# UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

**FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN**

## INTEGRACIÓN DE HERRAMIENTAS LIBRES PARA EL DESARROLLO DE UN SISTEMA DE ADMINISTRACIÓN DE CITAS

T R A B A J O  E S C R I T O

EN LA MODALIDAD DE SEMINARIOS Y CURSOS DE ACTUALIZACIÓN Y CAPACITACIÓN PROFESIONAL

QUE PARA OBTENER EL TÍTULO DE:

INGENIERO EN COMPUTACIÓN

P R E S E N T A

GUSTAVO EDUARDO LOVERA TORRES

ASESOR: ING. CÉSAR FRANCISCO GERMÁN ROSAS

MÉXICO 2006

*Doy infinitas gracias al Señor por haberme guiado y sustentado durante el desarrollo del presente trabajo y por el aliento que me dio para concluirlo; Sin su ayuda no estaría culminando esta etapa tan importante en mi vida.*

*A mis padres, Rosa y Sergio, gracias les doy porque ellos como instrumentos de Dios, han forjado en mí los valores primordiales de la vida los cuales guardo en mi corazón y estos me han llevado a actuar con diligencia en todo lo que hago.*

*A Agustín y Arturo, mis hermanos, les agradezco todos sus consejos y su apoyo; ustedes tuvieron mucho que ver para que yo llegara a este punto de mi vida.*

*Gracias Señor por darme nuevas fuerzas.*

"pero los que esperan a Jehová tendrán nuevas fuerzas;
levantarán alas como las águilas;
correrán, y no se cansarán;
caminarán, y no se fatigarán."
Is. 40:31

# - ÍNDICE -

## INTRODUCCIÓN

El desarrollo de sistemas en nuestros días se ha ido tornando cada vez más complejo, a medida que se incrementan los requerimientos de robustez y seguridad. Casi a la par que estos requerimientos han sido vitales para desarrollar sistemas confiables, eficientes y seguros, empresas de la talla de Microsoft, Sun Microsystems y Oracle empezaron a ofrecer herramientas de desarrollo bastante poderosas pero, así como los requerimientos de los sistemas, bastante complejas. Pero este, no es el verdadero problema, sino que estas empresas y sobre todo la primera proveen a muy altos costos de licenciamiento sus herramientas lo cual conlleva a elevados presupuestos de desarrollo e implantación de sistemas.

La contraparte al Software propietario es el Software libre que, la mayoría, al estar bajo la licencia GPL (General Public License) permite utilizar, modificar y distribuir sin ningún costo Software diverso como son manejadores de bases de datos, suites para desarrollo en lenguajes de alto nivel, Servidores Web, Sistemas Operativos entre muchos más. Este tipo de Software no por ser libre ofrece menos beneficios que el Software propietario, de hecho en algunos casos ofrece mayores ventajas, sobre todo de seguridad. Aunado a lo anterior, a recientes fechas el movimiento mundial de Software libre ha ido creciendo al grado de que empresas líderes como IBM, Compaq e incluso Oracle han colaborado con este movimiento y ofrecen sus productos con completa compatibilidad con Software libre apoyando así su crecimiento.

Es por todo lo anterior que el presente Trabajo se basa en la Integración de Herramientas Libres para Desarrollar un Sistema de Administración de citas; y es a su vez una síntesis del Diplomado: Desarrollo e Implementación de Sistemas con Software Libre en Linux cursado en el centro Mascarones de la DGSCA en nuestra casa de estudios. En este trabajo se plasman los conceptos básicos del Sistema Operativo Linux, que incluyen, su estructura interna, su sistema de archivos, los archivos de configuración y el manejo del mismo. Todo esto se encuentra comprendido en el Capítulo I.

Los conceptos que integran a los lenguajes de programación, su uso y ejemplos así como el modelado de bases de datos, creación de base de datos, uso de las bases de datos y explotación de datos se tratan en el Capítulo II. También en este capítulo se trata la relación que guardan los lenguajes de programación con las bases de datos.

En el Capítulo III se trata la Administración de los Servidores Web, los cuales son encargados de implantar el servicio Web en Internet. Este capítulo se basa en el Servidor Web Libre, Apache. Este Servidor es el más usado en la actualidad para servir sitios Web. La administración se enfoca principalmente en la configuración de las directivas que dan funcionalidad al Servidor.

Por último en este tercer capítulo, se ofrece una introducción a lo que se conoce como la Seguridad en Cómputo, donde se habla acerca de las políticas y servicios de seguridad, mecanismos de implantación de éstas así como herramientas que apoyan a la Seguridad en Cómputo, las cuales van desde protocolos de comunicación, analizadores de tráfico en una red hasta certificados digitales. Todo esto en conjunto con un análisis de ataques que un sistema puede sufrir; incluyendo ataques que van desde poner en riesgo la confidencialidad de información hasta ataques que afectan el funcionamiento del sistema a través de manipular información, cambiar identidades y denegar un servicio.

El Capítulo IV se enfoca en el desarrollo del Sistema de Administración de Citas, dando en primera instancia un planteamiento de la necesidad por la cual se desarrolla este sistema, así como los objetivos que persigue. Se plantean además los requerimientos mínimos del sistema, para desarrollarlo e implantarlo. Como una etapa inicial del desarrollo se presenta una solución teórica al sistema, haciendo mención de las herramientas Libres utilizadas, su integración y por último el desarrollo formal del sistema.

