más recurrentes. Para verificar que páginas están disponibles basta con listar el contenido del directorio `/usr/share/man`.

Para leer la documentación correspondiente a un comando la sintaxis es la siguiente:

```
man nombre_del_comando
```

Entonces se desplegará la información detallada del comando; basta con pulsar las teclas de flecha o las teclas `j`, `k`, `i`, `l` para navegar en la página.

Si ejecutamos `man -k palabra_clave` se mostrará todas las coincidencias de dicha palabra dentro de las páginas.

info, es muy similar a man. Su sintaxis es: `info nombre_del_comando; info` trabaja con entidades que denomina nodos, un nodo es una pieza de información que contiene hipervinculos a otras páginas de `info`.

La navegación a través de la documentación se realiza con las siguientes teclas:

`<espacio>`	Siguiente pantalla de texto
`<del>` o `<bs>`	Página anterior
`n`	Siguiente nodo
`p`	Nodo anterior
`u`	Ir hasta arriba del nodo
`b`	Ir al inicio del nodo
`e`	Ir al final del nodo
`s`	Buscar una cadena en el nodo actual
`?`	Ir a la ayuda
`l`	Abandonar la ayuda y volver al nodo
`q`	Salir de `info`
`<tab>`	Saltar hasta las siguiente referencia

`whereis`, es de gran utilidad cuando necesitamos localizar la ruta absoluta de un comando o un archivo binario, así como la ruta de la documentación de dicho comando. La sintaxis es la siguiente:

```
whereis nombre_del_comando [nombre_del_comando_2] ... [nombre_del_comando_n]
```

`which` es muy similar a `whereis`, muestra la ruta absoluta donde podemos localizar un comando o un archivo binario. Su sintaxis es:

```
which nombre_del_comando [nombre_del_comando_2] ... [nombre_del_comando_n].
```

2.2 Comandos para la manipulación de archivos y directorios

Los usuarios que han utilizado durante mucho tiempo MS-DOS/Windows encontrarán en GNU/Linux muchas ventajas al manipular archivos y directorios a través de comandos. A veces mover, copiar o eliminar uno o varios archivos de un lugar a otro se facilita con sólo teclear un par de caracteres en vez de dar varios clicks con el mouse.

GNU/Linux, al igual que todos los sistemas operativos que descienden de UNIX, manejan 3 tipos de archivos:

- Archivos ordinarios: pueden contener texto en ASCII (entendible por el usuario), o información codificada (entendible por la computadora).
- Directorios: contienen información que el sistema necesita para acceder a todos los tipos de archivos.
- Dispositivos o archivos especiales: representan dispositivos del sistema como discos duros, scanners, impresoras, etc..

Si recordamos el formato 8.3[1] de MS-DOS para nombrar archivos, GNU/Linux no tiene un formato para nombrar un directorio o archivo. Sin embargo, el nombre debe cumplir con las siguientes características:

- Debe describir al archivo o directorio.
- Debe usar sólo caracteres alfanuméricos: mayúsculas, minúsculas, números, `@`, `_`
- Es recomendable no utilizar espacios en blanco
- No debe comenzar con los caracteres `+` y `-`
- No debe contener metacaracteres: *?|><;!{}[] ' "()
- Un archivo es oculto si inicia con `.`
- El máximo número de caracteres es de 255

2.2.1 Navegación: `ls`, `cd`, `pwd`

El comando **`ls`** lista el contenido de un directorio, cuanta con un gran número de parámetros. La sintaxis básica es:

```
ls [-parámetros] directorio_1 [directorio_2] ... [directorio_n]
```

Los parámetros más utilizados son:

`-l` listado extenso, muestra información que describe a cada archivo o directorio.
`-a` muestra todos los archivos incluyendo a los ocultos.
`-t` lista ordenando por la modificación en la fecha del archivo.
`-R` lista de manera recursiva, es decir, los subdirectorios contenidos en el directorio.

Para cambiar de directorio basta con teclear el comando **`cd`**. Si sólo se teclea `cd` volvemos al directorio del usuario (home directory). La sintaxis es:

```
cd nombre_del_directorio.
```

Existen algunos atajos:
`cd ..` Va un directorio arriba
`cd` Va al directorio del usuario
`cd ~/` Va al directorio del usuario

El comando **`pwd`** (print working directory) muestra la ruta completa del directorio en el que nos encontramos ubicados.

2.2.2 Creación: `touch` y `mkdir`

El comando **`touch`** tiene dos propósitos, crear un archivo vació o actualizar la fecha de modificación de un archivo. Si el archivo ya existe sólo cambia la fecha de modificación.
La sintaxis es:

```
touch [-parámetros] archivo_1 [archivo_2] ... [archivo_n]
```

1 8 caracteres para el nombre y 3 para la extensión

Algunos de los parámetros son:
`-c` Si el archivo existe simplemente no lo crea.
`-t [YY]MMDDhhmm` Coloca la fecha de modificación que especificada.

La creación de un directorio se realiza con el comando **`mkdir`**. La sintaxis es:

```
mkdir [-parámetros] nombre_del_directorio.
```

Los parámetros más utilizados son:

`-p dir1/dir2/dir3`	Crea toda la estructura de subdirectorios simultáneamente
`-m modo`	Crea el directorio con los permisos que indica `modo`.

2.2.3 Copiar y mover: `cp` y `mv`

El comando **`cp`** permite copiar uno o varios archivos de un directorio fuente a un destino. También permite copiar un archivo y cambiar su nombre al mismo tiempo o copiar directorios completos. La sintaxis de éste comando es:

```
cp [-parámetros] archivo_1 [archivo_2] ... [archivo_n] directorio_destino
```

Algunos de los parámetros más usados son:
`-i` Copia interactiva, el usuario debe confirmar la operación.
`-R` Copia recursiva, copia todo el contenido de un directorio.

Para mover un directorio o archivo se usa el comando **`mv`**. También sirve para cambiar el nombre a un archivo.

Los dos parámetros más empleados son:
`-f` Forzar la operación, evita el modo interactivo.
`-i` Modo interactivo, el usuario debe confirmar la operación.

Como se ha podido observar, los comandos `cp` y `mv` trabajan sobre directorios y archivos.

2.2.4 Borrar: `rm` y `rmdir`

Cuando se elimina un archivo y directorio no hay forma de recuperarlo, es por eso que se debe estar muy atento al realizar éste tipo de operaciones.

El comando **`rm`** nos permite eliminar archivos o directorios. Es mucho muy flexible ya que cuenta con más parámetros que el comando `rmdir`. Su sintaxis es:

```
rm [-parámetros] archivo_1 [archivo_2] .. [archivo_n]
```

Los parámetros más utilizados son:

`-i` Modo interactivo, el usuario debe confirmar la operación.
`-r` o `-R` Elimina de manera recursiva, elimina todo el contenido del directorio.
`-f` Forzar la operación, evita el modo interactivo.

rmdir nos permite eliminar un directorio siempre que éste esté vació, de lo contrario mostrar un mensaje y no se concluirá la operación. La sintaxis de éste comando es:

```
rmdir [-parámetros] directorio_1 [directorio_2] ... [directorio_n]
```

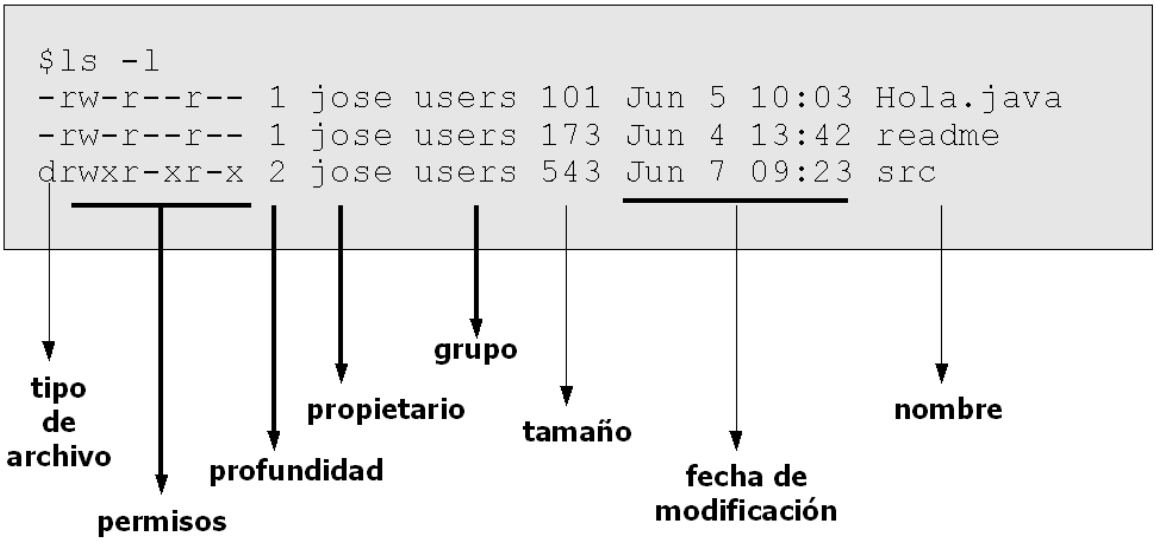
El parámetro `-p` nos permite eliminar todos los subdirectorios contenidos.

2.3 Permisos

Desde sus orígenes los arquitectos de UNIX pensaron en la seguridad, aunque no como la que conocemos en la actualidad, se diseño de modo que cada archivo, directorio y dispositivo tuviera permisos de escritura, lectura y ejecución. Todo esto con el fin de controlar el uso de los recursos y administrarlos de manera eficiente.

Con la llegada de la distribución UNIX de la Universidad de Berkeley (BSD) vinieron muchas mejoras para UNIX; se introdujeron mecanismos de seguridad como *password* cifrados[2], se implemento el protocolo TCP/IP, el manejo de memoria virtual paginado por demanda, entre otras. GNU/Linux ha adoptado lo mejor de las distintas distribuciones de UNIX, sin embargo hay características que no pueden ser portadas a GNU/Linux debido a que son funcionalidades con copyright y sin incompatibles con la licencia de GNU/Linux.

Si utilizamos el comando `ls -l` se mostraran una lista detallada de los atributos asociados a cada archivo o directorio, el campo ubicado más a la izquierda especifica los permisos del archivo; por ejemplo, el campo podría mostrar `-rw-r—r--` El primer carácter – corresponde al tipo de archivo y los nueve caracteres restantes representan los permisos de acceso a propietario, grupo y demás, respectivamente. Cada una de las categorías mencionadas ocupa tres caracteres en el campo de permisos, r para lectura, w para escritura y x para ejecución.



Para cambiar los permisos de un archivo o directorio se utiliza el comando **chmod**, cuya sintaxis es:

```
chmod <modo> archivo_1 [archivo_2] ... [archivo_n]
```

2 http://www.tldp.org/HOWTO/Shadow-Password-HOWTO-2.html

donde existen dos formas de escribir el modo: simbólica y octal.

Forma simbólica	***Forma octal***
`chmod <quien operador que> <file(s)>`	`chmod NNN <archivo(s)>`
`<quien>` : **u** para el propietario. **g** para el grupo. **o** para los demás. **a** para todos (propietario, grupo y demás). `<operador>`: **+** para agregar el permiso. **-** para eliminar el permiso. **=** para limpiar los permisos y colocar uno en especifico. `<que>`: puede ser la combinación de **r**, **w** y **x**.	Aquí cada terna de permisos adquieren los siguientes valores octales: **r** = 4 **w** = 2 **x** = 1 La suma de cada terna se concatena y obtenemos el permiso completo. Por ejemplo, el permiso para el campo `rwxr-x---` esta representado por 750: `rwx` = 4+2+1=7 `r-x` = 4+0+1=5 `---` = 0+0+0 = 0

2.3.1 Usuarios y Grupos

Existe una jerarquía de usuarios en GNU/Linux, el usuario que se encuentra en lo alto de la jerarquía se conoce como `root`, o superusuario.
Éste usuario no tiene ninguna restricción para realizar cualquier modificación al sistema.

Los elementos básicos de una cuenta de usuario son los siguientes:

- Nombre del usuario (*login*). Se trata generalmente de una abreviatura del nombre real del usuario.
- Contraseña. Se trata de una palabra clave que sólo el usuario debe conocer, junto con el *login* le permiten a l usuario accesar al sistema.
- UID. Número irrepetible que identifica al usuario dentro del sistema.
- GIU. Número que identifica al usuario dentro de un grupo.
- Descripción. información que describe al usuario, por lo general es el nombre completo.
- Directorio del usuario. Se trata del directorio asignado para el usuario.
- Shell. Programa con el cual el usuario se comunica con el sistema.

El encargado de crear nuevos grupos y dar de alta nuevos usuarios es el superusuario.

Un grupo es un conjunto de usuarios, cada usuario es miembro de al menos un grupo (grupo primario o *primary group*) y puede ser miembro de otros (*group set*).

Para agregar un grupo se emplea el comando **`groupadd`**, la sintaxis es la siguiente:

```
groupadd [-g gid][-n][-r][-f] nombre_del_grupo
```

Donde el parámetro `-g gid` indica cual sera el número que identificara a ese grupo. Los grupos se almacenan en el archivo `/etc/group`.

Para eliminar un grupo basta teclear el comando **groupdel** nombre_del_grupo.

Existen dos comandos para dar de alta nuevos usuarios: **adduser** y **useradd**. Dependiendo del archivo de configuración /etc/login.defs, el comando useradd crea el directorio del usuario (*home directory*).

La sintaxis useradd es:

```
useradd [-c comment] [-d home_dir]
    [-e expire_date] [-f inactive_time]
    [-g initial_group] [-G group[,...]]
    [-m [-k skeleton_dir] | -M] [-n] [-o] [-p passwd]
    [-s shell] [-u uid] login
```

donde cada parámetro tienen el siguiente significado:

-c comment	Comentario acerca del usuario.
-d home_dir	Directorio del usuario (home directory).
-e expire_date	Fecha en la cual expira el *password* del usuario y se bloquear la cuenta, el formato es YYYY-MM-DD.
-f inactive_days	Días que deben de transcurrir, después de que expira el *password*, para eliminar la cuenta. El valor 0 bloquea la cuenta en cuanto expira el *password*, mientras que el valor -1 deshabilita éste parámetro.
-G group,[...]	Lista de los grupos a los que pertenecerá el usuario.
-g initial_group	El nombre o número del grupo primario.
-m	Forzar la creación del directorio de usuario (*home directory*) sino existe, los archivos. Con el parámetro -k los directorios contenidos en skeleton_dir o /etc/skel serán creados dentro del directorio del usuario.
-M	El directorio del usuario no será creado al menos que éste indicado en el archivo de configuración /etc/login.defs.
-n	Crea un grupo con el mismo nombre del usuario.
-o	Permite crear usuarios con un UID duplicado.
-p	El sistema solicitara una contraseña para el nuevo usuario.
-s	Nombre del *shell* que utilizará el usuario.
-u uid	El valor numérico que identificara al usuario, el valor debe ser único al menos que se use el parámetro -o.

El comando adduser suele ser un comando interactivo, aunque algunos sistemas hacen que sea un enlace a useradd.

Finalmente, para dar de baja a un usuario se emplean los comandos: **userdel** y **usermod**. **userdel** elimina al usuario del sistema (si se agrega el parámetro -r elimina todo el contendido del directorio del usuario). El comando usermod, sirve para bloquear o desbloquear a un usuario, o bien para modificar el perfil del usuario.

2.4 Procesos

La definición más simple de proceso se refiere a un programa que está en ejecución. El kernel Linux fue diseñado para corre múltiples procesos y mantiene una tabla interna denominada tabla de procesos (*process table*) en la que mantiene información acerca de los procesos.
Cada proceso tiene su propio ambiente de proceso (*process enviroment*) donde se puede encontrar la siguiente información:

- Nombre del programa
- Datos internos
- Archivos abiertos
- Directorio donde se ejecuta
- Parámetros adicionales
- ID (identificador) de usuario y grupo
- ID del proceso
- ID del proceso padre
- Variables del programa

Todos los procesos son iniciados por otro proceso, a excepción del proceso `init` que es iniciado por el kernel y siempre tiene el PDI 1.

En general, los proceso no pueden corre por siempre, pueden ser terminados por dos razones:

- El proceso termina por si mismo, ya sea automáticamente (cuando terminan su trabajo) o cuando ejecuta los parámetros de entrada de finalización que el usuario proporciono.
- Cuando otro proceso le envía una "señal de finalización" al proceso.

2.4.1 Procesos en primer plano y segundo plano: *foregrounding y backgrounding*

Los procesos que se ejecutan desde un indicador de comandos (*prompt*) y requieren de la interacción con usuario se denominan procesos en primer plano (*foreground process*). Estos procesos no permitirán ejecutar otro proceso hasta que finalicen.
Los procesos que se ejecutan de manera independiente a la línea de comandos se denominan procesos en segundo plano (*background process*).

Todos los procesos pueden ser ejecutados en segundo plano, basta con agregar el carácter **`&`** al final de de la invocación del programa o comando.
Por ejemplo, si ejecutamos el comando `find` como se muestra a continuación, `$find / -name README &` , obtendremos un mensaje similar a éste:

```
[1]             462
```

Donde la primer columna se refiere al número de procesos que tenemos ejecutando en segundo plano mientras que la segunda columna indica ID del proceso.

Cuando se ejecuta un proceso en primer plano, es posible suspender el proceso en ejecución presionando `<Ctrl-z>`, esto no lo termina pero permite reiniciarlo con el comando **`bg`** `id_proceso`.

Es de gran utilidad el comando **`nohup`** para ejecutar procesos en segundo plano y mantener su ejecución después de que salgamos de la sesión.

Con el comando **jobs** es posible listar los procesos que corren en segundo plano, aunque los procesos que se han iniciado con el comando `nohup` no se listaran.
Para poner en primer plano un proceso basta con teclear el comando **fg** `número_de_proceso`.

2.4.2 Monitoreo de procesos: `ps, pstree` y `top`

El comando **ps** informa sobre el estado de los procesos que se ejecutan, cuando se ejecuta sin parámetros muestra sólo los procesos que se ejecutan en la terminal actual. Debido a que es un comando con muchos parámetros se recomienda consultar las páginas del manual de `ps`, sin embargo, la invocación más común es: `ps aux`.

Una forma alternativa de listar los procesos es **pstree**, esto mostrara un árbol de procesos en donde la raíz siempre será el proceso `init`.

El comando **top**, informa sobre la memoria, carga promedio del sistema y los proceso en ejecución. Todo esto en tiempo real.

2.4.3 Terminar procesos: `kill`

Como ya se ha mencionado, un proceso que se ejecuta en primer plano puede ser suspendido pulsando la combinación de teclas `<Ctrl-z>`. Es posible interrumpirlo en su totalidad pulsando `<Ctrl-c>`, sin embargo contamos con el comando **kill** y **killall** para modificar el estado de los procesos.

La sintaxis del comando `kill` es la siguiente:

```
kill -señal PID
```

mientras que la sintaxis del comando `killall` es:

```
killall -señal aplicación.
```

La única diferencia entre `kill` y `killall`, es que `killall` termina con todos los procesos hijos que haya generado un proceso padre.
Para ambos comandos debe ser proporcionada una señal, las siguientes son las señales más importantes:

Señal	***Combinación de teclas***	***Significado***	***Acción***
01		Suspender (*hangup*)	Finaliza el proceso
02	<Ctrl-c>	Interrumpir (*interrupt*)	Finaliza el proceso
03	<Ctrl-\>	Quitar (*quit*)	Finaliza el proceso y realiza un volcado de memoria
09		Matar (*kill*)	Finaliza el proceso inmediatamente
15		Terminar (*terminate*)	Finaliza el proceso

Para conocer el resto de las señales disponibles basta con teclear `kill -l`.

2.5 Vi, el editor de textos

vi es el editor de textos que ha acompañado a los sistemas UNIX desde su inicios. No se trata de un editor visualmente atractivo, pero es muy poderoso y funcional. Muchas herramientas como lectores de mail o lectores de noticias suelen emplear editores externos y recurren a vi.
Algunas distribuciones de GNU/Linux ha sustituido al tradicional vi por vim (*vi improved*, vi mejorado).

2.5.1 Modos de vi

Vi emplea un buffer para editar los archivos y cuenta con dos modos de operación: Modo comandos y Modo edición, aunque hay quienes afirman que existe un tercer modo, el modo *ex* (para procesar comandos complejos de vi). Es por esta razón que muchos usuarios encuentra al editor vi un tanto complejo de manejar.

2.5.2 Iniciar vi

Para iniciar vi basta teclear en el prompt **vi** [nombre_del_archivo], si se ha pasado como parámetro el nombre de un archivo vi copia el contenido en el buffer y esta listo para ser editado, de lo contrario el buffer esta limpio para editar.

Al iniciar vi, se muestra en cada línea del buffer el carácter **~**, éste no formará parte de nuestro texto pero son un indicativo de que nos encontramos en modo comando.

2.5.3 Navegación en modo comandos

Para movernos a través del texto es necesario estar en modo comandos, la siguiente tabla nos muestra las teclas de navegación:

Teclas (comandos)	***Acción***
<flecha izquierda> o h	Desplazarse un carácter a la izquierda
<flecha derecha> o l	Desplazarse un carácter a la derecha
<flecha arriba> o k	Desplazarse una línea arriba
<flecha abajo> o j	Desplazarse una línea abajo
^	Ir al inicio de la línea
$	Ir al final de la línea
1G	Ir a la primer línea
G	Ir a la ultima línea

2.5.4 Editar el texto en modo comandos

Para editar texto en modo comandos, tenemos los siguientes comandos:

Teclas (comandos)	*Acción*
x	Borra el carácter que esta arriba del cursor.
X	Borra el carácter que esta a la izquierda del cursor.
r	Remplazar un carácter
u	Deshacer la ultima acción
.	Repetir el ultimo comando
j	Unir dos líneas

2.5.5 Editar el texto en modo edición

Para editar texto en modo edición, tenemos los siguientes comandos:

Teclas (comandos)	*Acción*
l	Insertar texto al inicio de la línea
i	Insertar texto antes del cursor
a	Añadir texto después del cursor
A	Añadir texto al final de la línea

Para volver a modo comando es presionar la tecla <Esc>.

2.5.6 Búsqueda y remplazo de patrones

Para buscar una secuencia de caracteres o cadenas (patrones) debemos encontrarnos en modo comandos e introducir el siguiente comando: /<patrón>. Si deseamos repetir la búsqueda sólo tecleamos n.

Remplazar un patrón por otro (entiéndase una cadena por otra), se realiza en modo *ex* (finalmente modo comandos), debe introducirse la siguiente expresión: :%1,$s /patrón1/patrón2/g.
El carácter ":" indica que la acción se ejecutara en modo *ex*.

- "1,$" indican que el comando comenzara desde la primer línea y afectara hasta la ultima línea. Si deseamos restringir en un rango de líneas, por ejemplo de la línea 1 a la 5, se debería escribir "1,5". El carácter "%" indica que afectara todo el archivo.
- "s" significa que se realizara una búsqueda y una sustitución.
- El primer "/" indica el inicio de la expresión y el inicio del patrón a buscar, el segundo "/" indica el final del primer patrón y el inicio del patrón con el que sustituiremos y finalmente el tercer "/" indica el final de la expresión.
- "g" asegura que no sólo la primer coincidencia sea afectada, sino que toda coincidencia encontrada en el archivo sea afectada.

Existen muchos comandos que son útiles en el modo el modo *ex*, así que se recomienda consultar las páginas del manual de vi.

2.5.7 Copiar, cortar y pegar texto

Para copiar la línea en la que se encuentra ubicado el cursor, debemos asegurarnos de estar en modo comandos y teclear **`yy`**. Así la línea se almacena en un buffer y estará lista para otra acción.

Para cortar una línea debemos teclear **`dd`**, si lo que deseamos es cortar la palabra en la posición actual del cursor tecleamos **`dw`**.

Cortar o copiar múltiples líneas se debe anteponer a los comandos que ya vimos el número de líneas que deseamos procesar desde la posición actual del cursor, por ejemplo: **`3dd`** cortara tres líneas, **`3yy`** copiara tres líneas.

Finalmente, con el comando **`p`** pegamos el texto en la posición en la que se encuentre el cursor.

2.5.8 Guardar y salir de `vi`

Si introducimos, en modo comandos, el comando **`ZZ`** éste nos permitirá guardar los cambios realizados y cerrará `vi`; sino se le proporciono a `vi` como parámetro el nombre del archivo, en éste momento se solicitar un nombre para el archivo. Una manera análoga de hacer esta operación (en modo *ex*) es con el comando **`:wq`** o **`:x`**.
Con el comando **`:q!`** Abandonamos de manera tajante `vi` sin guardar los cambios.

Resumen

A través de éste capítulo se presentaron los comandos básicos para la manipulación de archivos, administración de recursos y edición de textos. Existe una diversidad de intérpretes de comandos, algunos comandos están disponibles a la mayoría de los shells y otros son exclusivos de cada intérprete de comandos. Los comandos que analizamos pertenecen a shell `bash`.

Parecerá absurdo trabajar con comandos cuando se tiene avanzadas interfaces gráficas en GNU/Linux, pero la mayor parte de la administración de los sistemas GNU/Linux se realiza a través de comandos en modo texto y de manera remota. Los comandos son herramientas poderosas y flexibles que le permiten al usuario realizar tareas complejas tecleando unos cuantos caracteres.

No es sencillo aprender todos los comandos disponibles en un shell, por eso existen las paginas de ayuda y una gran cantidad de documentación. Los mismos comandos proporcionan una breve ayuda con los parámetros `-h` o `-?`.

En el siguiente capítulo nos centraremos en la programación de páginas web con PHP y HTML.

Capítulo 3.

Páginas Web dinámicas con PHP y HTML.

"El lenguaje es un espejo de la mente en un sentido profundo y significativo: es un producto de la inteligencia humana, creado de nuevo en cada individuo mediante operaciones que se encuentran más allá del alcance de la voluntad o la conciencia. "

Chomsky, Noam. Sobre la capacidad cognitiva

3.1 ¿Qué es HTML?

En el año de 1962 el concepto de Internet fue concebido por el Departamento de la Defensa de los Estados Unidos. En los primeros años de los noventas el físico inglés Tim Berner-Lee, que trabajaba para el Laboratorio Europeo de Física de Partículas (CERN), propuso una forma de comunicación entre computadoras que facilitaba a los investigadores de todo el mundo conocer los resultados recientes de las investigaciones que ahí se realizaban. Su propuesta es lo que hoy conocemos como HTTP y HTML.

Las computadoras conectadas a la Internet suelen usar el protocolo de comunicación HTTP (HyperText Transfer Protocol) para compartir información en forma de páginas Web. Estas páginas Web están almacenadas en un servidor para que los usuarios (clientes) puedan accesar a ellas.

HTML, es el acrónimo inglés de HyperText Markup Language (lenguaje de marcado de hipertexto), se trata de un lenguaje utilizado para desarrollar páginas Web y documentos. No se trata de un lenguaje de programación como C++ o Java, realmente se trata de un lenguaje de marcado que implementó las especificaciones de SGML[1] conforme al estándar internacional ISO 8879.

Los documentos HTML están formados por un conjunto de etiquetas especialmente intercaladas escritas en texto plano (ASCII). Son interpretadas por los navegadores Web que formatean y muestran los documentos. Un conjunto de documentos HTML pueden estar entrelazados mediante enlaces, en ingles *hyperlinks* o *links.*

Los documentos HTML son almacenados en la computadora con la extensión `.html` o `.htm`.

XHTML es el acrónimo inglés de *eXtensible Hypertext Markup Language*, es una versión XML[2] de HTML, el objetivo es hacer de HTML un lenguaje de marcado formal y estandarizado bajo las reglas de XML. Es una recomendación que el W3C[3] lanzo en enero de 2000 con el fin de remplazar gradualmente a HTML.
Es compatible con la versión HTML 4.01 y los navegadores web más recientes lo soportan sin ningún problema.

XML es un lenguaje de marcado que permite definir otros lenguajes de marcado según las necesidades del usuario. Es un lenguaje que no especifica ni las etiquetas, ni la gramática del lenguaje, es por eso que se le dice lenguaje extensible. Fue desarrollado por el W3C, y su objetivo principal es describir la información y no la presentación como es el caso de HTML.

1 Standard Generalized Markup Language
2 eXtensible Markup Language, lenguaje de marcas extensible
3 World Wide Web Consortium, organización que produce estándares para la World Wide Web

3.1.1 Estructura de una página Web

Un documento HTML, o página web, está formado por etiquetas y cada etiqueta tiene o puede tener atributos. Las etiquetas se identifican fácilmente en una página web, pues se encuentran contenidas entre los símbolos <>. Una etiqueta por lo general cierra con su contra parte </>, por ejemplo:

<etiqueta> Mi Información </etiqueta>

Los navegadores web son capaces de interpretar las etiquetas si están escritas en mayúsculas o en minúsculas, sin embrago, una recomendación del W3C para XHTML especifica que cada etiqueta debe cerrar y estar escrita en minúsculas.

Las etiquetas se suelen anidar; la forma correcta de cerrar cada una, es cerrar la ultima etiqueta abierta, es decir:

```
<etiqueta1>
<etiqueta2> <etiqueta3> Mi información </etiqueta3> </etiqueta2>
</etiqueta1>
```

Los atributos que una etiqueta puede contener se definen siempre en la etiqueta de apertura y se encierra entre comillas dobles:

<etiqueta atributo="valor"> Mi Información </etiqueta>

Existen etiquetas que no necesariamente requieren de una etiqueta de cierre, pero de acuerdo a las recomendaciones del W3C para XHTML, es preferible que éste tipo de etiquetas sean de la forma:

<etiqueta atributo="valor" />

donde el cierre de la etiqueta lo marca el / al final de la etiqueta.

La estructura básica de un documento XHTML consiste en tres partes:

- Tipo de documento (<!DOCTYPE>)
- Encabezado (<head>)
- Cuerpo (<body>)

3.1.1.1 La etiqueta <!DOCTYPE>

La etiqueta <!DOCTYPE> debe ser la primer etiqueta de un documento XHTML, le indicará al navegador web la versión de HTML corresponde al documento y las reglas sintácticas que un DTD ha definido para el documento.

Existen tres tipos de documentos XML que corresponden a tres tipos de DTDs: Strict, Transitional, y Frameset.

<!DOCTYPE>	Uso
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">	Strict. Se emplea cuando se desea un documento limpio de errores, es decir, sintácticamente correcto. También se emplea cuando el documento hace uso de Hojas de estilo en cascada, CSS.

<!DOCTYPE>	Uso
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">	Transitional. Se emplea cuando se desea emplear las características propias de HTML y asumimos que el navegador web no soporta las hojas de estilo en cascada.
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">	Frameset. Se emplea cuando deseamos utilizar marcos, frames, para dividir la ventana del navegador en varios marcos.

3.1.1.2 Entidades o caracteres de escape.

Existen algunos caracteres que tienen un significado especial para HTML, tales como < o >, y cuando son introducidos como parte de un texto el navegador podría tener problemas para mostrarlos. La manera de representar éstos caracteres es a través de entidades o caracteres de escape.
Los caracteres de las lenguas europeas, junto con los caracteres especiales, están representados en el estándar de codificación ISO 8859-1[4](Latin-1).

Los siguientes son los caracteres de escape más utilizados en la lengua española.

Carácter	Nombre de la entidad	Número de la entidad
"	"	"
'	'	'
&	&	&
<	<	<
>	>	>
á	á	á
Á	Á	Á
é	é	é
É	É	É
í	í	í
Í	Í	Í
ó	ó	ó
Ó	Ó	Ó
ú	ú	ú
Ú	Ú	Ú
ñ	ñ	ñ
Ñ	Ñ	Ñ
ü	ü	ü

4 http://www.w3.org/TR/html401/sgml/entities.html

3.1.1.3 La etiqueta <html>

Ésta etiqueta se escribe inmediatamente después de la etiqueta <!DOCTYPE>, y especifica que el documento es un documento HTML, técnicamente es una redefinición de la etiqueta <!DOCTYPE>, sin embargo anida el resto de las etiquetas que conforman a la página web.

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

3.1.1.4 La etiqueta <head>

Esta etiqueta le proporciona más información al navegador web sobre el documento, ésta información no es mostrada al usuario, simplemente es procesada por el navegador. Entre las etiquetas que pueden ser anidadas dentro de la <head> están: <base>, <link>, <meta>, <script>, <style>, y <title>.

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

3.1.1.5 La etiqueta <title>

Es requerida desde la versión HTML 3.2, y ofrece un título para la página. Éste título se muestra en la barra de título del navegador.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
  <head>
  <title>
  Título de la página
  </title>
  </head>
</html>
```

3.1.1.6 La etiqueta <body>

Esta es la etiqueta que contiene la mayor parte de la información que aparecerá en el navegador web, tal como: texto, imágenes, texto, tablas, formularios, etc..

Después de la versión HTML 4.01 se desaprobaron los atributos que la etiqueta <body> podía contener, incluso la versión XHTML 1.0 no los soporta al emplear el *strict* DTD, esto es porque se trata de atributos de presentación y son las hojas de estilo las encargadas de proporcionarla.

Atributo	Valor	Descripción
alink	rgb(x,x,x) #xxxxxx nombre_color	El color del link activo.
background	nombre_archivo	La imagen que se mostrara como fondo.
bgcolor	rgb(x,x,x) #xxxxxx nombre_color	El color de fondo de la página.
link	rgb(x,x,x) #xxxxxx nombre_color	El color de los links presentes en la página.
text	rgb(x,x,x) #xxxxxx nombre_color	El color del texto.
vlink	rgb(x,x,x) #xxxxxx nombre_color	El color de los links visitados.

En adelante, sólo presentaremos los atributos que no han sido desaprobados.
A continuación mostramos un ejemplo de nuestra primera página web desplegándose en el navegador web Firefox.

hola.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title>Hola y adi&oacute;s</title>
</head>
<body>
Hola y adi&oacute;s
</body>
</html>
```

Salida

3.1.2 Dando formato al texto

Existen un número considerable de etiquetas que sirven para darle formato al texto de una página web, solamente veremos las más importantes, pero se recomienda visitar la página web de w3school[5] para una rápida referencia del resto de las etiquetas.

3.1.2.1 Encabezados: `<h1>` - `<h6>`

Las etiquetas de encabezado se emplean para crear títulos o subtítulos dentro del texto. Existen sólo seis etiquetas de encabezado que van desde la `<h1>` hasta la `<h6>`, siendo la etiqueta `<h1>` la que formatea el encabezado con el mayor tamaño de letra.

hola02.html

```
<!DOCTYPE ...>
<html>
<head><title>Hola y adi&oacute;s 2</title>
</head>
<body>
<h1>Hola y adi&oacute;s</h1>
<h2>Hola y adi&oacute;s</h2>
<h3>Hola y adi&oacute;s</h3>
<h4>Hola y adi&oacute;s</h4>
<h5>Hola y adi&oacute;s</h5>
<h6>Hola y adi&oacute;s</h6>
</body>
</html>
```

Salida

5 http://www.w3schools.com

3.1.2.2 Párrafos y saltos de línea: <p>,

Los párrafos se emplean justamente para formatear al texto y darle una buena distribución. Todos los atributos de presentación de ésta etiqueta fueron desaprobados en HTML 4.01, y como sucede con muchas etiquetas, es responsabilidad de las hojas de estilo dotar de presentación a ésta etiqueta.
Se conservan los atributos estándar `id`, `class`, `title`, `style`, `dir`, `lang`, `xml:lang`.

Los saltos de línea se indican con la etiqueta `<br>`, se trata de una etiqueta sin atributos y no tiene su contra parte de cierre, de modo que cierra de la forma `<br />`.

3.1.2.3 Estilo en la fuente: <b>, <i>, <tt>, <big>, <small>

A pesar de que se trata de etiquetas de presentación, no han sido desaprobadas y por lo tanto se conservan.

Etiqueta	Descripción
`<b>`	Texto en negritas
`<big>`	Texto más grande
`<code>`	Texto tipo código de computadora
`<cite>`	Texto a manera de cita
`<em>`	Texto enfatizado
`<kbd>`	Texto tipo teclado
`<i>`	Texto en cursiva
`<sub>`	Texto como subíndice
`<sup>`	Texto como superíndice
`<small>`	Texto más pequeño
`<tt>`	Texto tipo teletipo

3.1.2.4 Listas: <ul>,

Existen dos etiquetas que permiten enlistar información. Una lista enumerada se logra mediante la etiqueta `<ol>`, mientras que una lista sencilla se representa con la etiqueta `<ul>`. En ambos casos la información a enlistar se representa con la etiqueta `<li>`.

listas.html

```
<!DOCTYPE ...>
<html>
<head><title>Listas</title>
</head>
 <body>
 <ul>
  <li>Cafe</li>
  <li>Jugo</li>
  <li>Agua</li>
 </ul>
 <ol>
  <li>Cafe</li>
```

```
    <li>Jugo</li>
    <li>Agua</li>
   </ol>
  </body>
</html>
```

Salida

3.1.2.5 Hipervínculos o links : <a>

Los hipervínculos o links, son una forma de enlazar una página web con otros recursos disponibles en la red. Lo común es vincular páginas web con otras páginas web, sin embargo se pueden vincular recursos como: imágenes, video, audio, applets, etc..

Los vínculos se realizan especifican a través de la etiqueta `<a>`, que consta de 3 partes:

- La apertura de la etiqueta `<a>` con los atributos correspondientes, generalmente `href="url"`.
- Una dirección (URL) que le indica al navegador hacia donde ir a buscar el recurso solicitado.
- El texto o recurso entre la apertura y cierre de la etiqueta.

Ahora mostramos como puede ser enlazada una página web a través de un texto o una imagen. Es importante ver como se emplea el *URL* del atributo `href`, pues el primer y segundo enlace se refieren a un documento externo, mientras que el tercer enlace se refiere a un documento dentro del mismo directorio.

links01.html

```
<!DOCTYPE ...>
<html>
<head><title>Links</title>
</head>
<body>
 <a href="http://www.google.com">Ir a Google</a>
 <br />
 <a href="http://www.google.com"><img
 src="http://www.google.com.mx/intl/es/images/logo_sm.gif" /></a>
 <br />
 <a href="listas.html">Sobre listas</a>
</body>
</html>
```

Salida

3.1.3 Tablas: `<table>`

Las tablas facilitan la presentación de los datos en renglones y columnas. Una tabla, `<table>`, está dividida por renglones, `<tr>`, y cada renglón esta dividido en celdas `<td>`.
Durante los primeros días del diseño web, se solían usar tablas anidadas dentro de otras tablas para presentar una página web atractiva e interesante visualmente. Sin embargo, darle mantenimiento a páginas web diseñadas de ésta forma era una tarea muy dura y complicada. Los programadores y los diseñadores unieron esfuerzos para buscar una forma visualmente agradable de presentar la información sin utilizar miles de tablas en una página web, introdujeron la etiqueta `<div>`.

La sintaxis básica para presentar la información en una tabla es:

```
<table>
  <tr>
    <td>Dato</td>
    ...
    <td>Dato</td>
  </tr>
  ...
  <tr>
    <td>Dato</td>
    ...
    <td>Dato</td>
  </tr>
</table>
```

En otras palabras: se define la tabla, se divide en renglones y se agregan las celdas. Si se quiere mantener una consistencia en la presentación de la tabla cada renglón debe contener el mismo número de celdas.

Los elementos básicos de una tabla de html son:

Etiqueta	Uso
`<table>`	Crea una tabla
`<tr>`	Crea un renglón dentro de la

Etiqueta	Uso
	tabla
<td>	Crea una columna dentro de la tabla
<th>	Crea renglón de encabezado
<caption>	Crea un título para la tabla

Las etiquetas de las tablas contiene un número muy grande de atributos, algunos de ellos ya se han desaprobado por el W3C en favor a las hojas de estilo. A continuación mostramos los principales atributos de estas etiquetas:

Etiqueta	Atributo	Descripción
<table>	border	Especifica el tamaño del borde de la tabla
	cellpadding	Especifica el espacio entre la celda y su alrededor
	cellspacing	Especifica el espacio entre cada celda
	summary	Encabezado de la tabla

<tr>	aling	Define la alineación del texto dentro de la celda
	valing	Define la alineación vertical del texto dentro de la celda

<td>	align	Especifica el espacio horizontal que ocupara la celda
	colspan	Indica cuantas columnas se extenderá una celda
	rowspan	Indica cuantos renglones se extenderá una celda
	valign	Indica la alineación vertical del texto dentro de la celda.

<th>	align	Especifica el espacio horizontal que ocupara la celda
	colspan	Indica cuantas columnas se extenderá una celda
	rowspan	Indica cuantos renglones se extenderá una celda
	valign	Indica la alineación vertical del texto dentro de la celda.

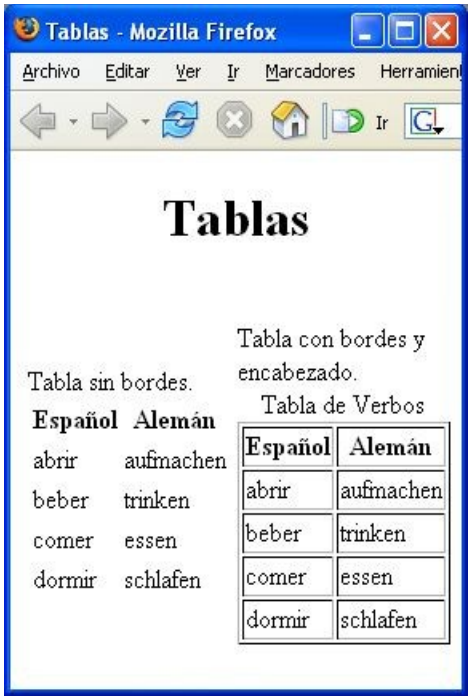
Sin duda alguna el uso de tablas en el diseño de páginas web sigue siendo una práctica común y el uso de hojas de estilo en cascada dota de más presentación a las tablas. Para finalizar con el tema de las tablas, el siguiente ejemplo muestra el uso de las tablas y al mismo tiempo como se pueden anidar las tablas.

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

```
<body>
      <center><h1>Tablas</h1></center>
      <br />

      <!-- Tabla principal -->
      <table>
      <tr>
            <td>
               <!-- Tabla sin bordes -->
            Tabla sin bordes.
            <table>
                  <tr>
                  <th>Espa&ntilde;ol</th>
                  <th>Alem&aacute;n</th>
                  </tr>

                  <tr>
                  <td>abrir</td>
                  <td>aufmachen</td>
                  </tr>

                  <tr>
                  <td>beber</td>
                  <td>trinken</td>
               </tr>
                <tr>
                  <td>comer</td>
                  <td>essen</td>
                  </tr>

                  <tr>
                  <td>dormir</td>
                  <td>schlafen</td>
                  </tr>
            </table>
            <!-- Fin de Tabla sin bordes -->
               </td>

            <td>
            <!-- Tabla con bordes -->
            Tabla con bordes y encabezado.
            <table border="1">
                  <caption>Tabla de Verbos</caption>
                  <tr>
                  <th>Espa&ntilde;ol</th>
                  <th>Alem&aacute;n</th>
                  </tr>

                  <tr>
                  <td>abrir</td>
                  <td>aufmachen</td>
                  </tr>

                  <tr>
```

```
                    <td>beber</td>
                    <td>trinken</td>
                    </tr>

                    <tr>
                    <td>comer</td>
                    <td>essen</td>
                    </tr>

                    <tr>
                    <td>dormir</td>
                    <td>schlafen</td>
                    </tr>
                </table>
                <!-- Fin de Tabla con bordes -->
                </td>
            </tr>
            </table>
            <!-- Fin de Tabla principal -->
    </body>
</html>
```

El resultado que arroja el navegador:



3.2 Formularios: `<form>`

Los formularios son componentes que permiten una interacción entre el usuario y la página web, esto permite capturar los datos del lado del cliente y procesarlos del lado del servidor.

Los formularios suelen constar de cajas de texto, cajas de selección, botones de elección, cuadros de selección, botones, etc.. Comúnmente son programas CGI[6](Interfaz común de pasarela) los encargados de procesar la información después de que el usuario envía el formulario de vuelta al servidor. Sin embargo, cualquier lenguaje de programación con capacidades de correr en un servidor puede realizar el procesamiento de los formularios, tal es el caso de Java o PHP.

A través de los formularios y el procesamiento de éstos, se puede genera contenido dinámico para las páginas web, es decir, páginas web dinámicas.

Los elementos que conforman a un formulario se mantiene entre las etiquetas `<from> </form>`.

3.2.1 Cajas de texto: `<input type="text">`

Las cajas de texto permiten insertar pequeñas cadenas de información, tales como nombres, teléfonos, etc..
Se crean mediante la etiqueta `<input type="text" />`
Los atributos que puede contener se muestran a continuación:

Atributo	Descripción
`id`	Identifica al componente dentro del formulario. Obligatorio en XHTML.
`maxlength`	Máximo número de caracteres que se pueden ingresar.
`name`	Identifica al componente dentro del formulario.
`size`	Tamaño con el que se presenta.
`value`	Valor inicial.