El desarrollo incluye el modelado de la base de datos y la creación de ésta contemplando la estructura de cada tabla. Se presenta un diagrama de flujo que muestra la operación del sistema y finalmente el diseño de la Interfaz Gráfica de Usuario.

# CAPÍTULO I

# LINUX?, ALGUIEN LO CONOCE?

## 1.1 ¿QUÉ ES LINUX?

Linux se define de una manera sencilla:

***"Es un Sistema Operativo de libre distribución basado en UNIX para computadoras personales, servidores y estaciones de trabajo"***

## 1.2 UN POCO DE HISTORIA

En 1990, Linus Torvalds, un estudiante de 23 años de la Universidad de Helsinki, en Finlandia, comenzó a desarrollar, como hobby, un proyecto basado en el MINIX de Andrew Tenembaum. Quería llevar a cabo, sobre una computadora con procesador Intel 80386, un sistema operativo tipo UNIX que ofreciese más capacidades que el limitado MINIX, que sólo se usaba para enseñar una cierta filosofía de diseño. Quería aprovechar la arquitectura de 32 bits, las propiedades de conmutación de tareas que incorporaba la interfaz en modo protegido del 80386 y eliminar las barreras del direccionamiento de memoria.

Linus empezó escribiendo el núcleo del proyecto en ensamblador, y luego comenzó a añadir código en lenguaje C, lo cual incrementó la velocidad de desarrollo. En octubre de 1991, anuncio la primera versión "oficial" de su Sistema Operativo al cual llamó "LINUX", la 0.02, que ya era capaz de ejecutar el SHELL bash y el compilador gcc de GNU. Después de haber escrito una carta en un foro en Internet, proponiendo a las miembros que probaran su sistema, Linux creció de una manera sorprendente hasta hoy en día. Muchos de los componentes de Linux, como drivers, protocolos o shells salieron de otro sistema UNIX de libre distribución llamado FreeBSD, desarrollado en la Universidad de Berkeley.

## 1.3 CARACTERÍSTICAS DE LINUX

Como sistema operativo, Linux es muy eficiente y tiene un excelente diseño. Es multitarea, multiusuario, multiplataforma y multiprocesador; en las plataformas Intel corre en modo protegido; protege la memoria para que un programa no pueda hacer caer al resto del sistema; carga sólo las partes de un programa que se usan; comparte la memoria entre programas aumentando la velocidad y disminuyendo el uso de memoria; usa un sistema de memoria virtual por páginas; utiliza toda la memoria libre para caché; permite usar bibliotecas enlazadas tanto estática como dinámicamente.

Se distribuye con código fuente; usa hasta 64 consolas virtuales; tiene un sistema de archivos avanzado pero puede usar los de los otros sistemas; y soporta redes tanto en TCP/IP como en otros protocolos.

Linux tiene una estructura en capas, las cuales son:

- º Hardware
- º Kernel
- º Shell
- º Usuario / Aplicaciones

La capa de Hardware es donde se encuentran todos los componentes físicos del sistema, como son: tarjeta madre, tarjeta de red, tarjeta de video, unidades ópticas, floppy, teclado, etc.

La capa del kernel es donde reside el núcleo del sistema. El núcleo es el único que interactúa con el hardware. Incluye entre sus funciones las operaciones más importantes de gestión del sistema operativo, como es la gestión de memoria, mantenimiento del sistema de archivos, asignación de tiempo del CPU a cada una de las tareas, el control del acceso mediante claves, etc.

La capa Shell es donde se encuentra el intérprete de comandos. Este intérprete como su nombre lo indica "interpreta" los comandos que el usuario va introduciendo al sistema. Además es como un pequeño lenguaje de programación con el cual programar nuevas funciones o personalizar algunas existentes

La capa del Usuario / Aplicaciones alberga los programas de aplicación del usuario. En esta capa se encuentra el usuario que se comunica con el shell, el cual a su vez se comunica con el kernel para pasarle las instrucciones que se tiene que realizar y el kernel finalmente es el mediador con el hardware.

## 1.4 PLATAFORMAS Y DISTRIBUCIONES MÁS COMUNES

Linux fue desarrollado para el i386 y ahora soporta los procesadores i486, i586, así como los clones AMD y Cyrix. También soporta máquinas basadas en SPARC, UltraSPARC, Alpha, PowerPC, y Motorola 680x0.

En la actualidad existe una gran cantidad de distribuciones de Linux; una distribución es un paquete que conforma el kernel junto con otros programas creados por diferentes compañías para formar el Sistema Operativo en su conjunto bajo un nombre.

Algunas de estas distribuciones son:

- Red Hat (ahora Fedora)
- Mandrake (ahora Mandriva)
- Suse
- Debian
- Knoppix
- Gentoo
- Ubuntu
- Slackware

De estas distribuciones la que se manejó durante el diplomado fue Slackware, por su estabilidad, seguridad, simplicidad y por su gran parecido al sistema UNIX.