Éste es un sencillo ejemplo:

cajasdetexto.html

```
<!DOCTYPE ...>
<html>
<head><title>Cajas de texto</title></head>
<body>
<form>
Nombre: <input type="text" id="nombre" name="nombre" />
<br />
Apellido Paterno: <input type="text" id="apPat" name="apPat" />
<br />
```

6 En inglés, Common Gateway Interface

```
  Apellido Materno: <input type="text" id="apMat" name="apMat" />
  </form>
  </body>
</html>
```

Salida

3.2.2 Cajas de contraseña: `<input type="password">`

Las cajas de contraseña tiene los mismos atributos que las cajas de texto, pero al escribir sobre ellas, mantienen ocultos los caracteres escritos y muestran otros caracteres (asteriscos). Se declaran: `<input type="password" />`

cajasdecontraseña.html

```
<!DOCTYPE ...>
<html>
<head><title>Cajas de contrase&ntilde;a</title></head>
<body>
<form>
Password: <input type="password" id="pass" name="pass" />
<br />
Confirmar Password: <input type="password" id="rpass"
name="rpass" />
</form>
</body>
</html>
```

Salida

3.2.3 Cajas de selección: `<select>`

Las cajas de selección o listas de selección, mejor conocidas en inglés como *drop down box*, son una lista de opciones que se despliegan al hacer click sobre dicho componente.

Una caja de selección se declara con la etiqueta `<select>` y cada elemento en la lista de selección se declara con la etiqueta `<option>`. El texto insertado entre la etiqueta `<option>` y su cierre `</option>` es el texto que se muestra, pero el valor del atributo `value` es el que se envía. Cuando no se incluye el atributo `value` dentro de la etiqueta `<option>`, se envía el texto encerrado en la etiqueta `<option>`, pero esto es una mala práctica.

En las cajas de selección se puede seleccionar una o varias opciones, y también bloquear algunas opciones, eso dependerá de los atributos en las etiquetas.

Atributo	Descripción
`disabled="disabled"`	Si es colocado como atributo de la caja de selección deshabilita todas la lista de opciones disponibles, si es colocado como atributo de una opción sólo deshabilita dicha opción.
`multiple="multiple"`	Permite seleccionar más de una opción. Se debe mantener presionada la tecla `<CTRL>` o `<SHIFT>`.
`name`	Identifica al componente dentro del formulario. Se recomienda utilizar también el atributo `id`.
`selected="selected"`	Se coloca como atributo de la etiqueta `<option>` para que por default una opción esté seleccionada.
`size`	Se coloca como atributo de la etiqueta `<select>` cuando se desea mostrar determinado número de opciones al mismo tiempo.
`value`	Valor de la opción

A continuación se presenta un ejemplo que muestra el uso de las cajas de selección.

cajasdeseleccion.html

```
<!DOCTYPE ...>
<html>
      <head><title>Cajas de Selecci&oacute;n</title>
</head>
<body>
```

```
        <center><h1>Cajas de Selecci&oacute;n</h1></center>
        <br />
        Selecciona un color:
        <select id="farben01" name="farben01">
              <option value="azul">blau</option>
              <option value="cafe">braun</option>
              <option value="amarillo">gelb</option>
              <option value="verde">gr&uuml;n</option>
              <option value="rojo">rot</option>
              <option value="negro">schwarz</option>
              <option value="blanco">weiss</option>
        </select>
        <br />
        Selecciona varios colores:
        <select id="farben02" name="farben02" multiple="multiple" size="4">
              <option value="azul">blau</option>
              <option value="cafe">braun</option>
              <option value="amarillo">gelb</option>
              <option value="verde">gr&uuml;n</option>
              <option value="rojo">rot</option>
              <option value="negro">schwarz</option>
              <option value="blanco">weiss</option>
        </select>
        <br />
        Selecciona un color:
        <select id="farben03" name="farben03" size="4">
              <option value="azul">blau</option>
           <option value="cafe">braun</option>
              <option value="amarillo" disabled="disabled">gelb</option>
              <option value="verde" disabled="disabled">gr&uuml;n</option>
              <option value="rojo" disabled="disabled">rot</option>
              <option value="negro" selected="selected">schwarz</option>
              <option value="blanco">weiss</option>
        </select>
  </body>
</html>
```

Salida

3.2.4 Botones de elección: `<input type="radio">`

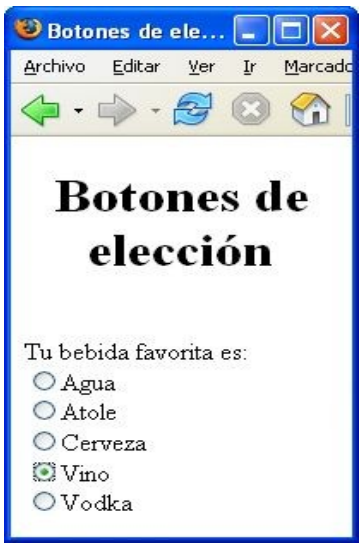
En inglés se les conoce como *radio buttons*, se trata de pequeños botones que se agrupan para formar una lista de opciones, y sólo se permite seleccionar una de las opciones.
Se definen con la etiqueta `<input type="radio">` y se agrupan colocando el mismo valor en el atributo `name` o `id`. Si se desea que por default una opción esté seleccionada, se debe colocar el atributo `checked="checked"`.

A continuación se muestra un ejemplo.

botonesdeeleccion.html

```
<!DOCTYPE ...>
<html>
<head>
      <title>Botones de elecci&oacute;n</title>
</head>
<body>
      <center><h1>Botones de elecci&oacute;n</h1></center>
      <br />
      Tu bebida favorita es:
      <br />
      <input type="radio" id="bebida" name="bebida" value="agua" />Agua
      <br />
      <input type="radio" id="bebida" name="bebida" value="atole" />Atole
      <br />
      <input type="radio" id="bebida" name="bebida" value="cerveza"
checked="checked"/>Cerveza
      <br />
      <input type="radio" id="bebida" name="bebida" value="vino" />Vino
      <br />
      <input type="radio" id="bebida" name="bebida" value="vodka" />Vodka
</body>
</html>
```

Salida



3.2.5 Cuadros de verificación: `<input type="checkbox">`

Conocidos en inglés como *checkboxes*. Son muy similares a los botones de elección, sin embrago, se pueden seleccionar más de una opción y no se necesita de agruparlos.

Se definen con la etiqueta <input type="checkbox">, si se coloca el atributo `checked="checked"` por default aparecerá seleccionada la opción.

A continuación se muestra un ejemplo de éste tipo de componente.

cuadrosdeverificacion.html

```
<!DOCTYPE ...>
<html>
<head>
	<title>Cuadros de verificaci&oacute;n</title>
</head>
<body>
	<center><h1>Cuadros de verificaci&oacute;n</h1></center>
	<br />
	Idiomas que dominas:
	<br />
	<input type="checkbox" id="aleman" name="aleman" value="aleman" checked="checked"
/>Alem&aacute;n
	<br />
	<input type="checkbox" id="espanol" name="espanol" value="espanol" />Espa&ntilde;ol
	<br />
	<input type="checkbox" id="frances" name="frances" value="frances" />Franc&eacute;s
	<br />
	<input type="checkbox" id="ingles" name="ingles" value="ingles" />Ingl&eacute;s
	<br />
	<input type="checkbox" id="italiano" name="italiano" value="italiano" />Italiano
	<br />
	<input type="checkbox" id="japones" name="japones" value="japones" />Japon&eacute;s
</body>
</html>
```

Salida

3.2.6 Botones de envió y borrado: `<input type="submit">` y `<input type="reset">`

El componente más utilizado para enviar un formulario es el botón de envió o *submit*; no es la única forma de enviar un formulario de vuelta al servidor, también se puede realizar ésta operación mediante un enlace o hipervínculo, pero ésta forma no es la más habitual.

Por lo general, un formulario cuenta con un botón de envió y uno de borrado o *reset*. El botón de *reset* limpia cada campo o componente que conforma el formulario, si existe un valor por default definido en el componente se coloca nuevamente dicho valor.

Un botón de envió se define: `<input type=”submit” value=”etiqueta_botón”>`.
Mientras que un botón de *reset* se define: `<input type=”reset” value=”etiqueta_botón”>`.

Finalmente mostraremos un ejemplo de éstos botones.

Salida

botones.html

```
<!DOCTYPE...>
<html>
<head>
      <title>Botones</title>
</head>
<body>
 <center><h1>Botones</h1></center>
  <form>
  Nombre de usuario: <input
  type="text" id="user"
  name="user" />
  <br />
  Password: <input type="password"
  id="pass" name="pass" />
  <br />
  <input type="submit"
  value="Login" />
  <input type="reset"
  value="Reset" />
  </form>
</body>
</html>
```

3.2.7 Métodos para enviar formularios

Después de que un usuario hace click en el botón *submit*, la información es devuelta al servidor web y procesada por el programa indicado en el atributo `action` de la etiqueta `<form>`. La forma en la que se envía la información al servidor dependerá del valor proporcionado en parámetro `method` de ésta misma etiqueta.

Cuando el atributo `method` tiene el valor de `get`, la información proporcionada por el usuario es enviada al servidor a través de la URL en texto plano y separada por, sin embargo cuando el valor del atributo `method` es `post` la información se adjunta a la cabecera de la petición y puede ser cifrada.

Es fácil identificar cuando un formulario fue enviado mediante el método `get`, pues se adjunta al URL, como ya se menciono, una cadena que comienza con el carácter `?` seguido del nombre de cada elemento, el signo de `=` y su respectivo valor, es decir:

```
http://www.miservidor.com/archivo.html?campo1=valor1&campo2=valor2&...
```

Se recomienda utilizar el método `post` cuando la información proporcionada por el cliente es confidencial.

Más adelante veremos como procesar los datos de un formulario con PHP.

3.3 El lenguaje de programación PHP

Los inicios de PHP se remontan al año de 1994 cuando el desarrollador Rasmus Lerdorf escribió en C y perl una serie de herramientas para mostrar su curriculum vitae y llevar estadísticas de cuanta gente lo visitaba en la red. En aquel entonces él le denomino a estas herramientas *Personal Home Page Tools*.

Muchas personas se interesaron por estas herramientas y comenzaron a pedirle el código, Rasmus Lerdorf accedió y formo una lista de correo para intercambiar idea y sugerencias. Finalmente la Universidad de Toronto le ofreció un empleo, donde tuvo que desarrollo un sistema de administración por web. Cuando maduro sus herramientas decidió lanzarlas como un producto de software libre.

Dos desarrolladores, Zeev Suraski y Andi Gutmans, reescribieron en 1997 parte del código de PHP (en el especial el analizador sintáctico o *parser*) y fue lanzado PHP-3 en 1998. Las siglas también cambiaron, se le denomino *Hypertext PreProcessor*. Así, el desarrollo de PHP continuo en manos de éstos dos programadores y en el año 2000 fue lanzado PHP-4 con el motor Zend 1.0. Para el año de 2004 se lanzo PHP 5 con el motor Zend 2, ésta ultima versión introduce el paradigma de la programación orientada a objetos a PHP.

PHP es hoy en día el lenguaje interpretado del lado del servidor más utilizado en el desarrollo de aplicaciones web, cuenta con un gran número de colaboradores a nivel mundial, está portado a las plataformas más utilizadas (Windows, GNU/Linux, Mac OSX, etc.), es un proyecto de código abierto y ofrece mucha documentación[7] en su sitio web[8].

3.3.1 ¿Cómo funciona?

Los scripts desarrollados en PHP se embeben dentro de una página HTML, es común que a los archivos se les coloque la extensión `.php` en lugar de `.html`.

Cuando un cliente hace la petición a un servidor web de un archivo con extensión `.php`, el servidor pasa el control al motor Zend para que interprete el contenido del archivo, el intérprete devuelve el resultado al servidor web y esté finalmente entrega al cliente el resultado en código HTML.

El motor Zend interpretará todo aquel código PHP embebido en el HTML que se encuentre entre la etiquetas:

- `<?php ... ?>`
- `<? ... ?>`
- `<script language="php"> ... </script>`
- `<% ... %>`

7 http://www.php.net/tut.php
8 http://www.php.net

Por comodidad, en nuestros ejemplos utilizaremos el etiquetado <?php ?>.
Las características avanzadas y el mejor desempeño de PHP se alcanzan en la plataforma GNU/Linux, los ejemplos que aquí mostraremos estarán ejecutándose en el servidor web Apache[9] sobre Windows[10] para desmotar la potabilidad del lenguaje, sin embrago, cuando sea necesario ejecutaremos sobre Slackware los ejemplos.

3.3.2 Separación de instrucciones, impresión y comentarios

PHP tiene una sintaxis similar a la del lenguaje C o Java. Cada instrucción se separa del resto con el ;. Otra forma es empleando la etiqueta de fin de bloque (?>) que implica el fin de la declaración.

La impresión de un resultado se realiza mediante las sentencias echo() o print(), como se trata de sentencias propias del lenguaje es opcional el uso de los paréntesis. Estas sentencia pueden interpolar variables entre cadenas, es decir se puede imprimir una cadena y el valor de una variable embebida en la cadena. También aceptan el etiquetado de un bloque para un texto muy largo mediante la sintaxis heredoc (<<<).

```
print "Hola a todos";
print "Imprime el valor de una variable: $variable";
echo 'Hola a todos';
echo <<<HOLA
Éste es un texto muy largo,
los saltos de línea también se imprimen
así como las tabulaciones.
HOLA;
```

Los comentarios en PHP son idénticos a los de C, C++ y UNIX. Los comentarios de varias líneas inician con /* y terminan en */, los comentarios de una sola línea pueden ser escritos después de // o bien #.
Es importante no anidar comentarios de varias líneas dentro de comentarios de varias líneas:

```
/* Éste comentario es incorrecto
  /* comentario anidado */
*/

/* Éste comentario no causa conflictos
   //comentario de una línea
*/
```

3.3.3 Tipos de datos y variables

PHP no se considera un lenguaje fuertemente tipificado, ya que no se debe indicar el tipo de dato de sus variables, esté es determinado por el contexto en el que la variable es usada en tiempo de interpretación.

Existen 8 tipos de datos primitivos en PHP: boolean, integer, float, string, array, object, resource y NULL.

Una variable en PHP se indica anteponiendo al nombre de la variable el signo de $. El nombre de las variables debe comenzar con una letra o un guion bajo (_), y no debe comenzar con un número o contener los siguientes caracteres: @, ., +, -, !.

9 http://httpd.apache.org/
10 http://www.easyphp.org/

Los nombres de las variables son sensibles a mayúsculas y minúsculas, de modo que las variable `$precio`, `$Precio` y `$PRECIO` son diferentes.

Una variable booleana o **boolean**, puede contener los valores TRUE o FALSE, no importa si se escribe en mayúsculas o minúsculas:

```
$isValid = true;
$isValid = True;
$isValid = FALSE;
```

Las variables enteras o **integer**, se pueden especificar en notación decimal (base 10), octal (base 8) y hexadecimal (base 16), y pueden tener signo - o +.
Cuando se emplea la notación octal, debe preceder el número con un 0 (cero), para usar la notación hexadecimal, preceda el número con 0x:

```
$a = 31; //decimal positivo
$a = -31; //decimal negativo
$a = 031; //octal (25 en decimal)
$a = 0x31; //hexadecimal (49 en decimal)
```

Los números reales o **float**, se representan con el punto decimal o bien en notación científica:

```
$a = 1.31;
$a = 1e31;
$a = 1E-31;
```

El tipo de dato cadena, **string**, es una secuencia de caracteres que se encierra entre comillas simples (`''`) o dobles(`""`). Existen muchas funciones para manipular cadenas que veremos más adelante.

```
$str = "Hola a todos";
$str = 'Hola a todos';
```

Los arreglos o matrices, **array**, son un tipo de dato más complejo que puede funcionan como vector, lista, cola, pila, etc.. Un arreglo se crea mediante la expresión `array()`, cada elemento del arreglo está ligado a una clave o indice, dicha clave puede ser un valor entero o una cadena. Más adelante dedicaremos un apartado para el estudio de éste tipo de dato.

```
$menu = array(1=>"Sopa", 2 => "Carne", "precio" = 23.99);
$menu = array(1,2,3,4,5,"números");
```

PHP es tan flexible que permite programar estructuradamente, orientado a objetos o una mezcla de ambas metodologías. El tipo de dato **object** representa un tipo de una clase determinada.

Cuando queremos que una variable carezca de valor debemos asignarle el valor de **NULL**.

3.3.3.1 Cadenas

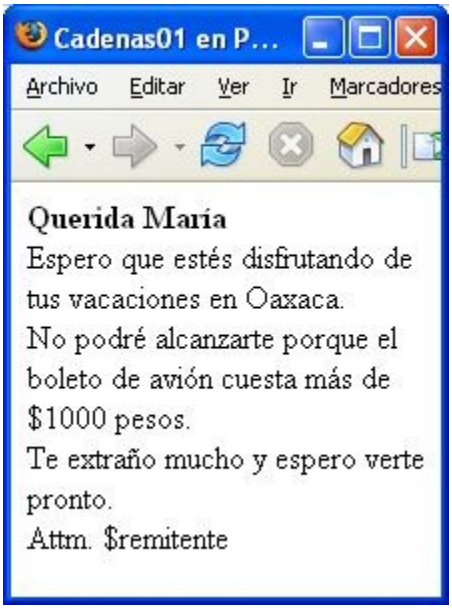
Cuando se declara o imprime una cadena rodeada por comillas simples (''), la cadena se imprime tal cual se escribió y no se realiza una interpolación de variables. En el caso de una cadena rodeada por comillas dobles la interpolación de variables si se lleva a cabo.

Existen ciertos caracteres que deben ser "escapados", de otra forma crearían un error de sintaxis durante la interpretación del script:

Carácter de escape	Significado
\n	Salto de línea
\r	Retorno de carro
\t	Tabulación
\\	El Carácter \
\$	El Carácter $
\"	El Carácter "
\'	El Carácter '

A continuación se muestra un ejemplo de caracteres "escapados", interpolación de variables e impresión.

Salida



cadenas01.php

```
<!DOCTYPE ...>
<html>
<head>
      <title>Cadenas01 en PHP</title>
</head>
<body>
<?php
      $persona = "Mar&iacute;a";
      $remitente = 'Jos&eacute;';
      $pretexto = "No podr&eacute; alcanzarte
porque el boleto de avi&oacute;n cuesta m&aacute;s
de \$1000 pesos.";

      print "<b>Querida $persona</b>";
      echo <<<SALUDO
      <br />Espero que est&eacute;s disfrutando de
tus vacaciones en Oaxaca.
      <br />$pretexto
      <br />Te extra&ntilde;o mucho y espero verte
pronto.
SALUDO;
      echo '<br />Attm. $remitente';
?>
</body>
</html>
```

En PHP la **concatenación** se realiza con el operador punto (.).

```
$pais = "México";
print "Yo vivo en ".$pais;
print "Un saludo desde " . "México";
```

Existe una diversidad de funciones para manipular y validar cadenas, a continuación mostraremos sólo unas cuantas funciones para éste propósito, pero se recomienda visitar la documentación[11] de PHP referente a cadenas.

Función	Significado	Ejemplo
`chop()`	Devuelve una cadena sin los espacios sobrantes y los saltos de línea al final de la cadena.	`$linea = "Hola  \n";` `$sinSaltos = chop($linea);`
`explode()`	Devuelve un arreglo de cadenas, cada elemento es una subcadena que fue extraído de una cadena a partir de un separador.	`$cad = "abc/def/ghi/jkl";` `$arr = explode("/",$cad);`
`htmlentities()`	Convierte todos los caracteres aplicables a entidades HTML.	`$lineaEnt =` `htmlentities($linea);`
`htmlspecialchars()`	Convierte sólo los caracteres especiales de marcado de HTML a entidades.	`$lineaEnt =` `htmlspecialchars($linea);`

`implode()`	Devuelve una cadena formada por la unión (a partir de un separado), de todos los elementos de un arreglo.	`$animal = array("gato", "perro",`
		`"conejo", "foca");`
		`$cad = implode(":",$animal);`

`join()`	Es un alias de `implode()`.	`$frutas = array("plátano",`
		`"sandía", "melón", "uva");`
		`$cad = join("/",$frutas);`

`ltrim()`	Devuelve una cadena sin espacios en blanco, tabulaciones o saltos de línea al inicio.	`$linea = "  hola";` `$cad = ltrim($linea);`
`rtrim()`	Devuelve una cadena sin espacios en blanco, tabulaciones o saltos de línea al final.	`$linea = "hola \n";` `$cad = rtrim($linea);`

11 http://www.php.net/manual/es/ref.strings.php

Función	Significado	Ejemplo

`strcmp()`	Compara dos cadenas, devuelve < 0 si la primer cadena es menor que la segunda, > 0 si es lo contrario y 0 si son iguales.	`$cad1 = "hola amigo";`
		`$cad2 = "hola amigo";`
		`$num = strcmp($cad1, $cad2);`

`strlen()`	Devuelve la longitud de una cadena.	`$cad = "Esta cadena es muy larga";` `$lonCad = strlen($cad);`
`strrev()`	Devuelve una cadena invertida.	`$linea = "Hola y adiós";` `$invLinea = strrev($linea);`
`strtolower()`	Devuelve una cadena con todos los caracteres en minúsculas.	`$cad = "HOLA A TODOS";` `$lowCad = strtolower($cad);`
`strtoupper()`	Devuelve una cadena con todos los caracteres en mayúsculas.	`$cad = "hola a todos";` `$upCad = strtoupper($cad);`
`trim()`	Devuelve una cadena sin espacios en blanco, saltos de línea o tabuladores al inicio y al final.	`$cad1 = " \nHola\n ";` `$cad2 = trim($cad1);`

3.3.3.2 Números

Ya se mencionó la forma de representar en PHP números en base 10, 8 y 16; es necesario precisar que el tamaño de un número de punto flotante (*float*) dependerá de la plataforma donde se esté ejecutando PHP, por lo general el número más grande que se puede representar es el 1.8e308, algo así como 14 dígitos decimales.

Es común que se pierda exactitud al convertir números de punto flotante a su representación binaria, por ejemplo 1/3 en forma decimal, 0.333333..., es una secuencia infinita que no puede ser representada con exactitud.
Una expresión como `floor((0.1+0.7)*10)` devolverá 7 en lugar del esperado 8 ya que el resultado de la representación interna es en realidad algo como 7.9999999999....

Por las razones expuestas, se recomienda no comparar directamente números flotantes para verificar si son iguales y si se requiere una mejor precisión es mejor emplear las Funciones matemáticas de precisión arbitraria BCMath[12].

Las cadenas pueden ser tratadas como números:

- El valor es dado por el inicio de la cadena,
- Si la cadena contiene cualquier carácter entre '.', 'e', o 'E' se evalúa como *float*, de lo contrario se evalúa como entero,
- Si la cadena comienza con caracteres numéricos válidos, éstos serán el valor usado. De lo contrario, el valor será 0 (cero).