## 1.5 REQUERIMIENTOS DE INSTALACIÓN

Los requerimientos mínimos de hardware para la instalación de Linux Slackware 10.0 son:

| | |
|---|---|
| **Procesador** | i586 o compatible |
| **RAM** | 32 MB |
| **Espacio en Disco** | 1 GB |
| **Unidades** | CD-ROM y/o floppy |

Si se cuenta con el CD booteable probablemente no se necesitará la unidad de floppy. Pero si no se cuenta con la unidad de CD-ROM se necesitará una unidad de floppy para hacer una instalación vía red.

El requerimiento de espacio en disco es un poco engañoso. La recomendación de 1 GB es usualmente para una instalación mínima, pero si se hace de una instalación completa se necesitarán alrededor de 3 GB disponibles de espacio en disco, además de espacio adicional para archivos personales.

Para la conexión con una red se necesita una interfaz de red (tarjeta de red). Con respecto a los periféricos es necesario contar con un monitor, ratón (si se va a utilizar la interfaz gráfica) y teclado.

## 1.6 SISTEMA DE ARCHIVOS

El sistema de archivos de Linux al igual que el Unix es jerárquico arborescente inverso, que empieza en un directorio raíz (root /) inicial. Todos los demás directorios se derivan de éste.

Los directorios que conforman el sistema de archivos de Linux son los siguientes[1]:

**/bin** Programas esenciales de usuario son almacenados aquí. Esto significa que se encuentra únicamente el set mínimo de programas requeridos por el usuario para usar el sistema. Cosas como el shell (intérprete de comandos) y los comandos del sistema de archivos (*ls*, *cp* y demás) están almacenados aquí. Este directorio usualmente no recibe modificaciones después de la instalación. Pero si esto pasa, usualmente es en forma de paquetes de actualización que uno le provee.

**/boot** Contiene los archivos utilizados por el cargador de Linux (LILO). Este directorio recibe pequeñas modificaciones después de la instalación

**/dev** Todo en Linux es tratado como un archivo, aún los dispositivos de hardware como, los puertos seriales, discos duros y scanners. La manera para acceder a estos dispositivos, es un archivo especial llamado "device node", que tiene que estar presente. Todos los "nodos de dispositivo" están almacenados en este directorio.

**/etc** Este directorio contiene los archivos de configuración del sistema. Todo desde el archivo de configuración de X Window, la base de datos de usuarios hasta los scripts de arranque del sistema se encuentran aquí.

**/home** Al ser Linux un sistema multiusuario, a cada usuario del sistema le es dado una cuenta y un directorio único para archivos personales. Este directorio es llamado "el directorio Home del usuario". El directorio /home es proveído como la locación default para los directorios home de los usuarios.

[1]HICKS, Alan. Slackware Linux Essentials, Ed. Slackware Linux Inc. USA 2005 pp. 41-44

**/lib** Las librerías del sistema que son requeridas para la operación básica, están almacenadas aquí. Las librerías de C, el cargador dinámico, las librerías de *ncurses* y los módulos del kernel, están entre las cosas que se encuentran almacenadas aquí.

**/mnt** Este directorio contiene puntos de montaje temporales para trabajar en disco duro o dispositivos removibles. Aquí se encuentran los puntos de montaje para los dispositivos como el CD-ROM y el floppy.

**/opt** Optional Software Packages. La idea detrás de /opt es que cada paquete de software se instale en /opt/software-package lo cual facilita su remoción más tarde. Slackware distribuye algunas cosas en este directorio (como el entorno gráfico KDE en /opt/kde), pero los usuarios son libres de anexar cualquier cosa que quieran a /opt.

**/proc** Este es un directorio único. En realidad no es parte del sistema de archivos, pero es un sistema de archivos virtual que provee acceso a información del kernel. Varias piezas de información que el kernel quiere que el usuario sepa son comunicadas a él a través de archivos en este directorio. Inclusive se puede mandar información al kernel a través de algunos de estos archivos.

**/root** Como se sabe el administrador del sistema es conocido como root, en el sistema de archivos el directorio home de root está en /root además de en /home/root; la razón es simple: ¿Qué pasaría si /home estuviera en una partición diferente que / y no pudiera ser montado?. Root o el administrador del sistema naturalmente querría entrar y reparar el problema; si su directorio home estaba en el sistema de archivos dañado, se le dificultaría entrar.

**/sbin** Aquí se encuentran los programas esenciales que son ejecutados por root y durante el proceso de inicio del sistema son mantenidos aquí. Los usuarios normales no podrán correr los programas en este directorio.

**/tmp** Se utiliza para colocar archivos temporales. Todos los usuarios tienen acceso de lectura y escritura a este directorio.

**/usr** Este es el directorio grande en un sistema Linux. Cualquier otra cosa se almacena aquí: programas, documentación, el código fuente del kernel y el sistema X Window. En este directorio es en dónde comúnmente se instalarán programas.

**/var** Se almacenan aquí los datos que son variables en el tiempo. Aquí se guardan los registros (logs) del sistema.

## 1.7 CONCEPTOS BÁSICOS

### 1.7.1 INICIO Y TÉRMINO DE SESIÓN

Para iniciar sesión en el equipo ya sea en entorno gráfico o desde línea de comandos, el sistema pide un login y un password:

*username*: root

*password*: (no aparecen caracteres mientras se teclea la password)

Para terminar sesión desde línea de comandos existen dos comandos:

*exit* y *logout*

### 1.7.2 CREACIÓN DE UNA CUENTA

Lo que permite a Linux, y a todos los sistemas basados en Unix, ser seguros, es que primeramente para acceder al sistema es necesario contar con una cuenta, es decir, un login y un password. La creación de una cuenta de usuario se realiza de la siguiente forma:

Se abre una consola o terminal y en la línea de comandos se escribe el comando:

*# adduser*

### 1.7.3 CONSOLAS VIRTUALES

Debido a que Linux es un sistema multiusuario, cuando se está trabajando y se necesitan hacer varias cosas, uno como usuario se puede logear varias veces; esto se logra a través de "consolas virtuales". Presionando la tecla Alt y una tecla de función (F1, F2,…, F12), se puede cambiar entre consolas virtuales, cada tecla de función corresponde a una. Slackware proporciona 6 consolas virtuales es decir hasta la tecla de función F6, el resto de las teclas de función están reservadas para sesiones de X (entorno gráfico). Cada sesión de entorno gráfico usa su propia consola virtual comenzando con la tecla de función F7 (Alt+F7), hacia arriba.

Cuando se está en una sesión de entorno gráfico y la combinación de teclas Alt+Tecla de función es reemplazada por Ctrl+Alt+Función y en el caso que se desee regresar a una consola en modo texto (y no se quiere terminar sesión gráfica) ctrl.+Alt+F3, por ejemplo, llevará a la consola virtual número tres y Alt+F7 conducirá de regreso a la sesión gráfica (asumiendo que se estaba usando la primera sesión gráfica).

### 1.7.4 CAMBIO DE CONTRASEÑA

Con la ejecución del programa passwd se pueden cambiar las passwords del sistema. Este programa lo que hace es ir al archivo /etc/shadow, donde se encuentran las passwords encriptadas de todos los usuarios del sistema y cambia la password del usuario indicado.

```
# passwd
```

### 1.7.5 ARCHIVOS Y DIRECTORIOS

En Linux al igual que en Unix todo se maneja como un archivo; un directorio es un archivo que contiene archivos e inclusive otros directorios. A continuación se da una breve explicación de cada comando disponible en Linux para el manejo de archivos y directorios, junto con un ejemplo.

> Directorios raíz y home

Como se mencionó en el apartado de Sistema de Archivos de Linux, todo los archivos del sistema parten de un directorio llamado directorio Raíz (/).

El directorio home del usuario root (superusuario), se encuentra en dos lugares: /home/root y /root. Los directorios home del resto de los usuarios se encuentran en /home.

> Directorio de trabajo, ***pwd***

El comando ***pwd*** indica cual es el directorio actual de trabajo.

> Cambio de directorio, ***cd***

Para poder moverse a través de directorios en el sistema de archivos se dispone del comando ***cd***. P. ejem.:

```
# cd /home/root
```

> Listado de directorios, ***ls***

Para ver el contenido en forma de lista de un directorio se cuenta con el comando ***ls***. Este comando además cuenta con varias opciones, de entre las cuales están:

**-l** muestra el listado en formato largo.
**-a** muestra todo el contenido del directorio, incluyendo archivos y directorios ocultos
**-i** muestra el listado con el número de i-nodo
**-r** muestra el listado en orden inverso alfabéticamente
**-R** muestra el listado en forma recursiva
**-t** muestra el listado tomando como opción la fecha de modificación

> Creación de directorios, ***mkdir***

Existen en Linux un comando básico para crear directorios, ***mkdir***.

```
# mkdir directorio
```

Sólo es necesario teclear el comando y a continuación el nombre del directorio que se desea crear.

> Copiar archivos y directorios, ***cp***

El comando ***cp*** permite hacer copias de archivos uno a uno, múltiples archivos e incluso hacer copias de estructuras de directorios. Este comando tiene varias opciones, entre las que se encuentran:

**-r** copia recursiva
**-v** forma verbosa, va informando el nombre del archivo del que se está haciendo copia
**-i** interactivo, va preguntando al usuario

```
# cp arch1 arch2 /home/eddie
```

En este ejemplo se copian los archivos arch1 y arch2 al directorio home del usuario eddie.

> Mover o renombrar archivos y directorios, ***mv***

El comando *mv* tiene dos funciones básicas; mover archivos o estructuras de directorios de un lugar a otro y la segunda es renombrar archivos o directorios.

```
# mv saludos musica/
```

Aquí se está moviendo el archivo saludos al directorio /musica.

```
# mv saludos hola
```

En este caso se está renombrando el archivo saludos con el nombre de hola.

> Borrar archivos y directorios, ***rm*** y ***rmdir***

***rm***, este comando borra archivos, puede ser borrando un archivo a la vez o varios haciendo uso de los metacarácteres * ?.

```
# rm pr??ba
```

En este ejemplo se borrará el archivo cuyo nombre tenga como las dos primeras letras la p y la r luego dos letras cualesquiera y al final las letras b y a.