12 http://www.php.net/manual/es/ref.bc.php

Tomando el ejemplo de cadenas de la documentación oficial de PHP, tenemos:

```
<?php
$foo = 1 + "10.5";                // $foo es flotante (11.5)
$foo = 1 + "-1.3e3";              // $foo es flotante (-1299)
$foo = 1 + "bob-1.3e3";           // $foo es entero (1)
$foo = 1 + "bob3";                // $foo es entero (1)
$foo = 1 + "10 Cerditos";         // $foo es entero (11)
$foo = 4 + "10.2 Cerditos";       // $foo es flotante (14.2)
$foo = "10.0 cerdos " + 1;        // $foo es flotante (11)
$foo = "10.0 cerdos " + 1.0;      // $foo es flotante (11)
?>
```

Al igual que las cadenas, existe una extensa lista de funciones y constantes matemáticas disponibles en PHP, en el sitio http://www.php.net/manual/es/ref.math.php se encuentran los detalles de cada una de ellas.

3.3.3.3 Arreglos

Un *array* es un contenedor que contiene múltiples valores y los asocia con una clave y un valor. Un elemento del *array* solamente puede tener una clave que deberá ser distinta del resto, ésta clave puede ser un entero, una cadena o un número flotante.

Además de la forma para ya se mostró para declarar que una variable es de tipo *array*, existe la siguiente forma:

```
$os[0] = "GNU/Linux";
$os[1] = "OS X";
$os[2] = 'FreeBSD';
...
$os[n] = 'MS-DOS';
```

```
$os["linux"] = "GNU/Linux";
$os['mac'] = "OS X";
$os['bsd'] = 'FreeBSD ';
...
$os['ms'] = 'MS-DOS';
```

Éste ejemplo ilustra dos cosas: las claves de cada elemento del *array* se indican en los corchetes y cuando se trata de calves numéricas no es necesario rodearlas con comillas.
Cuando no se especifica la clave para cada elemento del *array* se asocia por *default* un número entero, comenzando desde el 0.

```
<?php
$europa = array('Alemania', 'Bélgica', 'Croacia', 'Dinamarca', 'España');
print europa[0]; //imprime Alemania
print europa[3]; //imprime Dinamarca
$america[] = 'Argentina'; //clave 0
$america[] = 'Bolivia'; //clave 1
$america[] = 'Colombia'; //clave 2
print america[0]; //imprime Argentina
print america[2]; //imprime Colombia
?>
```

A continuación mostraremos las funciones más utilizadas para manipular arreglos, pero se sugiere nuevamente visitar la documentación oficial[13] para más detalles.

Función	Significado	Ejemplo
`array()`	Crea un arreglo.	`$frutas = array('sandía', 'melón', 'mango');`
`arsort()`	Ordena un arreglo en orden inverso y mantiene la asociación de las claves.	`$frutas = array('b'=>'sandía', 'a'=>'melón','c'=>'mango'); arsort($frutas);`
`asort();`	Ordena un arreglo y mantiene la asociación de las claves.	`$frutas = array('b'=>'sandía', 'a'=>'melón','c'=>'mango'); asort($frutas);`
`count()`	Devuelve el número de elementos en un array.	`$frutas = array('b'=>'sandía', 'a'=>'melón','c'=>'mango'); $size = count($frutas);`
`current()`	Devuelve el elemento al que apunta el puntero interno del arreglo. No se modifica el puntero.	`$frutas = array('sandía', 'melón', 'mango'); print current($frutas); //imprime sandía`
`end()`	Mueve el puntero interno del arreglo hasta el último elemento.	`$frutas = array('sandía', 'melón', 'mango'); print end($frutas); //imprime mango`
`key()`	Devuelve la clave del elemento al que apunta el puntero interno.	`$frutas = array('b'=>'sandía', 'a'=>'melón','c'=>'mango'); print key($frutas); //imprime b`
`next()`	Avanza una posición el puntero interno del arreglo.	`$frutas = array('sandía', 'melón', 'mango'); print next($frutas); //imprime melón`
`reset()`	Posiciona el puntero interno en el primer elemento del arreglo.	`$frutas = array('sandía', 'melón', 'mango'); next($frutas); next($frutas); print reset($frutas); //imprime sandía`

3.3.4 Operadores

Tenemos tres grupos de operadores: unarios, binarios y ternarios.
Los operadores unarios operan sobre un sólo valor, por ejemplo ++ (el operador de incremento); los binarios operan sobre dos valores y los ternarios, como `:?`, operan sobre tres valores.

De acuerdo a la documentación oficial la precedencia de los operadores[14] es la siguiente:

13 http://www.php.net/manual/es/ref.array.php
14 http://www.php.net/manual/es/language.operators.php

Asociatividad	Operadores	información Adicional
No asociativo	`new`	Crear una instancia de un objeto.
Izquierda	`[`	array()
No asociativo	`++ --`	Incremento/decremento
No asociativo	`! ~ - (int) (float) (string) (array) (object) @`	Tipos
Izquierda	`* / %`	Aritmética
Izquierda	`+ - .`	Aritmética y cadena
Izquierda	`<< >>`	Manejo de bits
No asociativo	`< <= > >=`	Comparación
No asociativo	`== != === !==`	Comparación
Izquierda	`&`	Manejo de bits y referencia
Izquierda	`^`	Manejo de bits
Izquierda	`|`	Manejo de bits
Izquierda	`&&`	Lógicos
Izquierda	`||`	Lógicos
Izquierda	`?:`	Ternario
Derecha	`= += -= *= /= .= %= &= |= ^= <<= >>=`	Asignación
Izquierda	`and`	Lógicos
Izquierda	`xor`	Lógicos
Izquierda	`or`	Lógicos
Izquierda	`,`	Varios usos

La asociatividad de izquierda quiere decir que la expresión es evaluada desde la izquierda a la derecha, la asociatividad de derecha quiere decir lo contrario.

3.3.5 Estructuras de control

Las estructuras de control permiten modificar el curso de ejecución de las instrucciones de un programa, tomar decisiones cundo ocurran determinados eventos y repetir instrucciones un determinado número de veces.

El Teorema de Dijkstra, demostrado por el físico Edsger Dijkstra en los años sesenta, demuestra que todo programa puede escribirse utilizando únicamente tres instrucciones de control:

- Secuencial de instrucciones.
- Instrucción condicional.
- Iteración, o bucle de instrucciones.

Como a continuación veremos la mayor parte de la estructuras de control que maneja PHP y tienen una sintaxis similar a las del lenguaje de programación C.

3.3.5.1 `if-else`

La estructura de control `if()` permite que ciertas acciones se ejecuten si se cumple una condición booleana `true`.
La sintaxis para ésta estructura de control es la siguiente:

```
if(expresión)
{
  sentencias
}
```

```
$a = 5;
$b = 3;
if($a > $b)
{
   print "$a es mayor que $b";
}
```

```
$a = 5;
$b = 3;
if(!($a < $b))
{
   print "$a es mayor que $b";
}
```

Ésta sintaxis evalúa la expresión a su valor booleano y si es verdadera ejecuta el bloque de sentencias que encierra entre llaves.

Cuando necesitamos que se ejecute una sentencia si se cumple con una condición y otra distinta en caso contrario, se emplea la estructura `if-else`.
Las sentencias dentro del bloque `else` se ejecutan si y sólo si la expresión que se evalúa en `if` se evalúa en `FALSE`. La sintaxis es la siguiente:

```
if(expresión)
{
  sentencias
}
else
{
  sentencias
}
```

```
$a = 5;
$b = 3;
if($a < $b)
{
   print "$b es mayor que $a";
}
else
{
   print "$a es mayor que $b";
}
```

```
$a = 5;
$b = 3;
if(!($a > $b))
{
   print "$b es mayor que $a";
}
else
{
   print "$a es mayor que $b";
}
```

Cuando necesitamos probar múltiples condiciones por separado la estructura `if-else` se amplia a `if-elseif-else`. Una vez que una de las expresiones de prueba `if()` o `elseif()` es `true`, el resto se ignora. Ésta construcción es equivalente a la estructura `switch`.

```
if(expresión)
{
  sentencias
}
elseif(expresión)
{
  sentencias
}
elseif(expresión)
{
  sentencias
}
else
{
  sentencias
}
```

```
$a = 5;
$b = 3;
$c = 1;
if($a < $c)
{
   print "$c es mayor que $a";
}
elseif($a > $b)
{
   print "$a es mayor que $b y $c";
}
else
{
   print "$a es menor que $b y $c";
}
```

Cuando necesitamos probar múltiples condiciones de manera seriada se emplea una construcción `if()-if()-if()...if()`, así cada expresión que se evalúa en `false` pasa a ser evaluada en el siguiente `if()`.

3.3.5.2 `while`

Cuando una estructura de un programa hace algo repetidas veces se le denomina bucle. Se le denomina iteración a cada ejecución de las sentencias del bucle.
El bucle más simple la estructura `while`, es como un `if` reiterativo. A diferencia del `if`, evalúa una expresión después de haber ejecutando el bloque de sentencias que alberga.

```
while(expresión)
{
  sentencias
}
```

```
$i = 0;
while($i < 10)
{
   print $i++;
}
```

Para asegurar que al menos una vez se ejecutaran las sentencias del bucle se cuenta con la estructura `do-while`. Con ésta estructura la expresión de condición se evalúa al final de cada iteración.

```
do
{
  sentencias
}
while(expresión);
```

```
$i = 0;
do
{
  print $i++;
}
while($i < 10);
```

Cuando se desea terminar con las iteraciones de un bucle se tiene la expresión `break`, por otro lado, cuando se desea brincar a la siguiente iteración e ignorar el resto de las sentencia dentro de un bucle se tiene la expresión `continue`.

3.3.5.3 `for`

Está formado de tres expresiones, la primera se ejecuta sólo una vez al principio del bucle y se le denomina expresión de inicialización, la segunda expresión es la expresión que controla las iteraciones del bucle y se evalúa al inicio de cada iteración, finalmente la tercera expresión se evalúa al final de cada iteración.

```
for(exp1; exp2; exp3)
{
   sentencias
}
```

```
for($i=0; $i < 10; $i++)
{
  print $i;
}
```

Tanto con el bucle `for` como con el bucle `while` se puede recorre un arreglo:

```
$frutas = array('sandía', 'melón',
'mango');
$size = count($frutas);
for($i=0; $i < $size; $i++)
{
  print $frutas[$i];
}
```

```
$frutas = array('sandía', 'melón', 'mango');
$size = count($frutas);
$i = 0;
while($i < $size)
{
  print $frutas[$i++];
}
```

3.3.5.4 `foreach`

Éste bucle se introdujo en la versión PHP 4 y funciona únicamente con arreglos, devuelve un error si se utiliza con otro tipo de dato o con variables no inicializadas.
Existen dos sintaxis utilizadas para éste bucle:

```
foreach($array as $value)
{
  sentencias
}
```

```
foreach($array as $key => $value)
{
  sentencias
}
```

Ambas formas recorren el array, la primera asigna a la variable `$value` el valor del elemento actual y el puntero interno del array se incrementa en una unidad (en la siguiente iteración el puntero interno del arreglo apuntara al elemento siguiente). La segunda forma hace lo mismo, pero la clave del elemento actual será asignada a la variable `$key` y el valor a la variable `$value`.

3.3.6 Funciones

Las funciones son una forma de reutilizar código, pueden recibir o no parámetros y devolver o no algún resultado. Las funciones guardan en su interior una serie de sentencias que sólo ellas conocen y manipulan, de modo que las variables que son utilizadas dentro de las funciones sólo existen dentro de la función. Toda función debe ser declarada anteponiendo al nombre de la función la palabra reservada `function`.

Las funciones que no reciben parámetros tienen la sintaxis o firma:

```
function mi_funcion(){}
```

Las funciones que reciben parámetros tienen la sintaxis:

```
function mi_funcion($arg_1, $arg_2, ..., $arg_n){}
```

```
<?php
function print_header()
{
 print '<html><head><title>
         Mi página Web
       </title></head><body>';
}

function print_footer()
{
  print '<hr />Vuelva
         pronto</body></html>';
}

print_header();
print 'Hola a todos';
print_footer();
?>
```

```
<?php
function print_header($title)
{
   print "<html><head><title>$title
          </title></head><body>";
}

function print_footer($mess)
{
  print "<hr />$mess</body></html>";
}

print_header('Hola');
print 'hola a todos';
print_footer('Bye2');
?>
```

Cuando se invoca sin algún parámetro a una función que ha sido definida para recibir parámetros, PHP genera una advertencia: `Missing arguments....`

Para evitar éste tipo de mensajes se pueden crear funciones con parámetros opcionales a los que se les asigna un valor por *default*.

```
function mi_funcion($arg_1, $arg_2=valor, ..., $arg_n=valor){}
```

Así, si se facilita un argumento a la función es usado, sino se usa el de *defualt*.

```
<?php
function print_header($title = 'Mi página Web')
{
 print
"<html><head><title>$title</title></head><body>";
}

function print_footer($mess = 'Vuelva pronto')
{
  print "<hr />$mess</body></html>";
}

print_header('Bienvenidos');
print 'Hola a todos';
print_footer();
?>
```

El paso de parámetros a una función se realiza por valor, es decir, si se llega a modificar el valor del parámetro dentro de la función afuera de ella no se vera afectado. Si se desea que una función modifique el valor del parámetro se debe de pasar por referencia. Para que el parámetro de una función siempre sea pasado por referencia, se debe anteponer un & al nombre del parámetro en la firma de la función:

```
function my_funcion(&$arg_1, &$arg_2, ..., &$arg_n){}
```

```
<?php
function concatena_algo(&$mess)
{
  $mess .= " para ti";
}

$regalo = 'Un regalo';
concatena_algo($regalo);
print $regalo; //imprime Un regalo para ti
?>
```

Las funciones pueden realizar un determinado proceso con los argumentos que reciben y devolver un resultado. Para que una función retorne el resultado se utiliza la palabra reservada `return` seguida de la expresión que devolverá:

```
function mi_funcion()
{
  senetncias
  return expresión;
}
```

```
function mi_funcion($arg_1, ..., $arg_n)
{
  senetncias
  return expresión;
}
```

```
<?php
function suma($op1, $op2)
{
  return $op1 + $op2;
}

$suma = suma(40 + 20);
print "40 + 20 = $suma";
?>
```

3.3.7 Procesar formularios

Hemos hablamos de los métodos que un formulario emplea para enviar formularios devuelta al servidor, es momento para hablar de como un programa del lado del servidor procesa la información que el usuario proporcionó.

Los datos enviados mediante el método `GET`, se alojan en un arreglo global llamado `$_GET`, mientras que los datos enviados por `POST` se alojan en el arreglo `$_POST`. Existe el arreglo `$_REQUEST` que aloja tanto los datos enviados por `POST` como por `GET`.

Vamos a mostrar con un ejemplo de como procesar un sencillo formulario:

formulario01.php

```
<?php

if(array_key_exists('nombre',$_POST))
{
  print 'Hola, '.$_POST['nombre'];
}
else
{
  print<<<_HTML_
      <form method="POST"
      action="$_SERVER[PHP_SELF]">
      Tu nombre: <input type="text"
       id="nombre" name="nombre" />
      <br />
      <input type="submit"
       value="Enviar" />
      </form>
_HTML_;
}
?>
```

Salida

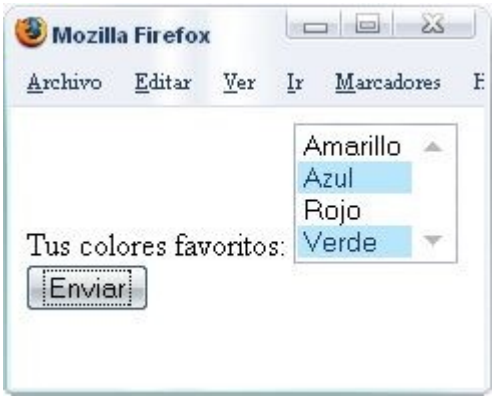
Lo primero que se verifica es que exista en el arreglo `$_POST` la clave `nombre`, si esto ocurre se imprime el valor al que está asociada esa clave dentro del arreglo. Sino está disponible dicha clave se imprime el formulario, que será enviado mediante el método `POST` al mismo script que lo genero (`$_SERVER[PHP_SELF]`).

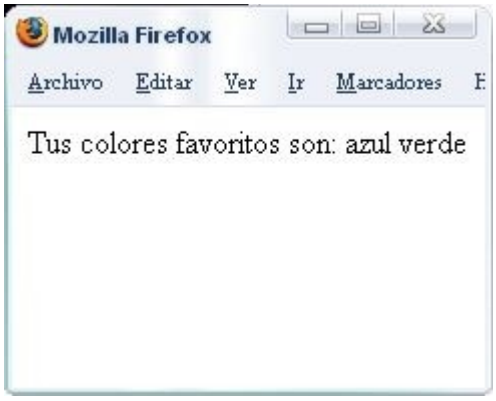
Cuando un elemento del formulario puede tener múltiples valores es necesario que su nombre y id terminen con `[]`, esto le indica al interprete que maneje múltiples valores para ese elemento.

form_colores.php

```
<?php
if(array_key_exists('colores',$_POST))
{
  print 'Tus colores favoritos son: ';
  foreach($_POST['colores'] as $color)
  {
      print $color.' ';
  }
}
else
{
  print<<<_HTML_
      <form method="POST"
     action="$_SERVER[PHP_SELF]">
     Tus colores favoritos:
<select id="colores[]" name="colores[]" multiple="multiple" />
<option value="amarillo">Amarillo</option>
<option value="azul">Azul</option>
<option value="rojo">Rojo</option>
<option value="verde">Verde</option>
</select>
      <br />
      <input type="submit"
      value="Enviar" />
      </form>
_HTML_;
}
?>
```

Salida



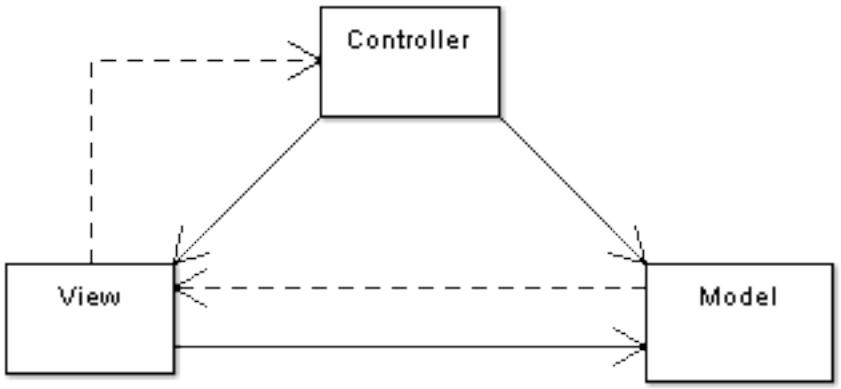


Es importante señalar que no es buena práctica de programación mezclar el procesamiento de la información con la presentación de la misma. La solución más utilizada hoy en día es emplear el patrón de diseño que se denomina MVC[15], Modelo Vista Controlador:

- Modelo. Es la capa de la lógica del negocio (funciones "con significado"), todo acceso a la base de datos.
- Vista. Es la interfaz del usuario, formatea los datos y los presenta.
- Controlador. Es el "director", que responde al usuario, interactua con el Modelo y la vista. Se limita a obtener valores, procesarlos y obtener otros valores.

En PHP existen diversos *frameworks* para trabajar bajo éste modelo, los más destacados Jaws, Symfony Framework y CakePHP.

Tomado de http://en.wikipedia.org/wiki/Model-view-controller tenemos la representación gráfica de éste patrón de diseño.



Resumen

HTML un día comenzó como un lenguaje para compartir documentos científicos a través de la Internet y es actualmente un estándar abierto del que el *e-commerce* se ha beneficiado ampliamante. Con el advenimiento de el XML el HTML se redefinió y llego a ser lo que hoy se conoce como XHTML, que no es más que el mismo lenguaje bajo las reglas del XML.

A lo largo de éste capítulo sólo estudiamos las etiquetas mas destacadas del HTML, pues existe un centenar de ellas que han ido desapareciendo o evolucionando, por lo que sería muy difícil abarcarlas en éste trabajo.

Por otra parte, PHP es el lenguaje mas utilizado para desarrollar aplicaciones web de manera sencilla y rápida. El termino LAMP (Linux, Apache, MySQL, PHP/Python/Perl) es lo que hoy miles de empresas han adoptado como solución a sus necesidades de *e-commerce*.

En el siguiente capítulo veremos lo sencillo que resulta trabajar con la base de datos MYSQL y PHP.

15 Model View Controller

Capítulo 4.

Interacción de PHP con MySQL.

- Pero, ¿qué dijo el Lirón? - pregunto uno de los miembros del jurado.
- No me acuerdo - dijo el sombrerero.
- Tienes que acordarte - comentó el Rey -; si no serás ejecutado.

Carrol, Lewis. Alicia en el país de las maravillas

4.1 SQL

SQL[1], Lenguaje de Consultas Estructurado, es desarrollado por IBM a mediados de la década de 1970 para la Manipulación de los datos de las bases de datos relacionales. Tenía como objetivo principal emplear el idioma inglés en la sintaxis del lenguaje para facilitar su uso y permitir una rápida comunicación entre personas y bases de datos. Durante un tiempo el lenguaje no cubrió las expectativas de IBM y fue Oracle quien lo introdujo como un producto comercial. Después de registrarse casos de éxito y el apoyo de un gran número de empresas, la ANSI[2] decide estandarizarlo en 1986.

SQL se descompone en subcomponentes para poder entenderlo mejor, a la primera parte se le denomina lenguaje de creación de datos (*Data Definition Language*, DDL) y a la segunda lenguaje de manipulación de datos (Data Manipulation Language, DML).

Como cualquier otro lenguaje, SQL tiene una sintaxis y una gramática. La primer palabra de una sentencia SQL es un verbo que le indica a la base de datos que es lo que se quiere hacer. Las palabras que siguen después el verbo son parámetros que se refieren a las columnas que se quieren manipular y condiciones que deben de cumplirse.

El DBMS[3] MySQL fue desarrollado por la empresa privada MySQL AB (antes TcX) a mediados de 1996, hoy en día cuenta con más de 6 millones de instalaciones en todo el mundo. Cumple con el estándar ANSI SQL 99 y actualmente soporta sentencias `SELECT` anidadas. Es sin duda el sistema de gestión de bases de datos más utilizado en aplicaciones Web y es multiplataforma. Se han desarrollado APIs para Perl, TCL, C/C++, Java y por supuesto PHP.

4.1.1 Lenguaje de Creación de datos (DDL)

La finalidad del DDL[4] es proporcionar un conjunto de instrucciones que se empleen sólo para definir la estructura donde se almacenaran los datos, esto es, la creación, eliminación y alteración de bases de datos, tablas, vistas o triggers.

Las sentencias definidas en DDL son: `CREATE`, `DROP` y `ALTER`.