Con el comando ***rmdir*** se borran directorios o estructuras de ellos si se utiliza la opción –p, siempre y cuando los directorios estén vacíos. De otra manera se debe de utilizar el comando ***rm*** con la opción –r.

> Mostrar contenido de archivos, ***cat*** y ***more***

Los comandos ***cat*** y ***more*** nos permiten ambos ver el contenido de archivos, pero con algunas diferencias. El comando ***cat*** muestra el contenido de uno o varios archivos pero de forma continua, es decir, aunque el contenido del archivo sea más grande que el tamaño de la pantalla, se muestra todo de una sola vez. Mientras que el comando ***more*** va mostrando página por página e incluso línea por línea el contenido del archivo.

Un comando similar a ***more*** es ***less***, que además de los beneficios que nos otorga ***more*** agrega la funcionalidad de poder ir hacia delante y hacia atrás en el contenido del archivo con las teclas de navegación (↑↓).

Un par de comandos funcionales además de los mencionados son: ***head*** y ***tail***. Estos comandos sirven para mostrar las primeras n líneas y las últimas n líneas respectivamente, de un archivo.

### 1.7.6 RUTAS ABSOLUTAS Y RUTAS RELATIVAS

Una ruta es el camino que se sigue a través del sistema de archivos para llegar hasta un archivo o directorio en particular. Las rutas absolutas son la representación "explícita" de todos los directorios por los cuales se pasa para llegar al directorio deseado.

```
# cd /etc/X11/sysconfig
```

Las rutas absolutas inician con la raíz del sistema de archivos / y van especificando los directorios por los que se tiene que pasar hasta llegar al directorio específico, en este ejemplo al directorio sysconfig.

Por su parte las rutas relativas comienzan con la representación del directorio en el que se esté actualmente (.). Y de ahí se parte hacia otros directorios con la representación del directorio (..) en el cual esta contenido el directorio actual.

# *cd ./../..*

En este ejemplo la ubicación última sería /etc. Esto es, al indicar con un punto la ubicación actual y después con .., el sistema nos ubica en el directorio que contiene al directorio sysconfig que es X11; al indicarle de nuevo al sistema otros dos puntos nos ubica en el directorio que contiene al directorio X11 que es etc.

## 1.7.7 AYUDA EN LÍNEA

Linux cuenta con una ayuda sobre los diferentes recursos del sistema. Esta ayuda es el manual que se invoca con el comando *man* y esta dividido en 8 secciones:

1 Herramientas de Usuario
2 Llamadas al sistema
3 Subrutinas y funciones de bibliotecas
4 Información de controladores de dispositivos
5 Archivos de configuración
6 Juegos
7 Paquetes
8 Administración del sistema

## 1.7.8 REDIRECCIONAMIENTO

> Entrada y salida estándar

En Linux la entrada y salida de datos están definidas. De manera que la entrada estándar de datos es el teclado y salida estándar de datos es la pantalla. Pero estas formas de introducir o sacar datos del sistema se pueden cambiar con un concepto llamado “Redireccionamiento”.

> Redireccionamiento de entrada y salida, >, <

Si se requiere que los datos que se obtienen, por ejemplo, de la ejecución de un comando como *ls*, se transfieran a un archivo para su posterior manejo, entonces en Linux se puede utilizar el redireccionamiento de salida >.

# *ls* > listado

En este ejemplo se toma la salida del comando ***ls*** y se introduce al archivo listado. Si el archivo existe se borra todo lo que tiene el archivo y se introducen los datos arrojados por el comando. Si el archivo no existe se crea.

El redireccionamiento de entrada < se usa para obtener datos desde un archivo. Los datos introducidos en el archivo listado del ejemplo ahora se pueden redireccionar como entrada del comando ***cat*** para visualizar el contenido del archivo.

# ***cat*** < listado

> Uso de tuberías, |

Las tuberías funcionan como un filtro, mandando la salida de la ejecución de un comando, por ejemplo, como entrada de otro comando. Por ejemplo si se quieren mostrar los primeros 3 archivos que contiene un directorio, entonces se ejecuta el comando *ls* luego se agrega la tubería | y a continuación el comando que permite ver las primeras tres líneas ***head*** -3.

# ***ls*** | ***head*** -3

> Redireccionamiento no destructivo

>> esta expresión es usada para hacer un redireccionamiento no destructivo, es decir, hace lo mismo que el redireccionamiento de salida, pero si el archivo ya está creado y tiene datos, agrega los nuevos datos al final de los que ya estaban, sin borrar los anteriores

## 1.7.9 PERMISOS

> Permisos de archivos

En Linux cada aspecto del sistema es multiusuario incluyendo el sistema de archivos. El sistema almacena información como quién es el propietario de un archivo y quien puede leerlo. Se almacena información del usuario para cada archivo y directorio. Esto incluye qué usuario y grupo es propietario de un archivo en particular.

Por otra parte todos los archivos tienen atributos, es decir, son de un tipo:

| | |
|---|---|
| **d** | directorios |
| **c** | archivos de tipo carácter (archivos de acceso por hardware) |
| **l** | liga |
| **b** | archivo de bloque (acceso por bloques) |
| **s** | archivos de tipo socket |
| **-** | archivos ordinarios |
| **p** | pipes |

La manera de poder ver esta información, es con el comando:

```
# ls -l
```

Al ejecutar este comando se muestra el listado de los archivos y directorios del directorio actual con información como el tipo de archivo y en las siguientes columnas los permisos de la siguiente forma:

**rwxrwxrwx**

***r*** lectura (read)
***w*** escritura (write)
***x*** ejecución (execution)

La primera tercia de letras se refieren a los permisos que tiene el propietario del archivo; la segunda tercia se refiere a los permisos que tiene el grupo sobre el archivo; y la tercera se refiere a los permisos que tiene cualquier otro usuario.

Cuando se crea un archivo ordinario, por default se le asignan los siguientes permisos:

r w - r - - r - -

Esto es, permisos de lectura y escritura para el propietario, permiso de lectura para el grupo y permiso de lectura para otros usuarios.

Para un directorio:

r w x r - x r - x

Esto es; permisos de lectura (ver contenido), escritura (escribir dentro) y ejecución (ingresar), para el propietario, permisos de lectura (ver contenido) y ejecución (ingresar) para el grupo y otros usuarios.

> Cambio de permisos, ***chmod***

Para cambiarle los permisos a un archivo existe el comando ***chmod***. Este comando trabaja de dos formas, una de ellas es utilizando la siguiente notación:

***u*** usuario

***g*** grupo

***o*** otros

\+ agregar

\- quitar

La otra forma de trabajar del comando ***chmod*** es con notación octal.

| **Permiso** | **Valor octal** |
|---|---|
| Leer | 4 |
| Escribir | 2 |
| Ejecutar | 1 |

De manera que si queremos darle al archivo listado los permisos siguientes: leer y ejecutar para el usuario, grupo y otros, el comando a ejecutar es:

```
# chmod 555 listado   // el primer número es para permisos de usuario, el segundo para
                      // permisos del grupo y el tercero para permisos a otros.
```

### 1.7.10 LIGAS

> Ligas duras y suaves

Las ligas son apuntadores entre archivos. Con las ligas se pueden tener archivos en muchos lugares del sistema y ser accedidos por diferentes nombres. Existen dos tipos de ligas: duras y suaves.

Las ligas duras son nombres para un archivo en particular, sólo pueden existir en un sistema de archivos simple y sólo son removidas cuando el nombre real es removido del sistema.

La liga suave, también llamada liga simbólica, puede apuntar a un archivo fuera de su sistema de archivos. De hecho es un pequeño archivo que contiene la información que necesita. Se pueden agregar o remover ligas suaves sin afectar al archivo actual. Y por el hecho de que una liga simbólica es un archivo que contiene su propia información, puede apuntar también a un directorio. En ocasiones es común tener /var/tmp siendo una liga simbólica a /tmp por ejemplo. Las ligas no tienen su propio set de permisos o propietario en lugar de eso reflejan estos datos del archivo al que están apuntando.

> Creación de ligas

Para crear ligas ya sean duras o suaves, se utiliza el comando ***ln*** (link). Si se desea crear una liga suave sólo se necesita utilizar la opción -s:

```
# ln –s /home/eddie/passwd2.jpg pass
```

En este ejemplo se creó una liga suave hacia el archivo passwd2.jpg que se encuentra en el directorio home del usuario eddie, y se creó con el nombre pass.

Para la creación de una liga dura sólo es necesario utilizar el comando ***ln*** sin ninguna opción, solo el nombre del archivo al que se ve a hacer la liga y el nombre de la liga.

### 1.7.11 PROCESOS

Cada programa que se encuentra ejecutándose en el sistema es llamado proceso[2]. Estos procesos son desde el sistema X Window hasta programas del sistema (demonios) que son inicializados desde que la computadora arranca. Cada proceso corre como propiedad de un usuario en particular. Procesos que son inicializados en el tiempo de arranque usualmente corren como propiedad del usuario root o nobody. Procesos que uno como usuario inicializa correrán como propios.

Cada usuario tiene control sobre los procesos que inicializa. Adicionalmente, root tiene control sobre todos los procesos del sistema incluyendo aquellos que son inicializados por otros usuarios. Los procesos pueden ser controlados y monitoreados a través de varios programas, así como con algunos comandos.

> Primer plano y segundo plano

Los programas inicializados desde la línea de comandos inician en el primer plano. Esto permite ver la salida del programa e interactuar con él. A veces hay ocasiones en las que sería mejor que el programa corra sin tomar el control de la terminal, es decir, que no impida el poder ejecutar otros programas o comandos en lo que ese programa se encuentra corriendo. A esto se le llama correr un programa en segundo plano, y hay algunas maneras de hacer esto.

La primer forma de ejecutar en segundo plano un proceso es agregando un ampersand (&) al final del comando cuando se inicia un programa. Por ejemplo si se quiere usar el reproductor de mp3 de línea de comandos amp para reproducir un directorio lleno de archivos mp3, pero se necesitan realizar otras tareas en la misma terminal. El siguiente comando va a iniciar amp en segundo plano:

```
# amp *.mp3 &
```

[2]Ibid p. 133

La otra forma de correr un proceso en segundo plano es hacerlo mientras está corriendo. Primero se inicia el programa, mientras está corriendo se presiona ctrl.+z, esto suspende el proceso. Un proceso suspendido está básicamente pausado, momentáneamente deja de correr, pero puede ser iniciado otra vez en cualquier momento. Una vez que se ha suspendido el proceso, el prompt es regresado (recuérdese que cuando un programa se está ejecutando el prompt deja de ser visible y por lo tanto no se puede introducir ninguna instrucción al sistema), es entonces cuando se puede mandar a segundo plano el proceso con el siguiente comando:

# *bg*

Ahora el proceso suspendido estará corriendo en segundo plano.