La sentencia `CREATE` es empleada para crear nuevas bases de datos, tablas, etc., tiene la siguiente sintaxis:

1 Structured Query Language
2 American National Standards Institute
3 Data Base Manager System
4 Data Definition Language

Crear una Base de datos	Crear una Tabla
CREATE DATABASE nombre_base_datos	**CREATE TABLE** nombre_tabla (nombre_columna1 tipo_dato [atributos], nombre_columna2 tipo_dato [atributos], nombre_columnaN tipo_dato [atributos])

La sentencia DROP es empleada para eliminar bases de datos, tablas, etc., tiene la siguiente sintaxis:

Eliminar una Base de datos	Eliminar una tabla
DROP DATABASE nombre_base_datos	**DROP TABLE** nombre_tabla

La sentencia ALTER es utilizada cuando se necesita modificar el tipo de dato almacenado en una columna determinada, agregar una columna a una tabla, o bien, eliminar una columna. La sintaxis es:

Cambiar el tipo de dato de una columna	Agregar una columna
ALTER TABLE nombre_tabla **CHANAGE** nombre_columna nombre_columna tipo_dato	**ALTER TABLE** nombre_tabla **ADD** nombre_columna tipo_dato
Cambiar el nombre a una columna	**Eliminar una columna**
ALTER TABLE nombre_tabla **CHANAGE** nombre_columna nuevo_nombre_columna tipo_dato	**ALTER TABLE** nombre_tabla **DROP** nombre_columna
Cambiar el nombre a una tabla	
ALTER TABLE nombre_tabla **RENAME** nuevo_nombre_tabla	

4.1.2 Lenguaje de Manipulación de datos (DML)

DML proporciona un conjunto de sentencias para manipular y modificar los datos de una tabla.
Las acciones que se pueden realizar son la extracción de información (SELECT), inserción (INSERT INTO), actualización (UPDATE) y eliminación (DELETE).

Estas sentencias por lo general están acompañadas de **cláusulas**. Las cláusulas son condiciones de modificación utilizadas para definir los datos que se desean seleccionar o manipular. Tenemos las siguientes cláusulas:

Cláusula	Descripción
FROM	Especifica la(s) tabla(s) con la(s) que se trabajara.
WHERE	Condiciona la acción que se realizara de acuerdo a un criterio especifico.
GROUP BY	Separar los registros seleccionados en grupos específicos.

Cláusula	Descripción
ORDER BY	Ordena las filas de acuerdo a un orden especifico.
HAVING	Es la cláusula que condiciona a ORDER BY.

Una cláusula trabaja bajo un criterio y un operador lógico o de comparación. Una sentencia DML sin cláusulas y operadores lógicos no es lo suficientemente completa como para manipular los datos de manera compleja.

A continuación presentamos los operadores disponibles en SQL:

Operadores Lógicos	Descripción
AND	"Y" lógico. Evalúa dos condiciones y devuelve verdadero si ambas son ciertas.
OR	"O" lógico. Evalúa dos condiciones y devuelve verdadero si alguna de ellas es cierta.
NOT	"No" lógico. Devuelve el valor contrario de la expresión.
Operadores de comparación	**Descripción**
<	Menor que
>	Mayor que
<>	Distinto de
<=	Menor o igual que
>=	Mayor o igual que
BETWEEN	Especifica un intervalo de valores
LIKE	Se emplea para comparar contra un patrón
IN	Especifica un conjunto de patrones.

Para hacer una consulta a una tabla se emplea la sentencia SELECT, cuya sintaxis es la siguiente:

Consulta básica	Consulta condicionada
SELECT columna1, ..., columnaN **FROM** nombre_tabla	**SELECT** columna1, ..-, columnaN **FROM** nombre_tabla **WHERE** expresión_condicional
Consulta varias tablas	**Consultar toda la tabla**
SELECT col1_tabla1, ..., colN_tablaN **FROM** nombre_tabla1, ..., nombre_tablaN	**SELECT * FROM** nombre_tabla
Consulta con alias	
SELECT Alias.columna1, ..., **Alias**.columnaN FROM nombre_tabla AS **Alias**	

Cuando deseamos insertar nuevos registros en una tabla debemos emplear la sentencia `INSERT INTO`, esta es la sintaxis:

Inserción básica	Inserción especifica
`INSER INTO nombre_tabla VALUES ('dato1', ..., 'datoN')`	`INSER INTO nombre_tabla (columna1, ..., columnaN) VALUES ('dato1', ..., 'datoN')`

`UPDATE` es la sentencia que actualiza los datos de una tabla, su sintaxis es:

Actualización de todos los registros	Actualización condicionada
`UPDATE nombre_tabla SET columna1 = 'dato1', ... columnaN = 'datoN'`	`UPDATE nombre_tabla SET columna1 = 'dato1', ... columnaN = 'datoN' WHERE expresión_condicional`

La eliminación se realiza con la instrucción `DELATE`, y tiene la siguiente sintaxis:

Eliminación de todos los registros	Eliminación condicional
`DELATE FROM nombre_tabla`	`DELATE FROM nombre_tabla WHERE expresión_condicional`
Eliminación limitada	
`DELATE FROM nombre_tabla WHERE expresión_condicional LIMIT limite`	

4.2 Tipos de datos y modificadores en MySQL

Los tipos de datos numéricos soportados por MySQL son los siguientes:

Tipo	Tamaño	Rango de valores	Sin signo
`TINYINT`	1 byte	-128 a 127	0 a 255
`SMALLINT`	2 bytes	-32768 a 32767	0 a 65535
`MEDIUMINT`	3 bytes	-8388608 a 8388607	0 a 16777215
`INT`	4 bytes	-2147483648 a 2147483647	0 a 4294967295
`BIGINT`	8 bytes	-9223372036854775808 a 9223372036854775807	0 a 18446744073709550615
`FLOAT(M, D)`	4 bytes	Varia según los valores	
`DOUBLE(M, D)`	8 bytes	Varia según los valores	
`DECIMAL(M, D)`	M+2 bytes	Varia según los valores	

Estos son los tipos de datos cadena que están soportados en MySQL:

Tipo	Tamaño máximo	Espacio en disco
CHAR(X)	255 bytes	X bytes
VARCHAR(X)	255 bytes	X+1 bytes
TINYTEXT	255 bytes	X+1 bytes
TINYBLOB	255 bytes	X+2 bytes
TEXT	65535 bytes	X+2 bytes
BLOB	65535 bytes	X+2 bytes
MEDIUMBLOB	1.6 MB	X+3 bytes
MEDIUMTEXT	1.6 MB	X+3 bytes
LONGTEXT	4.2 GB	X+4 bytes
LONGBLOB	4.2 GB	X+4 bytes

Además de los tipos de datos que hemos mencionado, existen los modificadores o atributos de columna:

Modificador	Tipos aplicables	Descripción
AUTO_INCREMENT	Todos los tipos INT	Incrementa automáticamente en uno el valor de una campo después de la inserción.
BINARY	CHAR, VARCHAR	Cadenas binarias.
DEFAULT	Todos, excepto BLOB y TEXT	Especifica el valor por *default* de una columna.
NOT NULL	Todos los tipos	Especifica que una columna debe tener valores.
NULL	Todos los tipos	Especifica que una columna debe o no tener valores.
PRIMARY KEY	Todos los tipos	Especifica que la columna será llave primaria.
UNIQUE	Todos los tipos	Especifica que los datos de la columna deben ser únicos.
UNSIGNED	Tipos numéricos	Tipo de dato sin signo.
ZEROFILL	Tipos numéricos	Desplegar ceros a la izquierda de un número.

4.3 Las funciones `mysql_xxx` de PHP

PHP ha integrado a su amplia lista de funciones más de 40 funciones para interactuar con un servidor de base de datos MySQL. A continuación mostraremos sólo las más importantes funciones `mysql_xxx()`, para una referencia más amplia recomendamos ir al manual de PHP y consultar la sección de "Funciones MySQL"[5].

5 http://www.php.net/manual/es/ref.mysql.php

Función	Descripción
Conexión y desconexión	
mysql_connect	Abre una Conexión con el servidor MySQL
mysql_pconnect	Abre una Conexión persistente con MySQL
mysql_select_db	Selecciona una base de datos
mysql_close	Cierra la conexión con el servidor MySQL
mysql_change_user	Cambia al usuario de la conexión activa
Creación y eliminación de bases de datos	
mysql_create_db	Crear una base de datos
mysql_drop_db	Elimina una base de datos
Realización de consultas	
mysql_db_query	Envía una sentencia MySQL al servidor
mysql_query	Envía una sentencia SQL al servidor
Manejo y resultados de las consultas	
mysql_fetch_array	Extrae la fila de resultado como una array asociativo, un array numérico o ambos
mysql_result	Devuelve los datos de una consulta
mysql_fetch_row	Devuelve una fila de resultado como matriz
mysql_afected_rows	Devuelve el número de filas que fueron afectadas por la operación anterior: INSERT/UPDATE/DELETE
mysql_num_rows	Devuelve el número de filas de un resultado
mysql_fetch_field	Extrae la información de una columna y la devuelve como un objeto
mysql_fetch_lengths	Devuelve la longitud de cada salida en un resultado
mysql_fetch_object	Extrae una fila de resultado como un objeto
mysql_field_name	Devuelve el nombre del campo especificado en un resultado
mysql_list_fields	Lista los campos del resultado de MySQL
mysql_num_fields	Devuelve el número de campos de un resultado.
mysql_field_seek	Asigna el puntero del resultado al desplazamiento del campo especificado.
mysql_field_type	Devuelve el tipo del campo especificado en un resultado.
mysql_field_flags	Devuelve las banderas asociados con el campo especificado en un resultado.
mysql_insert_id	Devuelve el ID generado en la operación anterior
mysql_data_seek	Mueve el puntero interno

Función	Descripción
mysql_free_result	Libera la memoria del resultado
Manejo de errores	
mysql_errno	Devuelve el número del mensaje de error de la última operación MySQL
mysql_error	Devuelve el texto del mensaje de error de la última operación MySQL
Información de la base de datos	
mysql_list_dbs	Lista las bases de datos disponibles en el servidor MySQL
mysql_list_tables	Lista las tablas en una base de datos MySQL
mysql_field_len	Devuelve la longitud del campo especificado
mysql_field_table	Devuelve el nombre de la tabla donde está el campo especificado
mysql_tablename	Devuelve el nombre de la tabla de un campo

4.3.1 Insertar registros

Durante el Diplomado de "Desarrollo e Implementación de Sistemas con Software Libre en Linux" se realizaron diversos ejercicios utilizando los recursos que hasta el momento hemos tratado. Las principales operaciones que una aplicación web realiza en una base de datos son: altas, bajas, consultas y actualizaciones.
Con el fin de mostrar un ejemplo didáctico y explicito vamos a mostrar el uso de las funciones `mysql_xxx()` más utilizadas en los ejercicios realizados.

Primero mostraremos la forma de ingresar información proveniente de un formulario HTML a una tabla ubicada en la base de datos; después, la forma de realizar consultas y mostrarlas en HTML, también, aprenderemos a eliminar registros; finalmente, trataremos la actualización de registros.

Lo primero que debemos hacer antes de trabajar con la base de datos es realizar la conexión con el servidor MySQL empleando la función `mysql_connect()`. La sintaxis es:

```
mysql_connect(nombre_host, usuario, password)
```

Esta función devuelve un identificador de conexión que es empleado por algunas otras funciones `mysql_xxx()` para realizar operaciones con la base de datos, si la conexión no se puede realizar devolverá un valor `false`.

La siguiente operación es seleccionar la base de datos con la que trabajaremos, empleamos la función `mysql_db_select()`, cuya sintaxis es:

```
mysql_db_select(nombre_basedatos, id_conexion)
```

Si la operación se realiza con éxito devuelve un valor `true` y `false` en caso contrario.

El siguiente paso será ejecutar alguna sentencias SQL y procesar los resultados, esto se logra mediante la función `mysql_query()`:

```
mysql_query(sentencia_SQL, id_conexion)
mysql_query(nombre_base_datos, sentencia_SQL, id_conexion)
```

`mysql_query()` devuelve un tipo de dato `resource` cuando realiza sentencias `SELECT`, `SHOW`, `DESCRIBE` o `EXPLAIN`. Para sentencias como `INSERT`, `UPDATE`, `DELETE`, `DROP`, etc. Devuelve un valor `true` cuando se realizó con éxito la operación y `false` en caso contrario.
La función `mysql_db_select()` se desaprobó desde la versión PHP 4.0.6, de modo que no se recomienda su uso.

Finalmente, debemos cerrar la conexión con el servidor de base de datos:

```
mysql_close(id_conexion)
```

Trabajaremos sobre una sencilla tabla llamada `LibroVisitas`, ubicada en la base de datos `MiTienda`, y un formulario HTML que nos permitirá ingresar nuevos registros a la tabla.

librovisitas.sql

```
USE MiTienda;

CREATE TABLE LibroVisitas (
id_comentario INT NOT NULL AUTO_INCREMENT ,
nombre VARCHAR( 40 ) NOT NULL ,
email VARCHAR( 40 ) NOT NULL ,
comentario VARCHAR( 200 ) NOT NULL ,
PRIMARY KEY ( id_comentario )
);
```

insertar_form.html

```
<!DOCTYPE...>
<html>
<head>
<title>Libro de Visitas</title>
</head>
<body>
<center><h1>Libro de Visitas</h1></center>
<hr />
<h3>Env&iacute;anos tus comentarios.</h3>
<br />
<form method="post" action="insertar.php">
Tu nombre: <input type="text" id="nombre" name="nombre" />
<br />
Tu e-mail: <input type="text" id="email" name="email" />
<br />
Comentarios: <textarea id="comentario" name="comentario" rows="5" cols="30">Aqui tus
comentarios...</textarea>
<br />
<input type="hidden" id="submit_ok" name="submit_ok" value="true" />
<input type="submit" value="Enviar" />
<input type="reset" value="Reset" />
</form>
</body>
</html>
```

Salida

El script que se encargará de procesar el formulario se llama `insertar.php`, primero comprueba que el campo oculto en el formulario haya sido enviado, después extrae y filtra cada uno de los datos que el usuario proporciono y finalmente se ingresan a la base de datos.

insertar.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title>Gracias por tus comentarios</title>
</head>
<body>
<center><h1>Libro de Visitas</h1></center>
<hr />
<?php

      if(isset($_POST['submit_ok']))
      {
            $db_user = 'root';
            $db_pass = '';
            //Realizamos la conexion
            $id_conexion = mysql_connect('localhost', $db_user, $db_pass);
            if(!$id_conexion)
            {
                  die('Es imposible conectarse al servidor: ' . mysql_error());
            }

            //Seleccionamos la base de datos
            $db_status = mysql_select_db('MiTienda', $id_conexion);
```

```
            if (!$db_status) {
             die ('Es imposible trabajar con la base de datos' . mysql_error());
            }

            //Obtenemos la informacion del formulario
            //y filtramos los caracteres especiales
            $nombre = htmlentities($_POST['nombre']);
            $email = htmlentities($_POST['email']);
            $comentario = htmlentities($_POST['comentario']);

            //Insertamos los datos
            $query = "INSERT INTO LibroVisitas VALUES (0, '$nombre','$email',
'$comentario')";
            $in_status = mysql_query($query, $id_conexion);

            if(!$in_status)
            {
                  die('Es imposible insertar comentarios: ' . mysql_error());
            }

            print "<b>$nombre</b> gracias por tus comentarios";

            //cerramos la conexion
            mysql_close($id_conexion);
      }
      else
      {
            print "No hay comentarios que insertar";
      }
?>
</body>
<html>
```

Salida

4.3.2 Consultar registros

Toca el turno para mostrar los resultados de una sencilla consulta `SELECT`, seguiremos los mismos pasos que hicimos la inserción:

- Establecer una conexión con el servidor MySQL
- Seleccionar la base de datos con la que trabajaremos
- Realizar la consulta
- Procesar los resultados obtenidos
- Cerrar la conexión

mostrarvisitas.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title>Mostrar Visitas</title>
</head>
<body>
<center><h1>Los comentarios de nuestras visitas</h1></center>
<hr />
<?php
          $db_user = 'root';
          $db_pass = '';

          //Realizamos la conexion
          $id_conexion = mysql_connect('localhost', $db_user, $db_pass);
          if(!$id_conexion)
          {
               die('Es imposible conectarse al servidor: ' . mysql_error());
          }

          //Seleccionamos la base de datos
          $db_status = mysql_select_db('MiTienda', $id_conexion);
          if (!$db_status) {
           die ('Es imposible trabajar con la base de datos' . mysql_error());
          }

          //Realizamos la consulta
          $query = "SELECT nombre, email, comentario FROM LibroVisitas";
          $res = mysql_query($query, $id_conexion);

          if(!$res)
          {
               die('Es imposible realizar la consulta: ' . mysql_error());
          }
               //Procesamos los resultados
          print <<<_HTML_
     <table border="1">
     <tr>
     <th>Nombre</th>
     <th>Email</th>
     <th>Comentario</th>
     </tr>
```

```
_HTML_;

            while($row = mysql_fetch_array($res))
            {
                  print <<<_HTML_
                  <tr>
                  <td>$row[nombre]</td>
                  <td>$row[email]</td>
                  <td>$row[comentario]</td>
                  </tr>
_HTML_;
            }
            print '</table>';
            //Cerramos la conexion
            mysql_close($id_conexion);
?>
</body>
<html>
```

Salida

4.3.3 Eliminar registros

Para eliminar un registro debemos hacer uso la llave primaria de cada uno, ésta se asocia a un cuadro de verificación (*checkbox*), y de esta forma el usuario selecciona los registros que desea eliminar. El siguiente script puede realizar dos cosas: mostrar los registros a eliminar y eliminar los registros seleccionados.

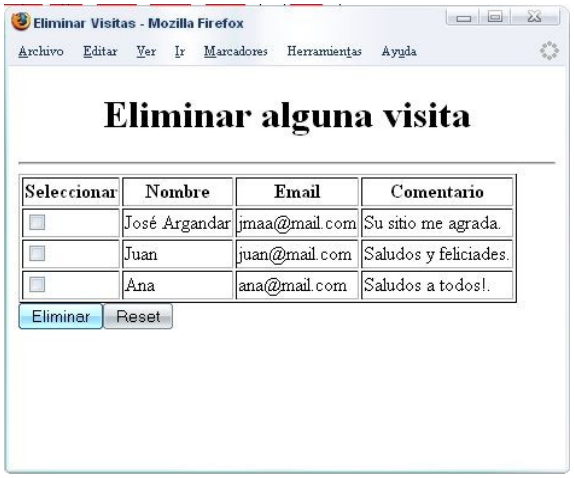
eliminar.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title>Eliminar Visitas</title>
</head>
<body>
<center><h1>Eliminar alguna visita</h1></center>
```

```
<hr />
<?php

	$db_user = 'root';
	$db_pass = '';



	//Realizamos la conexion
	$id_conexion = mysql_connect('localhost', $db_user, $db_pass);
	if(!$id_conexion)
	{
		die('Es imposible conectarse al servidor: ' . mysql_error());
	}



	//Seleccionamos la base de datos
	$db_status = mysql_select_db('MiTienda', $id_conexion);
	if (!$db_status)
	{
		die ('Es imposible trabajar con la base de datos' . mysql_error());
	}


	//  --------- Eliminar registros -----------------------
	if(isset($_POST['submit_ok']))
	{
		foreach($_POST['visita'] as $id_comentario)
		{
			$query = "DELETE FROM LibroVisitas WHERE id_comentario =
'$id_comentario'";
			$del_status = mysql_query($query, $id_conexion);
			if(!$del_status)
			{
				die('Es imposible eliminar el registro: ' .
mysql_error());
			}
		}
	}


	// --------- Mostramos los registros ------------------

	//Realizamos la consulta
	$query = "SELECT id_comentario, nombre, email, comentario FROM
LibroVisitas";
	$res = mysql_query($query, $id_conexion);

	if(!$res)
	{
		die('Es imposible realizar la consulta: ' . mysql_error());
	}
```

```
      //Procesamos los resultados
      print <<<_HTML_
      <form method="post" action="$_SERVER[PHP_SELF]">
      <table border="1">
      <tr>
      <th>Seleccionar</th>
      <th>Nombre</th>
      <th>Email</th>
      <th>Comentario</th>
      </tr>
_HTML_;

      while($row = mysql_fetch_array($res))
      {
            print <<<_HTML_
            <tr>
            <td><input     type="checkbox"     id="visita[]"     name="visita[]"
value="$row[id_comentario]" /></td>
            <td>$row[nombre]</td>
            <td>$row[email]</td>
            <td>$row[comentario]</td>
            </tr>
_HTML_;
      }
            print '</table>';
            print    '<input    type="hidden"    id="submit_ok"    name="submit_ok"
value="true" />';
            print '<input type="submit" value="Eliminar" />';
            print '<input type="reset" value="Reset" /></from>';

            //Cerramos la conexion
            mysql_close($id_conexion);

?>

</body>
<html>
```

Salida



4.3.4 Actualizar registros

El actualizar un registro incrementa la complejidad del script, ya que involucra mostrarle al usuario los registros disponibles, permitirle seleccionar el de su interés y modificar los datos que contiene.
El script `actualizar.php` presenta los registros disponibles en la tabla `LibroVisitas`, el usuario deberá seleccionar uno y presionar el botón `Seleccionar`. Los datos son capturados por el script `actualizar_form.php` en donde el usuario realiza las modificaciones requeridas y guarda los cambios.

actualizar.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title>Actualizar Libro de Visitas</title>
</head>
<body>
<center><h1>Actualizar Libro de Visita</h1></center>
<hr />
<?php
      $db_user = 'root';
      $db_pass = '';

      //Realizamos la conexion
      $id_conexion = mysql_connect('localhost', $db_user, $db_pass);
      if(!$id_conexion)
      {
            die('Es imposible conectarse al servidor: ' . mysql_error());
      }

      //Seleccionamos la base de datos
      $db_status = mysql_select_db('MiTienda', $id_conexion);
      if (!$db_status)
      {
            die ('Es imposible trabajar con la base de datos' . mysql_error());
      }


      // --------- Mostramos los registros -----------------

      //Realizamos la consulta
      $query = "SELECT id_comentario, nombre, email, comentario FROM
LibroVisitas";
      $res = mysql_query($query, $id_conexion);

      if(!$res)
      {
            die('Es imposible realizar la consulta: ' . mysql_error());
      }

      //Procesamos los resultados
      print <<<_HTML_
      <form method="post" action="actualizar_form.php">
      <table border="1">
```

```
      <tr>
      <th>Actualizar</th>
      <th>Nombre</th>
      <th>Email</th>
      <th>Comentario</th>
      </tr>
_HTML_;

      while($row = mysql_fetch_array($res))
      {
            print <<<_HTML_
            <tr>
            <td><input type="radio" id="visita" name="visita"
value="$row[id_comentario]" /></td>
            <td>$row[nombre]</td>
            <td>$row[email]</td>
            <td>$row[comentario]</td>
            </tr>
_HTML_;
      }
            print '</table>';
            print '<input type="hidden" id="submit_ok" name="submit_ok"
value="true" />';
            print '<input type="submit" value="Seleccionar" />';
            print '<input type="reset" value="Reset" /></from>';

            //Cerramos la conexion
            mysql_close($id_conexion);

?>
</body>
<html>
```

actualizar_form.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title>Actualizar Libro de Visitas</title>
</head>
<body>
<center><h1>Actualizar Libro de Visita</h1></center>
<hr />
<?php
      $db_user = 'root';
      $db_pass = '';

      //Realizamos la conexion
      $id_conexion = mysql_connect('localhost', $db_user, $db_pass);
      if(!$id_conexion)
      {
            die('Es imposible conectarse al servidor: ' . mysql_error());
      }

      //Seleccionamos la base de datos
      $db_status = mysql_select_db('MiTienda', $id_conexion);
```

```
	if (!$db_status)
	{
		die ('Es imposible trabajar con la base de datos' . mysql_error());
	}

	// ------------- Realizamos la Actualizacion ---------------
	if(isset($_POST['update_ok']))
	{
		//Obtenemos la informacion del formulario
		//filtrando los caracteres especiales
		$nombre = htmlentities($_POST['nombre']);
		$email = htmlentities($_POST['email']);
		$comentario = htmlentities($_POST['comentario']);
		$id_comentario = $_POST['id_comentario'];

		$query = "UPDATE LibroVisitas SET nombre = '$nombre', email = '$email',
comentario = '$comentario'";
		$query .= " WHERE id_comentario = $id_comentario";

		$upd_status = mysql_query($query, $id_conexion);

		if(!$upd_status)
		{
			die('Es imposible actualizar el registro: ' . mysql_error());
		}

		print "<b>Registro Actualizado</b><br />";
		print '<a href="actualizar.php">Regresar</a>';
	}

	// ------------- Mostramos el formulario de actualizacion ------------
	else if(isset($_POST['submit_ok']))
	{
		//Realizamos la consulta
		$id_comentario = $_POST['visita'];
		$query = "SELECT nombre, email, comentario FROM LibroVisitas WHERE
id_comentario = $id_comentario";
		$res = mysql_query($query, $id_conexion);

		if(!$res)
		{
			die('Es imposible realizar la consulta: ' . mysql_error());
		}

		//Procesamos los resultados
		print <<<_HTML_
		<form method="post" action="$_SERVER[PHP_SELF]">

_HTML_;

		while($row = mysql_fetch_array($res))
		{
			print <<<_HTML_
			Nombre: <input type="text" id="nombre" name="nombre"
value="$row[nombre]"/>
			<br />
			e-mail: <input type="text" id="email" name="email" value="$row[email]"
/>
			<br />
```

```
                Comentarios: <textarea id="comentario" name="comentario" rows="5"
cols="30">$row[comentario]</textarea>
                <br />
_HTML_;
            }
            print <<<_HTML_
            <input type="hidden" id="update_ok" name="update_ok" value="true" />
            <input type="hidden" id="id_comentario" name="id_comentario"
value="$id_comentario" />
            <input type="submit" value="Actualizar" />
            <input type="reset" value="Reset" />
            </from>
_HTML_;

      }
      else
      {
            print "<b>No hay registro que actualizar</b>";
      }
            //Cerramos la conexion
            mysql_close($id_conexion);

?>
</body>
<html>
```

Salida

1

2

3

4

Resumen

Debido a la popularidad que MySQL ha tenido en el mundo de las Bases de Datos se le suele confundir con el lenguaje SQL. MySQL es un Manejador de Bases de Datos que administra, registra, controla y manipula datos; mientras que SQL es un lenguaje de consultas desarrollado por IBM.

La teoría matemática sobre la cual están sustentadas las base de datos relacionales se le debe al matemático inglés Edgar Frank Codd, que en 1970 publicó en un trabajo titulado *A Relational Model of Data for Large Shared Data Banks*.

A lo largo de éste capítulo analizamos el conjunto de instrucciones de SQL más importantes, los tipos de datos de MySQL y las funciones con que PHP interactúa con MySQL. Los tipos de datos y sus tamaños suelen variar entre los diversos Manejadores de Bases de Datos, MySQL esta equipado con un aceptable conjunto de tipos de datos.

Sin duda alguna, la integración que tiene MySQL con PHP es de las mejores: el API está muy bien documentada, traducida a varios idiomas y ofrece una serie de ejemplos muy ilustrativos.
Sin embargo no es la única forma en que PHP puede interactuar con MySQL, existe un *framework* llamado PEAR(*PHP Extension and Application Repository*) cuya principal característica es la orientación a objetos de sus componentes.

En el siguiente capítulo hablaremos veremos sobre PostgreSQL, un manejador de bases de datos mucho más robusto que MySQL.

Capítulo 5.

Interacción de PHP con PostgreSQL.

Apenas Gandalf hubo dicho estas palabras cuando se oyó un gran ruido, como si algo rodara retumbando en los abismos lejanos, estremeciendo el suelo de piedra.

Tolkien,J. R. R.. La comunidad del anillo

5.1 Breve historia de PostgreSQL

PostgreSQL tiene sus orígenes en el proyecto Ingres de la Universidad de Berkeley, comercializado en 1982 por Michael Stonebraker. En 1986 Stonebraker continuó el desarrollo de Ingres liberando una serie de documentos que describían la arquitectura del sistema de bases de datos objeto-relacional denominado Postgres. En 1994, Andrew Yu y Jolly Chen agregaron un interprete de SQL a Postgres, cambiando asi el nombre a PostgreSQL95, fue entonces cuando se libero bajo la licencia publica BSD.

Actualmente ha sido portado a GNU/Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64), y Windows. Soporta los tipos de datos especificados por ANSI SQL92 y SQL99, cuenta con características que otros manejadores de bases de datos comerciales ofrecen: foreign keys, joins, views, triggers, y procedimientos almacenados (escritos en Java, Perl, Python, Ruby, Tcl, C/C++, PL/pgSQL, etc.).

5.2 Instalación de PostgreSQL

Sin duda alguna PostgreSQL ha demostrado un mejor rendimiento en sistemas GNU/Linux, pues nació en sistemas UNIX, de modo que realizaremos su instalación en Slackware 10.2.

Lo primero será descargar la ultima versión estable de PostgreSQL del sitio oficial[1], en éste caso la 8.1.5.

- Desde un terminal descomprimimos y desempaquetamos:
  `$tar -xzvf postgresql-8.1.5.tar.gz`
- Ingresamos al directorio `postgresql-8.1.5` y ejecutamos el script de configuración: `$./configure`
- Realizamos la construcción: `$gmake`
- Nos cambiamos a la cuenta de superusuario: `$su`
- Realizamos la comprobación de la construcción: `#gmake check`
- Realizamos la instalación: `#gmake install`

Hasta aquí no debería haber ocurrido ningún error y el servidor se ha instalado en `/usr/local/pgsql` y está casi listo para correr, pero es necesario agregar un usuario que ejecute el servidor:

- Agregar usuario: `#adduser postgres`
- Colocar password: `#passwd postgres`

El siguiente paso es crear un clúster de bases de datos de PostgreSQL (PostgreSQL database cluster), que contendrá las bases de datos que vayamos creando. Al momento de inicializar un clúster de bases de datos de PostgreSQL se tiene que indicar el directorio donde se desea que se cree. El propietario de ese directorio tiene que ser un usuario que no sea root.

- Creamos el directorio: `#mkdir /usr/local/pgsql/data`
- Cambiamos el propietario del directorio:
  `#chown postgres /usr/local/pgsql/data`

1 http://www.postgresql.org/download/

Nos cambiamos al usuario `postgres` e inicializamos el clúster.

- Cambiamos de usuario: `#su postgres`
- Inicializamos el clúster: `$/usr/local/pgsql/bin/initdb -D /usr/local/pgsql/data`

Ahora si estamos listos para arrancar el servidor con el usuario `postgres`. Por comodidad es recomendable editar el archivo `.bash_profile` del usuario `postgres` para agregar el `path` y no tener que escribir rutas largas a archivos o directorios:

```
PGDATA=/usr/local/pgsql/data
PATH=/usr/local/pgsql/bin:$PATH
export PATH PGDATA
```

Hay varias formas de arrancar el servidor PostgreSQL.

- Levantar en un puerto especificado:
  `$postmaster -i -p puerto -D $PGDATA &`
- Levantar en el puerto por default: `$pg_ctl -D $PGDATA start`

Finalmente, para detener el servidor: `$pg_ctl -D $PGDATA stop`

Para que el intérprete de PHP tenga soporte de las funciones de PostgreSQL, debe compilarse PHP con los siguientes parámetros:

```
./configure --with.apxs2=/usr/local/apache2/bin/apxs --with-pgsql
```

5.3 El shell de PostgreSQL

Existen diversas herramientas que nos permiten administrar un servidor PostgreSQL, las más destacadas son pgAdmin[2] y phpPgAdmin[3]. La primera es una aplicación *stand alone* y la segunda es una aplicación web.

Como no siempre es posible contar con una terminal gráfica, PostgreSQL cuenta con una terminal de texto o shell para realizar la administración del servidor. Para entrar al dicho shell debemos entrar con el usuario `postgres` y después ejecutar el comando `psql`.

Los comando más útiles son los siguientes:

Comando	Descripción
`\?`	Ayuda sobre comandos
`\d`	Muestra las tablas disponibles
`\d nombre_tabla`	Muestra la estructura de la tabla
`\h`	Ayuda sobre sentencias SQL
`\i`	Procesar un archivo externo con instrucciones SQL
`\q`	Salir del shell PostgreSQL
`\z nombre_objeto`	Muestra los permisos de un objeto determinado

2 http://www.pgadmin.org/
3 http://phppgadmin.sourceforge.net/index.php

5.4 Gestión de usuarios y grupos

La conexión a PostgreSQL se realiza a través de un usuario registrado previamente en el servidor, estos usuarios son independientes a los registrados en el sistema operativo. La tabla de usuarios controla los permisos de acceso y las acciones que puede o no realizar cada usuario con las base de datos. Cada usuario cuenta con un ID interno a PostgreSQL llamado `sysid`, y una contraseña. Éste ID es utilizado para asociar objetos de la base de datos con su propietario (quien puede dar o quitar privilegios sobre el objeto).

Existen dos formas de **crear usuarios**, mediante una sentencia SQL que se introduce dese el shell de PostgreSQL o mediante un comando que se ejecuta desde el shell del sistema operativo. Los usuarios se almacenan en la tabla `pg_user` y `pg_shadow` del catalogo del sistema.

La sentencia SQL tiene la sintaxis siguiente:

```
CREATE USER nombre_usuario
[WITH [SYSID uid] [PASSWORD 'passwd']]
[CREATEDB|NOCREATEDB]
[CREATEUSER|CREATEUSER]
[IN GROUP grname[,...]]
[VALID UNTIL 'aaaa-mm-dd']
```

Así para crear al usuario `jose` y que a su vez pueda crear otros usuarios, tenemos:

```
CREATE USER jose CREATEUSER
```

Un usuario con más restricciones, por ejemplo `jmargandar`, que sólo tendrá acceso hasta fin de año:

```
CREATE USER jmargandar WITH PASSWORD '1a2b3c4567' VALID UNTIL '2006-11-31'
```

Cuando es invocado el comando `createuser`, se abre un modo interactivo que solicita ciertos parámetros.

Los usuarios existentes sólo pueden ser **modificados** por superusuarios, las modificaciones posibles son las que se definieron durante la creación.

```
ALTER USER nombre_usuario
[WITH [PASSWORD 'passwd']]
[CREATEDB|NOCREATEDB]
[CREATEUSER|CREATEUSER]
[VALID UNTIL 'aaaa-mm-dd']
```

Para **eliminar un usuario** debemos asegurarnos que no existan bases de datos que pertenezcan a dicho usuario. Hay dos formas de llevar a cabo ésta operación, con la sentencia `DROP USER nombre_usuario` o con el comando `dropuser -U nombre_superusuario nombre_usuario_a_eliminar`.

Los grupos simplifican la asignación de privilegios, de otro modo a cada usuario habría que otorgarle cada uno de los privilegios. Los grupos se almacenen en la tabla `pg_group`. Para **crear un grupo** se tiene la sentencia SQL:

```
CREATE GROUP nombre_grupo
[WITH [SYSID gid]]
[USER user1 [,...]]
```

Por ejemplo, si deseamos agregar un grupo denominado `geeks` y que los usuarios `jose` y `jmargandar` pertenezcan a él:

```
CREATE GROUP geeks WITH USER jose, jmargandar
```

Para **eliminar un grupo** se emplea la sentencia SQL `DROP GROUP nombre_grupo`. Si deseamos eliminar o agregar un usuario a un grupo tenemos la sentencia:

```
ALTER GROUP nombre_grupo {ADD|DROP} USER nombre_usuario[,...]
```

Los **permisos** pueden ser otorgados tanto a tablas, como a vistas o sobre cualquier otro objeto dentro del servidor, mediante la sentencia:

```
GRANT privilegio[,...] ON objeto[,...] TO {PUBLIC|nombre_usuario|GROUP nombre_grupo}
```

Donde los privilegios son:

Privilegio	Símbolo	Descripción
`SELECT`	`r`	Lectura
`INSERT`	`a`	Inserción
`UPDATE, DELETE`	`w`	Escritura
`RULE`	`R`	Crear regla
`ALL`	`arwR`	Todos los anteriores

De manera similar para **remover los permisos** tenemos la sentencia:

```
REVOKE privilegio[,...] ON objeto[,...] FROM {PUBLIC|nombre_usuario|GROUP nombre_grupo}
```

5.5 Tipos de datos en PostgreSQL

PostgreSQL cuenta con una amplia variedad de tipos de datos, como son los tipos numéricos, monetarios, carácter, binarios, booleanos, fecha/tiempo, geométricos, direcciones de red, etc.; también permite definir tipos de datos por el usuario. Una referencia más extensa sobre los tipos de datos disponibles se localiza en la documentación oficial[4].
A continuación mostraremos los tipos de datos más utilizados.

Nombre	Alias	Descripción
`bigint`	`int8`	Entero con signo de 8 bytes
`bigserial`	`serial8`	Entero de 8 bytes autoincrementable
`bit(n)`		Cadena de bits de tamaño fijo `n`
`bit varying(n)`	`varbit`	Cadena de bits de tamaño variable
`boolean`	`bool`	Valor lógico, `true` o `false`
`box`		Caja rectangular sobre el plano

4 http://www.postgresql.org/docs/8.1/static/datatype.html

Nombre	Alias	Descripción
bytea		Byte array o dato binario
character varying(n)	varchar(n)	Cadena de caracteres de tamaño variable
character(n)	char(n)	Cadena de caracteres de tamaño fijo n
cidr		Dirección de red IPv4 o IPv6
circle		Circulo sobre el plano
date		Fecha (año-mes-día)
double precision	float8	Numero flotante de doble precisión
inet		Dirección IP de IPv4 o IPv6
integer	int, int4	Entero con signo de 4 bytes
interval(p)		Tiempo de duración
line		Línea infinita sobre el plano
lseg		Segmento de línea sobre el plano
macaddr		Dirección MAC
money		Moneda
numeric(p, s)	decimal(p, s)	Numero con precisión seleccionable
path		Ruta de puntos sobre el plano
point		Punto sobre el plano
polygon		Ruta cerrada sobre el plano
real	float4	Numero de punto flotante
smallint	int2	Entero de 2 bytes
serial	serial4	Entero de 4 bytes autoincrementable
text		Cadena de caracteres de tamaño variable
time(p) sin zona horaria		Hora del día
time(p) con zona horaria	timetz	Hora del día con zona horaria
timestamp(p) sin zona horaria		Fecha y hora del día
timestamp(p) con zona horaria	timestamptz	Fecha y hora del día con zona horaria

5.6 Las funciones pg_xxx en PHP

La interacción de PHP con el servidor de base de datos PostgreSQL se logra mediante un las funciones pg_xxx(). El nombre de estas funciones nos debe sonar familiar, ya que son muy similares a las de la API de MySQL y funcionan prácticamente igual. La siguiente es una lista de las funciones más importantes, en la documentación de PHP sobre las funciones PostgreSQL[5] se detalla cada una de ellas.

5 http://www.php.net/manual/es/ref.pgsql.php

Función	Descripción
Conexión y desconexión	
pg_connect	Establece la conexión con el servidor
pg_pconnect	Establece una conexión persistente con el servidor
pg_connection_busy	Verifica si la conexión esta ocupada o no
pg_connection_reset	Reinicia la conexión con el servidor
pg_connection_status	Devuelve el estado de la conexión
pg_close	Cierra la conexión con el servidor
Realización de consultas	
pg_query	Envía una sentencia SQL al servidor
pg_prepare	Prepara una consulta
pg_execute	Realiza la consulta preparada
pg_insert	Inserta un registro con los valores proporcionados por un arreglo.
pg_delete	Elimina registros de acuerdo a las llaves y valores de un arreglo asociativo.
pg_select	Selecciona los registros de acuerdo un arreglo asociativo.
pg_update	Actualiza registros de acuerdo a un array de datos y un array de condiciones.
Manejo y resultados de las consultas	
pg_fetch_array	Extrae la fila de resultado como una array asociativo, un array numérico o ambos
pg_fetch_row	Devuelve una fila de resultado como matriz
pg_afected_rows	Devuelve el numero de filas que fueron afectadas por la operación anterior: INSERT/UPDATE/DELETE
pg_num_rows	Devuelve el número de filas de un resultado
pg_fetch_all_columns	Devuelve una arreglo que contiene todos los registros de una columna especificada
pg_fetch_all	Devuelve un arreglo con todas filas del resultado
pg_fetch_assoc	Devuelve una fila como arreglo asociativo
pg_field_name	Devuelve el nombre del campo específicado en un resultado
pg_num_fields	Devuelve el número de campos de un resultado.
pg_field_type	Devuelve el tipo del campo especificado en un resultado.
pg_free_result	Libera la memoria del resultado
Manejo de errores	

Función	Descripción
pg_last_error	Devuelve el ultimo mensaje de error.
pg_error_result	Devuelve el mensaje de error asociado a un resultado
pg_last_notice	Devuelve el ultimo mensaje generado por el servidor
Información del servidor	
pg_dbname	Nombre de la base de datos utilizada
pg_ping	Verifica la conexión
pg_port	Devuelve el puerto de conexión
pg_version	Devuelve el numero de versión del servidor

5.6.1 Insertar registros

Lo que a continuación haremos será modificar el código del libro de visitas que ya habíamos trabajado en el capítulo dedicado a MySQL y lo adaptaremos para que pueda interactuar con el servidor PostgreSQL.

Debemos crear una base de datos llamada `MiTienda`, conformada por una tabla llamada `LibroVisitas`.

mitienda.sql

```
CREATE TABLE LibroVisitas (
id_comentario INT NOT NULL,
nombre VARCHAR( 40 ) NOT NULL ,
email VARCHAR( 40 ) NOT NULL ,
comentario VARCHAR( 200 ) NOT NULL ,
PRIMARY KEY ( id_comentario )
);

$su postgres
$createdb MiTienda
$psql MiTienda
MiTienda=# \i mitienda.sql
MiTienda=# \q
```

El formulario para insertar registros, `insertar_form.html`, permanece sin modificaciones, mientras que el script `insertar.php` tendrá algunos cambios. A diferencia de MySQL, PostgreSQL no soporta AUTO_INCREMENT, de modo que no es posible delegar la tarea de generar valores únicos automáticamente para cada registro. La solución es generarlos nosotros mismo:

- Obtener el ultimo id generado (el valor más grande)
- Incrementarlo en una unidad

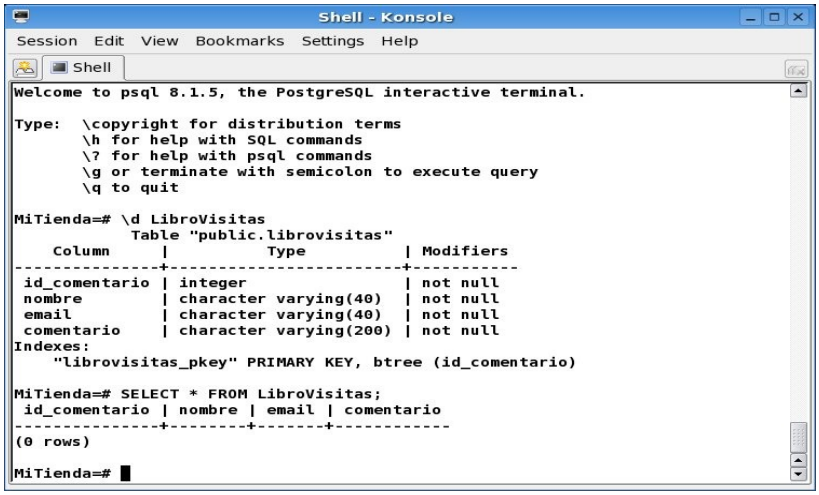
insertar.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title>Gracias por tus comentarios</title>
</head>
<body>
<center><h1>Libro de Visitas</h1></center>
```

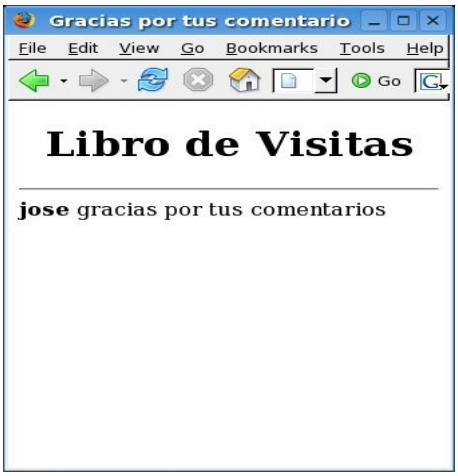
```
<hr />
<?php

      if(isset($_POST['submit_ok']))
      {
            $db_user = 'postgres';
            $db_pass = 'postgres';

            //Realizamos la conexion
            $id_conexion = pg_connect("dbname=MiTienda user=$db_user"
password=$db_pass);
            if(!$id_conexion)
            {
                  die('Es imposible conectarse al servidor: ' . pg_last_error());
            }

            //Generar una llave primaria
            //Obtenemos el ultimo Id
            $query = 'SELECT MAX(id_comentario) FROM LibroVisitas';
            $res = pg_query($id_conexion, $query);
            if(!$res)
            {
                  die('Imposible obtener el ultimo id: ' .  pg_last_error());
            }
            $currId = pg_fetch_row($res);
            //Lo incrementamos una unidad
            $nextId = $currId[0] + 1;

            //Obtenemos la informacion del formulario
            //filtrando los caracteres especiales
            $nombre = htmlentities($_POST['nombre']);
            $email = htmlentities($_POST['email']);
            $comentario = htmlentities($_POST['comentario']);

            //Insertamos los datos
            $query = "INSERT INTO LibroVisitas VALUES ($nextId,
'$nombre','$email', '$comentario')";
            $in_status = pg_query($id_conexion, $query);

            if(!$in_status)
            {
                  die('Es imposible insertar comentarios: ' . pg_last_error());
            }

            print "<b>$nombre</b> gracias por tus comentarios";

            //cerramos la conexion
            pg_close($id_conexion);
      }
      else
      {
            print "No hay comentarios que insertar";
      }
?>
```

```
</body>
<html>
```