Si se necesita interactuar con proceso que esta corriendo en segundo plano, se puede traer de nuevo al primer plano. Si sólo se tiene un solo proceso corriendo en segundo plano, se puede traer al primer plano tecleando:

# *fg*

Si el programa no ha terminado de correr, el programa tomará el control de la terminal, y como ya he comentado antes, si esto ocurre el prompt no va a ser regresado. Algunas veces el programa termina mientras está corriendo en segundo plano. En este caso el sistema nos mostrará un mensaje como el siguiente:

**[1]+ Done /usr/bin/ls $LS_OPTIONS**

Esto indica que el proceso ***ls*** que se mando a segundo plano ha terminado.

Es posible tener varios procesos en segundo plano a la vez. Cuando esto ocurre se necesita saber cual proceso es el que se quiere traer a primer plano. Con sólo teclear el comando ***fg***, se trae a primer plano el último proceso que fue mandado a segundo plano, pero si se quisiera traer a primer plano otro proceso que no sea éste, ¿como se haría?. Pues bien el shell bash cuenta con un comando muy útil que muestra los procesos que están corriendo en segundo plano o detenidos; el comando ***jobs***. De esta manera se puede traer al primer plano al proceso que se desee con el comando # ***fg*** + *numero_proceso*

Se sabe que hay procesos que están corriendo todo el tiempo en el sistema. Para poder ver estos procesos se cuenta con el comando ***ps***.

Aparecen varias columnas. La primera columna contiene el PID del proceso, es decir, un identificador de proceso que puede estar entre 1 y 32767. La segunda columna indica en cual terminal esta corriendo dicho proceso. En este caso como el comando se esta corriendo sin opciones, sólo va a mostrar los procesos que se están ejecutando en la terminal actual. De manera que todos los procesos van a indicar la misma información en la columna TTY. La tercer columna, TIME, indica que cantidad de tiempo de procesador ha ocupado el proceso para correr. Este tiempo es diferente a la cantidad de tiempo que un proceso corre. Finalmente la columna CMD, muestra que programa es el que está corriendo. Sólo muestra el nombre básico del programa, no opciones de comandos o información similar. Se puede obtener más información sobre los procesos utilizando las opciones indicadas, que pueden ser consultadas en el manual.

En ocasiones cuando un programa se esta portando mal, se necesita ponerlo en su lugar. El programa para realizar este tipo de administración se llama ***kill*** (y se ejecuta con el comando del mismo nombre), y puede ser usado para manipular procesos de muchas maneras. El uso más obvio de ***kill*** es para "matar" un proceso. Se necesita hacer esto si un programa se sale de control y empieza a utilizar muchos recursos del sistema, o si simplemente se desea que deje de correr.

Para matar un proceso lo único que se debe de saber es su PID o su nombre. Por ejemplo si se quiere matar el proceso 1210, se tiene que ejecutar el comando ***kill*** de la siguiente forma:

```
# kill 1210
```

Claro que para poder hacer esto, se debe ser dueño del proceso. De otra manera el sistema no permitirá que se mate tal proceso, a excepción que se sea root, el cual puede matar cualquier proceso.

Hay ocasiones en que la ejecución simple de ***kill*** no hace bien el trabajo, es decir, hay procesos que no mueren con un ***kill***, sino que se necesita algo más potente, como la opción -9 que seguramente causará la muerte del proceso que se desea.

Finalmente existe un comando que se utiliza para mostrar información actualizada acerca de los procesos que están corriendo en el sistema. Este comando es ***top***.

Al ejecutarse despliega una pantalla completa de información acerca de los procesos corriendo, además de otra información del sistema como, promedio de carga, número de procesos, el status del CPU,.memoria libre y detalles de los procesos incluyendo PID, usuario, prioridad, información de uso de CPU y memoria, tiempo de corrida y nombre del programa.

## 1.8 INSTALACIÓN DE LINUX

Antes de empezar la instalación de Slackware 10.0, se realizan algunos pasos previos. Lo primero que se hace es particionar el disco duro con la utilidad fdisk, para poder guardar diferentes directorios en cada partición. Las particiones son las siguientes:

**Swap……. 256 MB**

**/……. 3 GB**

**/home…… 4 GB**

**/var/spool/mail…….. 2 GB**

**/usr/local……... 5 GB**

La partición swap se utiliza como espacio de intercambio entre la memoria RAM y el disco duro. Como una recomendación se debe de crear esta partición tomando en cuenta la memoria RAM total; la partición swap debe ser el doble de la memoria RAM, pero por otro lado no debe ser mayor de 256 MB. En nuestro caso como las máquinas del aula tenían 512 MB nos limitamos a hacer la partición swap del máximo recomendado (256 MB).