Si realizamos una consulta a la tabla dentro del shell de PostgreSQL, tendremos la siguiente salida:



Después de insertar algunos datos:




```
MiTienda=# SELECT * FROM LibroVisitas;
 id_comentario | nombre |     email     |     comentario
---------------+--------+---------------+--------------------
             1 | jose   | jose@mail.com | Saludos nuevamente
(1 row)
```

5.6.2 Consultar registros

El script `mostrarvisita.php` sólo sufre algunas modificaciones.

mostrarvisita.php

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

```
<title>Mostrar Visitas</title>
</head>
<body>
<center><h1>Los comentarios de nuestras visitas</h1></center>
<hr />
<?php

          $db_user = 'postgres';
          $db_pass = 'postgres';

          //Realizamos la conexion
          $id_conexion = pg_connect("dbname=MiTienda user=$db_user
password=$db_pass");
          if(!$id_conexion)
          {
               die('Es imposible conectarse al servidor: ' . pg_last_error());
          }

          //Realizamos la consulta
          $query = "SELECT nombre, email, comentario FROM LibroVisitas";
          $res = pg_query($id_conexion, $query);

          if(!$res)
          {
               die('Es imposible realizar la consulta: ' . pg_last_error());
          }

          //Procesamos los resultados
          print <<<_HTML_
     <table border="1">
     <tr>
     <th>Nombre</th>
     <th>Email</th>
     <th>Comentario</th>
     </tr>
_HTML_;

          while($row = pg_fetch_array($res))
          {
               print <<<_HTML_
               <tr>
               <td>$row[nombre]</td>
               <td>$row[email]</td>
               <td>$row[comentario]</td>
               </tr>
_HTML_;
          }
          print '</table>';

          //Cerramos la conexion
          pg_close($id_conexion);
?>
</body>
</html>
```

Salida

5.6.3 Eliminar registros

Prácticamente sólo debemos modificar un par de líneas en el código para que nuestra aplicación interactúe con PostgreSQL. El script para eliminar registros tiene algunas modificaciones; al igual que el resto de los script, las funciones `mysql_xxx` son sustituidas por sus equivalentes `pg_xxx`.

eliminar.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title>Eliminar Visitas</title>
</head>
<body>
<center><h1>Eliminar alguna visita</h1></center>
<hr />
<?php
      $db_user = 'postgres';
      $db_pass = 'postgres';

      //Realizamos la conexion
      $id_conexion = pg_connect("dbname=MiTienda user=$db_user
password=$db_pass");
      if(!$id_conexion)
      {
            die('Es imposible conectarse al servidor: ' . mysql_error());
      }

      //  --------- Eliminar registros -----------------------
      if(isset($_POST['submit_ok']))
      {
            foreach($_POST['visita'] as $id_comentario)
            {
```

```
                $query = "DELETE FROM LibroVisitas WHERE id_comentario =
'$id_comentario'";
                $del_status = pg_query($id_conexion, $query);
                if(!$del_status)
                {
                     die('Es imposible eliminar el registro: ' .
pg_last_error());
                }
           }
     }


     // --------- Mostramos los registros ------------------

     //Realizamos la consulta
     $query = "SELECT id_comentario, nombre, email, comentario FROM
LibroVisitas";
     $res = pg_query($id_conexion, $query);

     if(!$res)
     {
          die('Es imposible realizar la consulta: ' . pg_last_error());
     }

     //Procesamos los resultados
     print <<<_HTML_
     <form method="post" action="$_SERVER[PHP_SELF]">
     <table border="1">
     <tr>
     <th>Seleccionar</th>
     <th>Nombre</th>
     <th>Email</th>
     <th>Comentario</th>
     </tr>
_HTML_;

     while($row = pg_fetch_array($res))
     {
          print <<<_HTML_
          <tr>
          <td><input type="checkbox" id="visita[]" name="visita[]"
value="$row[id_comentario]" /></td>
          <td>$row[nombre]</td>
          <td>$row[email]</td>
          <td>$row[comentario]</td>
          </tr>
_HTML_;
     }
          print '</table>';
          print '<input type="hidden" id="submit_ok" name="submit_ok"
value="true" />';
          print '<input type="submit" value="Eliminar" />';
          print '<input type="reset" value="Reset" /></from>';
```

```
        //Cerramos la conexion
        pg_close($id_conexion);

?>
</body>
<html>
```

Salida

5.6.4 Actualizar registros

El script para actualizar registros tiene cambios casi imperceptibles pero muy significativos.

actualizar.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title>Actualizar Libro de Visitas</title>
</head>
<body>
<center><h1>Actualizar Libro de Visita</h1></center>
<hr />
<?php
      $db_user = 'postgres';
      $db_pass = 'postgres';

      //Realizamos la conexion
      $id_conexion = pg_connect("dbname=MiTienda user=$db_user
password=$db_pass");
      if(!$id_conexion)
      {
            die('Es imposible conectarse al servidor: ' . pg_last_error());
      }

      // ------------ Realizamos la Actualizacion --------------
```

```
	if(isset($_POST['update_ok']))
	{
		//Obtenemos la informacion del formulario
		//filtrando los caracteres especiales
		$nombre = htmlentities($_POST['nombre']);
		$email = htmlentities($_POST['email']);
		$comentario = htmlentities($_POST['comentario']);
		$id_comentario = $_POST['id_comentario'];

		$query = "UPDATE LibroVisitas SET nombre = '$nombre', email =
'$email', comentario = '$comentario'";
		$query .= " WHERE id_comentario = $id_comentario";

		$upd_status = pg_query($id_conexion, $query);

		if(!$upd_status)
		{
			die('Es imposible actualizar el registro: ' . pg_last_error());
		}

		print "<b>Registro Actualizado</b><br />";
		print '<a href="actualizar.php">Regresar</a>';
	}

	// ------------- Mostramos el formulario de actualizacion ------------
	else if(isset($_POST['submit_ok']))
	{
		//Realizamos la consulta
		$id_comentario = $_POST['visita'];
		$query = "SELECT nombre, email, comentario FROM LibroVisitas WHERE
id_comentario = $id_comentario";
		$res = pg_query($id_conexion, $query);

		if(!$res)
		{
			die('Es imposible realizar la consulta: ' . pg_last_error());
		}

		//Procesamos los resultados
		print <<<_HTML_
		<form method="post" action="$_SERVER[PHP_SELF]">

_HTML_;

		while($row = pg_fetch_array($res))
		{
			print <<<_HTML_
			Nombre:	<input	type="text"	id="nombre"	name="nombre"
value="$row[nombre]"/>
			<br />
			e-mail:	<input	type="text"	id="email"	name="email"
value="$row[email]" />
			<br />
			Comentarios:	<textarea	id="comentario"	name="comentario"
```

```
rows="5" cols="30">$row[comentario]</textarea>
                <br />
_HTML_;
           }

           print <<<_HTML_
           <input type="hidden" id="update_ok" name="update_ok" value="true" />
           <input    type="hidden"    id="id_comentario"    name="id_comentario"
value="$id_comentario" />
           <input type="submit" value="Actualizar" />
           <input type="reset" value="Reset" />
           </from>
_HTML_;

      }
      else
      {

           print "<b>No hay registro que actualizar</b>";
      }

           //Cerramos la conexion
           pg_close($id_conexion);
?>
</body>
</html>
```

Resumen

La popularidad de MySQL se debe sobre todo a que es un DBMS muy ligero y de fácil administración, sin embrago, en el mundo de las aplicaciones de alta disponibilidad, concurrencia de usuarios, transacciones y seguridad PostgreSQL es el manejador de bases de datos más indicado.
PostgreSQL a madurado lo suficiente y es una fuerte competencia de los DBMS comerciales, como Oracle por ejemplo.

Una de las principales características que hacen atractivo el uso de PostgreSQL son los llamados *Stored Procedures,* que actualmente están siendo soportados en MySQL 5.0 pero con algunas restricciones.

La administración de PostgreSQL mediante el shell no es tan sencilla como ocurre con MySQL. No obstante existen herramientas gráficas, como es el caso de phpPgAdmin o pgAdmin, que nos permiten realizar la administración de manera rápida y amigable.

El API con el que PHP interactua con PostgreSQL es similar al de MySQL, de modo que no hay que memorizar nuevas funciones, basta con recordar las de MySQL y adaptarlas a las de PostgreSQL.

Capítulo 6.

Introducción a los CGIs con Perl

Pero ¿por qué conformarse con el seudocódigo cuando puedes hacerlo de verdad? Lo que sigue es Pontifex escrito como código Perl.

Stephenson, Neal. Criptonomicón: El código Pontifex

6.1 Introducción a Perl

El nacimiento de Perl[1] se remonta a 1987 cuando su creador, Larry Wall, anuncio el lanzamiento de la versión 1.0 el 18 de diciembre en el grupo de noticias `comp.sources.misc`. Con los años fue creciendo el desarrollo de Perl y en 1995 se lanzo Perl 5 con una completa reescritura del intérprete, nuevas características y disponible en varias plataformas.

Las características de Perl son:

- Es un lenguaje interpretado
- Su sintaxis es similar a la de C
- Es orientado a objetos y también estructurado
- Independiente a la plataforma
- Optimizado para la manipulación de cadenas
- Es una alternativa para generar páginas Web dinámicas
- Cada sentencia se finaliza con el `;`
- Los comentarios se indican con el símbolo `#`

Desventajas de Perl:

- Al tratarse de un lenguaje interpretado, su desempeño puede ser lento.
- La sintaxis avanzada de Perl suele ser confusa, sobre todo cuando se trabajan expresiones regulares.

Perl viene instalado en casi todos los sistemas GNU/Linux, basta teclear el comando `whereis perl` para descubrir la ruta del interprete. ActivePerl es una distribución de los binarios de Perl y esta disponible para distintas plataformas como Mac OS X, Windows, Solaris, AIX HP-UX y GNU/Linux.

6.1.1 Variables y tipos de datos

Existen cuatro tipos de variables según su alcance:

- Variables locales
- Variables de bloque
- Variables globales
- Variables predefinidas

Una variable no puede tener más de un alcance, pero siempre tendrá uno.
Para indicar una variable debe comenzar con algunos de los siguientes identificadores:

Tipo de dato	Identificador
Escalar	`$`

1 *Practical Extraction and Report Language* (Lenguaje Práctico para la Extracción e Informe)

Tipo de dato	Identificador
Vectorial o array	@
Arreglo asociativo o hash	%
Subrutina	&

El nombre de la variable puede contener caracteres, números o el carácter _.
Las variables son sensibles a mayúsculas y minúsculas.

Los tipos **escalares** son números enteros, numero flotantes y cadenas de caracteres.

escalares.pl

```
#!/usr/bin/perl
$edad = 24; #entero
$salario = 12500.80; #flotante
$nombre = 'Antonio'; #cadena
$apPat = "González"; #cadena
print "$nombre $apPat tiene $edad años y su salario es de $salario";
```

La entrada de datos desde el teclado se realiza con la expresión `<STDIN>`.

entrada.pl

```
#!/usr/bin/perl

print 'Escribe tu nombre: ';
$nombre = <STDIN>;
chomp($nombre); # elimina el salto de línea
print "Buen día $nombre";
```

Una variable de tipo **array** contiene una colección de elementos que se identifican mediante un indice entero. Existen varias funciones de Perl que manejan arreglos:

- `push`, Agrega un elemento al final del arreglo.
- `pop`, Remueve y devuelve el elemento al final del arreglo.
- `shift`, Remueve y devuelve el primer elemento del arreglo.
- `unshift`, Agrega un elemento al inicio del arreglo.

array_01.pl

```
#!/usr/bin/perl

@colores = ('rojo', 'azul', 'verde', 'amarillo');
print "Colores disponibles\n @colores \n";
print 'Agrega un nuevo color: ';
$color = <STDIN>;
chop($color);
push(@colores, $color);
print "Colores disponibles\n @colores \n";
print "El 3r color es: $colores[2]";
```

Las variables de tipo **hash** o arreglo asociativo, contiene elementos que son asociados a una clave. Las funciones comúnmente usadas son:

- `delete $hash{$key}`, Elimina el par llave/valor indicado, devuelve el valor
- `exist $hash{$key}`, Verifica si la llave existe, devuelve `true` si la llave existe.
- `keys $hash`, Devuelve un arreglo con las llaves de cada valor.
- `values $hash`, Devuelve un arreglo con los valores.

hash_01.pl

```
#!/usr/bin/perl

%frutas = ('MNZ', 'manzana',
        'PRA', 'pera',
        'FRS', 'fresa',
           'MGO', 'mango');

@llaves = keys(%frutas);
@valores = values(%frutas);
print "Llaves: @llaves \n";
print "Valores: @valores \n";
```

6.1.2 Operadores

Los operadores **aritméticos** son:

Operador	Operación
+	Suma
-	Resta
/	División
*	Multiplicación
%	Modulo
**	Exponencial
<	Menor que
>	Mayor que
<=	Menor igual que
>=	Mayor igual que
==	Igual que
!=	Distinto de
++	Incrementa una unidad
--	Decrementa una unidad

Cuando se realizan operaciones con numero enteros el resultado será un numero entero, cuando uno de los operados es un numero flotante el resultado será un numero flotante.

Perl ofrece un extenso juego de operadores para las **cadenas**:

Operador	Descripción	Ejemplo	Salida
.	Concatenación	'Hola'.'humano'	'Holahumano'
x	Replicación	'a'x4	'aaaa'
gt	Compara si el valor de la izquierda es alfabéticamente mayor que el de la derecha.	'cat' gt 'rat'	false
lt	Compara si el valor de la izquierda es alfabéticamente menor que el de la derecha.	'cat' lt 'rat'	true
ge	Compara si el valor de la izquierda es alfabéticamente mayor o igual que el de la derecha.	'cat' ge 'cat'	true
le	Compara si el valor de la izquierda es alfabéticamente menor o igual que el de la derecha.	'cat' le 'rat'	false
eq	Compara si el valor de la izquierda es alfabéticamente igual al de la derecha.	'cat' eq 'cat1'	false
ne	Compara si el valor de la izquierda es completamente diferente al de la derecha.	'cat' ne 'cat1'	true

Los operadores **lógicos** &&(AND), ||(OR) y !(NOT) también están disponibles.

6.1.3 Estructuras de control

La estructura de control if-else es similar a la del lenguaje PHP, mientras que en la estructura if-elseif-else cambia en Perl por if-elsif-else.

Perl introduce la estructura unless, similar a un while, que ejecuta las sentencias de su cuerpo hasta que la expresión que evalúa sea verdadera. Perl carece de la estructura switch.

La sintaxis de los ciclos while y for se mantiene igual que en PHP, sin embargo el ciclo foreach cambia en su sintaxis:

Sintaxis	Ejemplo

foreach $valor (@array)
{
  sentencias
}
	@empleados = ('hugo', 'paco', 'luis');
	foreach $empleado (@empleados)
	{
	  print "$empleado";
	}

	%frutas = ('MNZ', 'manzana', 'PRA', 'pera', 'FRS', 'fresa',

Sintaxis	Ejemplo

	'MGO', 'mango');
	foreach $llave (keys(%frutas))
	{
	print "$llave es $frutas{$llave} \n";
	}

6.2 CGI

Conocida como Interfaz Común de Pasarela[2], provee un mecanismo mediante el cual un servidor web y un programa puede atender peticiones de la Internet. En éste caso la respuesta del programa es enviada en HTML al cliente.

En el paradigma CGI un programa recibe una serie de peticiones de un cliente (browser), las procesa y devuelve el resultado nuevamente al cliente.
Algunas de las características de CGI son:

- Naturaleza no interactiva. El programa CGI inicia su ejecución al recibir una petición, envía los resultados y finaliza. No permanece activo en espera de más peticiones.
- Interacción de usuario segmentada. Se refiere a guardar el estado de la información, la forma de recordar que sucedió un par de instantes atrás es utilizar Cookies.

Las entradas de un CGI son:

- El método GET. Los datos de un formulario se envían a través de la variable de entorno `QUERY_STRING`, que es una cadena de caracteres variable desde la barra de navegación del browser.
- El método POST. Los datos de un formulario se envían a través de la variable de entorno `CONTENT_LENGTH`, que es una cadena invisible para el usuario.
- Variables de Entorno. Son variables ocultas que el servidor web envía a cada programa CGI.

La salida de un CGI son:

- Primeramente, un CGI devuelve una cabecera con la cadena '`Content-type: text/html`', seguida de una líneas en blanco.
- Finalmente toda la información procesada, que puede ser HTML, imágenes, sonidos o texto.

6.3 `'Hola mundo'` con CGI

Como se ya se menciono, la cabecera que devuelve un script CGI contiene la cadena `Content-type: text/html` seguida de una línea en blanco y el resto del texto.

Crear programas CGI con Perl es sumamente sencillo, basta con colocarlos en la carpeta `cgi-bin` del servidor web e invocarlos desde un navegador web.

Vamos a ejecutar el primer scripts sobre el servidor web Apache; es importante que la primer línea del script indique la ruta hacia el interprete de Perl.

2 Common Gateway Interface

Apache sobre Windows	Apache sobre GNU/Linux

holamundo.pl
`#!C:\ActivePerl\perl\bin\perl.exe`
`print "Content-type: text/html\n\n";`
`print 'Hola mundo!';`
	holamundo.pl
	`#!/usr/bin/perl/`
	`print "Content-type: text/html\n\n";`
	`print 'Hola mundo!';`

Salida 	**Salida** 

En la mayoría de los casos devolveremos código HTML:

Apache sobre Windows	Apache sobre GNU/Linux

holamundo2.pl
`#!C:\ActivePerl\perl\bin\perl.exe`
`print "Content-type: text/html\n\n";`
`print "<html><head>\n";`
`print "<title>Hola mundo!'</title></head>\n";`
`print "<body><h1>Hola mundo!</h1>";`
`print "</body></html>";`
	holamundo2.pl
	`#!/usr/bin/perl`
	`print "Content-type: text/html\n\n";`
	`print "<html><head>\n";`
	`print "<title>Hola mundo!'</title></head>\n";`
	`print "<body><h1>Hola mundo!</h1>";`
	`print "</body></html>";`

Salida 	**Salida** 

Las variables de entorno donde se ejecuta el script, se almacenan en un arreglo asociativo llamado `%ENV`:

Apache sobre GNU/Linux

var_entorno.pl
`#!/usr/bin/perl`
`print "Content-type: text/html\n\n";`
`foreach $key (keys %ENV)`
`{`

```
     print "$key --> $ENV{$key}<br>";
}
```

Salida

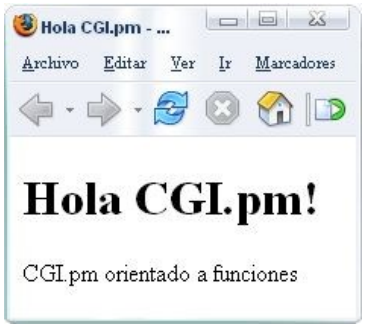
```
SCRIPT_NAME --> /cgi-bin/var_entorno.pl
SERVER_NAME --> localhost
SERVER_ADMIN --> you@example.com
HTTP_ACCEPT_ENCODING --> gzip,deflate
HTTP_CONNECTION --> keep-alive
REQUEST_METHOD --> GET
HTTP_ACCEPT -->
text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8
,image/png,*/*;q=0.5
SCRIPT_FILENAME --> /usr/local/apache2/cgi-bin/var_entorno.pl
SERVER_SOFTWARE --> Apache/2.0.54 (Unix) PHP/4.4.0
HTTP_ACCEPT_CHARSET --> ISO-8859-1,utf-8;q=0.7,*;q=0.7
QUERY_STRING -->
REMOTE_PORT --> 32768
HTTP_USER_AGENT --> Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.7.10)
Gecko/20050716 Firefox/1.0.6
SERVER_PORT --> 80
SERVER_SIGNATURE -->
Apache/2.0.54 (Unix) PHP/4.4.0 Server at localhost Port 80

HTTP_ACCEPT_LANGUAGE --> en-us,en;q=0.5
REMOTE_ADDR --> 127.0.0.1
HTTP_KEEP_ALIVE --> 300
SERVER_PROTOCOL --> HTTP/1.1
PATH -->
/usr/local/sbin:/usr/sbin:/sbin:/usr/local/bin:/usr/bin:/bin:/usr/X11R6/bin:/us
r/games:/opt/www/htdig/bin:/usr/lib/java/bin:/usr/lib/java/jre/bin:/opt/kde/bin
:/usr/lib/qt/bin:/usr/share/texmf/bin
REQUEST_URI --> /cgi-bin/var_entorno.pl
GATEWAY_INTERFACE --> CGI/1.1
SERVER_ADDR --> 127.0.0.1
DOCUMENT_ROOT --> /usr/local/apache2/htdocs
HTTP_HOST --> localhost
```

6.4 El modulo CGI.pm

Éste modulo de Perl 5 es usado para crear y analizar contenido HTML, así como formularios y procesar las peticiones del cliente. Puede ser empleado de dos modos: orientado a funciones (*function-oriented)* u orientado a objetos(*object-oriented*). Vamos a trabajar ambos modos para notar la diferencia en la sintaxis:

function-oriented	object-oriented

`#!C:\ActivePerl\perl\bin\perl.exe`
`use CGI qw/:standard/;`
`print header(),`
`  start_html(-title=>'Hola CGI.pm'),`
	`#!C:\ActivePerl\perl\bin\perl.exe`
	`use CGI;`
	`$q = new CGI;`
	`print $q->header(),`
	`  $q->start_html(-title=>'Hola`

function-oriented	object-oriented
h1('Hola CGI.pm!'), 'CGI.pm orientado a funciones', end_html();	CGI.pm'), $q->h1('Hola CGI.pm!'), 'CGI.pm orientado a objetos', $q->end_html();
Salida  	**Salida** 

En el caso de script orientado a funciones, la primera línea carga, mediante el operador `use`, la definición de CGI.pm e importa las funciones `standard`. El script orientado a objetos se comunica directamente con el modulo a través de un objeto CGI. Éste último modo tiene la ventaja de consumir menos memoria que el estilo orientado a funciones y poder guardar, si se requiere, el estado del objeto en alguna base de datos o sistema de almacenamiento.

6.5 Procesar Formularios con Perl

Hemos mencionado los métodos con los que se pueden envían los datos de un formulario al servidor web. CGI.pm proporciona una forma sencilla de obtener los datos del cliente a través del método `param()`, enviados tanto por POST como por GET, así como varios métodos para crear formularios y elementos HTML.

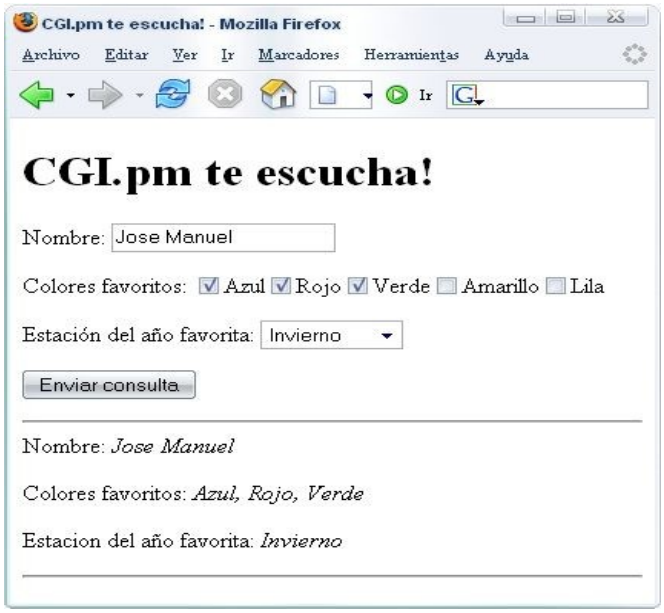
Vamos a mostrar el uso de los métodos para crear un formulario y procesarlo de manera dinámica.

cgiform.pl

```
#!C:\ActivePerl\perl\bin\perl.exe
use CGI qw/:standard/;
print header;
print start_html('CGI.pm te escucha!'),
      h1('CGI.pm te escucha!');
#Imprimimos el formulario
print start_form,
      "Nombre: ", textfield('nombre'),
      p,
      "Colores favoritos: ",
       checkbox_group(-name=>'colores',
        -values=>['Azul', 'Rojo', 'Verde', 'Amarillo', 'Lila'],
        -defaults=>['Verde']),
      p,
      "Estación del año favorita: ",
      popup_menu(-name=>'estacion',
      -values=>['Primavera', 'Verano', 'Otoño' , 'Invierno']),
      p,
      submit,
```

```
      end_form,
      hr;

#Procesamos el formulario
if(param)
{
     print "Nombre: ", em(param('nombre')),
     p,
     "Colores favoritos: ", em(join(", ",param('colores'))),
     p,
     "Estacion del año favorita: ", em(param('estacion')),
     hr;
}
print end_html;
```

Salida



Resumen

Las variables en Perl son identificadas con un signo precedente, denominado *sigil*, de modo que para cada tipo de dato corresponde un *sigil* diferente. Esto hace suponer a los programadores que se trata de un lenguaje con una sintaxis muy compleja, pero es cuestión de acostumbrarse y explotar las bondades que ofrece Perl.

La popularidad de Perl se debe a que fue uno de los primeros lenguajes utilizados en la programación de CGIs, ha facilitado la administración de servidores, ha sido muy útil para generar reportes de bitácoras (*log files*), manipula eficientemente grandes textos y porque actualmente es un lenguaje muy utilizado en la bioinformática.

Con Perl se puede trabajar de manera estructurada o bien orientada a objetos.

Perl trabaja como ningún otro lenguaje las expresiones regulares, pues tiene una sintaxis especializada para escribir expresiones regulares, sin embrago, puede resultar complejo entenderlas cuando no se tiene experiencia.

Capítulo 7.

Algunas diferencias entre Java y PHP

"Write once, run anywhere."

Sun Microsystems

7.1 El origen de Java

Java nació en 1991 como parte del proyecto *Green Project* de Sun Microsystems, del que formaban parte James Gosling, Patrick Naughton y Mike Sheridan, con la misión de desarrollar una tecnología para programar dispositivos inteligentes. Al inicio el equipo comenzó a trabajar con C++, pero debido a los inconvenientes de potabilidad a otras plataformas, la falta de un recolector de basura y el manejo de memoria por parte del programador descartaron el uso de C++. Gosling intentó modificar y ampliar C++, sin embargo descarto la idea para crear un lenguaje completamente nuevo, al que llamó Oak, en honor al roble que tenía junto a su oficina.
En el verano de 1992 tenían listo un sistema operativo y algunas librerías escritas en Oak, además de una PDA (Personal Digital Assistant o Asistente Digital Personal), denominada Star7, que mostraba el potencial del lenguaje.

En junio de 1994 el equipo reoriento la tecnología hacia la Web, debido al éxito que el navegador Mosaic estaba teniendo por ser una forma interactiva de comunicación. El nombre del lenguaje fue cambiado a Java, debido a que ya existía una marca de tarjetas gráficas con el nombre de Oak y se desarrollo un navegador web al que bautizaron como HotJava.

El 23 de mayo de 1995, durante la conferencia anual de Sun Microsystems fue presentado a la luz publica Java y el navegador HotJava, también Netscape anuncio que soportaría los applets escritos en Java.

En enero de 1996 Sun Microsystems fundo JavaSoft para continuar con el desarrollo y expansión de Java.

7.1.2 Principales características de Java

Independencia de la plataforma. Es la capacidad que tiene una aplicación Java de ejecutarse en cualquier sistema operativo y cualquier arquitectura de hardware sin que esté obligado el programador de compilar para cada plataforma el mismo programa. Los programas escritos en Java se compilan en un código intermedio denominado *Byte codes*, que son interpretados por una maquina virtual (JVM) especifica para cada plataforma.

Orientado a objetos. Como muchos otros lenguajes de programación modernos, Java está orientado a objetos; sin embargo guarda una diferencia importante, Java no soporta la herencia múltiple y a sustituido éste concepto por uno más sencillo: las interfaces.

Distribuido. El lenguaje facilita la comunicación de los programas a través de redes, la invocación de métodos remotos y el uso de objetos distribuidos.

Seguridad. Java se diseño para hacer imposible ataques muy conocidos en programación: desbordamiento de la pila de ejecución, corromper la memoria fuera del espacio de proceso, leer o escribir archivos son permiso. Java elimino el uso explicito de punteros, de modo que el programador no tiene que preocuparse por la administración de la memoria.

Independiente a la plataforma. Los programas en Java se compilan en un código intermedio, *bytecode*, que es interpretado por una maquina virtual especifica para cada plataforma. Así un programa puede ser ejecutado en Windows o en GNU/Linux sin que requiere una modificación o una compilación especial.

Multihilo. Java aprovecha las capacidades del microprocesador para trabajar con multiprocesos, sin embrago delega esta tarea al sistema operativo donde se esté ejecutando la maquina virtual.

Interpretado. Los lenguajes interpretados tiene un menor desempeño que los lenguajes compilados. Java ofrece una herramienta para compilar bajo una plataforma especifica, y aun que esto limita la potabilidad e independencia mejora el desempeño.

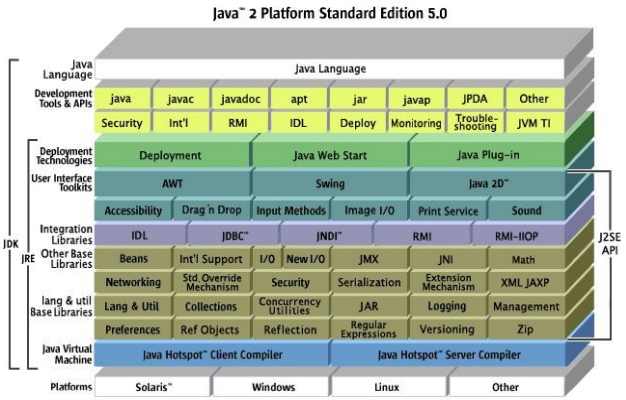
7.1.3 Las APIs de Java

Sun Microsystems tomo en cuenta las distintas necesidades de los usuarios y dividió la distribución de Java en tres versiones: J2SE (*Java 2 Standard Edition*), J2EE (*Java 2 Enterprise Edition*) y J2ME (*Java 2 Micro Edition*).

J2SE. Esta versión contiene el conjunto esencial de herramientas orientada al desarrollo de Applets y aplicaciones con interfaz gráfica, multimedia, comunicación a través de una red, etc..

J2EE. Esta distribución está orientada al entorno empresarial, en donde las aplicaciones se encuentra distribuidas en una red, compartiendo métodos y objetos remotos mediante EJBs (*Enterprise Java Beans*), los servicios web, servicios de nombres, comunicación mediante XML, etc..

J2ME. Con esta distribución se pretende cubrir el desarrollo de aplicaciones para dispositivos con pocos recursos computacionales, como son teléfonos celulares, PDAs y electrodomésticos inteligentes.



Podemos decir que J2EE es un superconjunto de J2SE, mientras que J2ME es un subconjunto de J2SE.

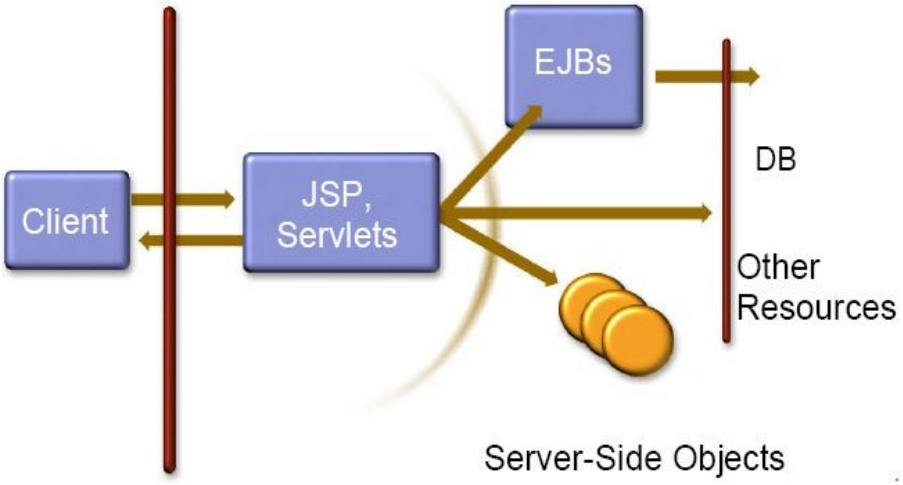
7.1.4 Servlets y JSP

Los servlets son aplicaciones escritas en Java para la comunicación a través de CGI. Se ejecutan dentro de un ambiente espacial denominado contenedor web (*web container*) que sirve como puente entre el servidor web y el servlet. Cuando un cliente le hace una petición HTTP al servidor web, esté la direcciona al contenedor web y es atendida por un servlet.

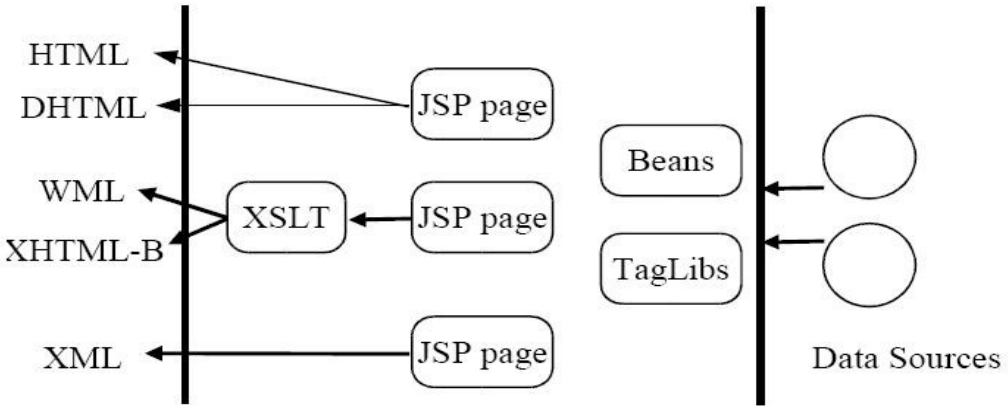
Los servlets son más eficientes que los programas CGI tradicionales debido a que se carga una copia en

memoria del servlet y se ejecutan `n` subprocesos de esté. Son altamente configurables y transportables ya que mediante un archivo XML (*deployment descriptor*) informan al contenedor las características del servlet, de esta forma pueden se ejecutado tanto en contenedores web comerciales como libres.
La tecnología JSP (*JavaServer Pages*) permite embeber código Java en páginas HTML, generando contenido dinámico. Es mucho más fácil generar páginas dinámicas con JSP que con servlets, sin embargo, internamente el *web container* transforma los JSP en servlets. El código Java se embebe entre las etiquetas `<% %>` o con su equivalente XML `<jsp:xxx />`.

Las JSP y los servlets son tecnologías que se complementan y aunadas con los EJB logran construir aplicaciones web robustas bajo el patrón de diseño MVC.



Java fue uno de los primeros lenguajes en impulsar el uso de XML como forma de comunicación entre aplicaciones, el uso de XML se ve reflejado en la mayor parte de las soluciones que Java ha ofrecido. El uso de las JSP y XML vuelven tan flexible a una aplicación que puede servir a múltiples clientes, como son teléfonos móviles o PDAs, a través de los lenguajes de marcado que estos suelen entender.



7.1.5 Elementos básicos

Vamos a trabajar con Tomcat 5.5[1], un servidor web de código abierto escrito completamente en Java y con soporte para servlets y JSPs, desarrollado y mantenido por la *Apache Software Foundation* y colaboradores de todo el mundo.

Los cuatro tipos de elementos básicos definidos en la sintaxis de un JSP son:

- ***Template content***. Todo aquello que no es propiamente una etiqueta de JSP, como el contenido estático, se le denomina *Template content*.
- **Directivas**. Son instrucciones que le indican al motor de JSP como lucirá el JSP después de ser construido, por ejemplo, un instrucción que incluye un archivo externo.
- **Elementos de script**. Se trata de declaraciones, expresiones y scriptles (código java embebido).
- **Acciones**. Proveen de funcionalidades de alto nivel a las JSP en forma de etiquetas XML personalizadas.

Existen una serie de objetos implícitos que le proporcionan a la JSP acceso a las variables de entorno, como son:

Objeto implícito	Descripción
`output`	Flujo de salida hacia el cliente.
`request`	Objeto que encapsula la petición del cliente al JSP.
`response`	Encapsula la respuesta del JSP a la petición del cliente.
`session`	Representa una sesión del cliente.
`pageContext`	Punto único de acceso a los atributos de una página.
`config`	Permite la inicialización de los parámetros iniciales.
`page`	Objeto generado durante la traslación de JSP a servlet.
`application`	Proporciona un ServletContext para el manejo de datos persistentes.

El código Java embebido, o también llamado scriptlet, se coloca dentro de las etiquetas `<%` y `%>`, sin embrago también está disponible el etiquetado estilo XML: `<jsp:declaracion> </jsp:declaracion>`. Hay que aclarar que no todas las herramientas de desarrollo soportan en su totalidad esta ultima forma de etiquetado.

La sintaxis de las directivas JSP luce así:

```
<%@ directiva atributo=valor %>
```

o bien,

```
<jsp:directiva.directiva [...] />
```

Las tres principales directivas son las siguientes:

Directiva	Propósito
`page`	Controla las propiedades de la JSP
`include`	Incluye el contenido de un archivo

1 http://tomcat.apache.org/

Directiva	Propósito
`taglib`	Indica el uso de una librería personalizada.

Las páginas JSP se guardan con la extensión `.jsp`, en el caso de Tomcat se colocan dentro del directorio `webapps`.

hola.jsp

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
      <title>Hola JSP</title>
</head>
      <%@ page import="java.util.*" %>
<body>
      <h3>
      <center>
      Hola, &eacute;ste es mi numero aleatorio:<br />
      <%= (int) (Math.random()*1000) %>
      </h3>
      </center>
</body>
</html>
```

Salida

7.2 JSP: ventajas y desventajas

La documentación oficial que Sun Microsystems proporciona sobre JSPs menciona algunas ventajas que tienen las JSP sobre otras tecnologías web, sin embargo el uso de PHP, JSP o cualquier otra tecnología web dependerá de las necesidades de cada usuario y los recursos con los que se cuenten.

	CGI/Perl	Mod_perl	ASP	PHP	JSP
Portabilidad hacia otros Servidores Web	Si, con algunas limitaciones	No, Depende de Apache	No, Microsoft IIS o Personal Web Server	Si, con algunas limitaciones	Si, sin problemas
Portabilidad a través de otras arquitecturas de hardware o SO	No	No	No	No	Si
Reusabilidad de código	No	No	No	No	Si
Lenguaje	C/Perl	Perl	VBScript,	PHP	Java

	CGI/Perl	Mod_perl	ASP	PHP	JSP
			JScript		
Protección de memoria	Si	No	No	Si	Si
Acceso sin la creación de un proceso independiente	No	Si	Si	Si	Si
Separación de la lógica del negocio y la presentación.	Parcial	Parcial	Parcial	Parcial	Completamente
Soporte por parte de herramientas de desarrollo comerciales o industriales.	No	No	No	No	Si
Extensible a través de componentes de comerciales.	No	No	No	No	Si

Resumen

Java a recibido a lo largo de una década el apoyo de cientos de grandes empresas del mundo del software y hardware, por tratarse de un lenguaje independiente a la plataforma se ha convertido un estándar para el desarrollo de aplicaciones empresariales.

Java es un lenguaje totalmente orientado a objetos, compilado en un código intermedio e interpretado por una maquina virtual. Existen 3 distribuciones de Java: J2SE para el desarrollo de aplicaciones de escritorio, J2EE para desarrollar aplicaciones empresariales y J2ME para desarrollar aplicaciones para dispositivos móviles o limitados de recursos computacionales.

En respuesta a la demanda de una API de Java para desarrollar aplicaciones CGIs se desarrollaron los Servlets y las JSPs. Con las JSPs se puede mezclar HTML estático con contenido dinámico generado por Servlets.

Decidir entre Java o PHP no es una tarea sencilla, todo dependerá de los requerimientos del cliente, el diseño del software, el tipo y la magnitud del problema, los recursos económicos y humanos con los que se dispone, etcétera.

Conclusiones.

En el área de la computación e informática, la actualización profesional siempre ha sido una necesidad, pues es un área que cambia con mucha rapidez. El terminar la carrera de Ingeniería en Computación no garantiza estar al día en lo que a tecnologías de software se refiere.

Haber tomado el diplomado de "Desarrollo e Implementación de Sistemas con Software Libre en Linux" fue una oportunidad para actualizar y reforzar mis conocimientos sobre desarrollo de aplicaciones Web. En algún momento de la carrera de ingeniería, de manera autodidacta, me dedique un tiempo a estudiar las tecnologías para el desarrollo Web, sin embargo la retroalimentación que se obtiene en un diplomado con los alumnos y los instructores no se puede adquirir en un libro.

El software libre está ganando aceptación no solo en el ámbito académico, ahora en el mundo de los negocios, empresas como IBM lo han adoptado y construyen herramientas comerciables, por ejemplo WebSphere Application Server fue construido utilizando Apache Web Server. GNU/Linux es otro caso de éxito comercial, la empresa Novell compro todos los derechos sobre la distribución SuSE.

Las empresas y los profesionistas deben estar preparados para manejar y desarrollar software libre, ya que los estándares abiertos de la industria del software lo demandan.

Las aplicaciones empresariales que no fueron construidas pensando en que hoy se conoce como SOA (Service-oriented architecture) están siendo integradas a Servicios, ésta es una oportunidad para los desarrolladores Web. Por otra parte, cada vez más empresas quieren tener un sitio Web y poder brindar sus servicios a sus clientes mediante Internet, esto demanda una vez mas de los desarrolladores Web.

Sería muy recomendable que el diplomado de "Desarrollo e Implementación de Sistemas con Software Libre en Linux" ampliara sus módulos y agregara temas como: XML, JavaScript y Ajax, pues son tecnologías que el mercado laboral actual está demandando junto con PHP.

Bibliografía.

J. Tackett, S. Burnett, *"Edición Especial Linux"*, Prentice Hall, 2000.

J. Turnbull, *"Hardening Linux"*, Apress, 2005.

B. McCarty, *"Learning red hat Linux"*, O'Reilly, 2004.

R. L. Ziegler, *"Guía avanzada Firewalls Linux"*, Pearson Educación, 2000.

H. Esser, T. Forster, *"Linux : big pack"*, Marcombo, 1999.

J. P. Hekman, *"Linux : in a nutshell"*, O'Reilly, 1997

L. Lemay, R. Cadenhead, *"Aprendiendo Java 2 en 21 Días"*, Prentice Hall, 1999.

C. S. Horstmann, G. Cornell, *"Core Java 2, Volumen I - Fundamentos"*, Prentice Hall, 2003.

C. S. Horstmann, G. Cornell, *"Core Java 2, Volumen II – Características Avanzadas"*, Prentice Hall, 2003.

B. Eckel, *"Piensa en Java"*, Prentice Hall, 2002.

M. Hall, *"Servlets y JavaServer Pages, Guía practica"*, Prentice Hall, 2001.

B. McLaughlin, *"Java y XML"*, Anaya Multimedia – O'Reilly, 2001.

J. D. Zawodny, D. J. Balling, *"MySQL Avanzado"*, Anaya Multimedia – O'Reilly, 2004.

M. Maslakowski, T. Butcher, *"Aprendiendo MySQL en 21 Días"*, Pearson Educación, 2001.

H. E. Williams, D. Lane, *"Web database applications with PHP and MySQL"*, O'Reilly, 2004.

R. Sheldon, *"SQL : a beginner's guide"*, McGraw-Hill, 2003.

D. Sklar, *"Introducción a PHP 5"*, Anaya Multimedia – O'Reilly, 2005.

H. E. Williams, D. Lane, *"Web database applications with PHP and MySQL"*, O'Reilly, 2004.

L. Ullman, *"PHP and MySQL for dynamic Web sites"*, Peachpit, 2003.

P. P. Fabrega, *"PHP 4"*, Prentice Hall, 2000.

J. L. Lopez Quijado, *"Domine HTML y DHTML"*, Ra-Ma, 2003.

C. Musciano, B. Kennedy, *"HTML, the definitive guide"*, O'Reilly, 1997.

N. Patwardhan, E. Siever, S. Spainhour, *"Perl in a nutshell"*, O'Reilly, 2002.

J. P. Bañeres, "Perl : paginas Web interactivas", Alfaomega, 1999.

E. Herrmann, *"Teach yourself CGI programming with Perl 5 in a week"*, Sams, 1997.

S. Gundavaram, *"CGI programming : on the World Wide Web"*, O'Reilly, 1996.

D. Medinets, *"Perl 5 a través de ejemplos"*, Prentice-Hall Hispanoamericana, 1997.