/ (partición raíz), es dónde reside en sí el sistema operativo. Es aquí dónde se instalan todos los componentes del sistema; desde archivos propios del sistema, utilerías, programas y más.

La partición /home se crea para que los usuarios tengan disponible este espacio en disco para sus archivos. Esto permite que los usuarios no excedan el espacio que tienen asignado. El tamaño de esta partición depende de cuántos usuarios va a tener el sistema, en nuestro caso de manera arbitraria le asignamos 4 GB.

Para el caso de la partición /var/spool/mail, ésta se crea para almacenar los correos electrónicos de los usuarios; de igual manera que la partición /home, esta partición depende de los usuarios que vaya a tener el sistema.

/usr/local, esta partición se hace generalmente para que ahí residan todos los programas que se instalan, razón por la cual su tamaño su tamaño es más grande que la de las otras particiones inclusive que la del sistema.

Cada disco y dispositivo en el sistema es considerado como un gran sistema de archivos; algunas particiones, CD-ROM's, y floppies, son colocados en el mismo árbol. Para adjuntar estas unidades al sistema de archivos, y poder accederlos se usan los comandos *mount* y *umount*.

Algunos dispositivos son automáticamente montados cuando arranca el sistema. Estos están enlistados en el archivo /etc/fstab.

Continuando con el proceso de instalación, después de hacer las particiones se procede con la instalación propia del sistema. Para correr el programa que instala el s.o. se utiliza el comando:

# *setup*

Este comando comienza la instalación del sistema operativo

Se muestra un menú de varias opciones entre las cuales están: Ayuda para la instalación, remapeo del teclado, establecer una partición swap, seleccionar categorías de software para instalar, entre otras. En este momento se establece la partición swap.

Como ya se había hecho la partición swap, el programa de instalación la detecta y pregunta si se desea instalar ésta como la partición swap. Se acepta (Yes).

Después de haber establecido la partición swap, el programa pregunta por la partición en dónde se va a instalar Linux. De igual manera que con la partición swap se detectan automáticamente las particiones en el disco y pregunta en cual se va a instalar el s.o.; se seleccionó la indicada (Select)

En seguida se debe de formatear la partición, de manera que se muestra una nueva pantalla dónde pregunta por modo de formateo: rápido, lento o no formatear. Si a la hora de instalar el tiempo es crucial, se escoge el formateo rápido; aunque un formateo lento es mejor, ya que se hace un chequeo de bloques dañados en el disco duro.

Existen varios tipos de sistemas de archivos, el programa de instalación muestra tres opciones de estos sistemas para formatear la partición: ext2, ext3 y reiserfs. Por sus características y desempeño se escogió el sistema **reiserfs**.

Este sistema de archivos trabaja de la siguiente forma[3]:

En esencia este sistema de archivos trata la partición del disco duro como si fuera una tabla simple de una base de datos. Directorios, archivos y metadatos de archivos son organizados en una eficiente estructura de datos llamada "árbol balanceado". Esto difiere un poco de la forma en la que los sistemas de archivos tradicionales operan, pero ofrece mejorías de velocidad para muchas aplicaciones, especialmente aquellas que usan muchos archivos pequeños. Además este sistema es "journaled" (bitacoreado).

Una vez que termina el formateo de la partición /, el programa pregunta por los paquetes de software que se desean instalar. Desde el sistema base hasta entornos para el sistema X.

Ya que se seleccionan los paquetes de software, aparece una nueva pantalla donde el programa pregunta por el modo en el que se desea que vaya informando acerca de lo que se va instalando. En este caso se selecciona el modo completo (full), que requiere un poco más de 3 GB de espacio en disco.

Después de esto el programa de instalación pregunta si se desea instalar un kernel, si se prefiere instalarlo desde el disco de Slackware escogiendo algún kernel en especial o si se prefiere saltar este paso y dejar algún kernel que ya haya sido instalado. En este caso se salta este paso. Seguida la instalación se pregunta por la configuración de un modem. En este punto no se configura; pero si se desea configurarlo, una vez instalado el sistema operativo se puede hacer.

El siguiente paso es la instalación del cargador de Linux (LILO). Aparece una pantalla con tres opciones:

La primera opción es la forma simple, es decir, trata de instalar LILO de forma automática, aún si encuentra otro sistema operativo como Windows. La forma experta ofrece el control sobre le proceso de instalación. En este caso esta forma se utiliza. Y finalmente la opción de saltar este paso para poder después instalar LILO con el comando liloconfig.

[3] http://www.linuxplanet.com/linuxplanet/tutorials/2926/4/

Enseguida pregunta el programa por la utilización de la frame buffer console o la standard Linux console. Para tener una mejor resolución se utiliza la frame buffer console de 1024x768x32k, aunque esto depende directamente que la tarjeta de video y el monitor soporten está resolución. Para este caso se escoge la standard Linux console, que es lo más seguro cuando no se saben las capacidades de la tarjeta de video y del monitor. Esta resolución se aplica para el arranque del LILO y permite ver más filas y columnas de texto en la pantalla a la hora del arranque.

Por último en la instalación del LILO, el programa pregunta por el destino donde se va a instalar el cargador. Como siempre presenta varias opciones. Se instala en el MBR (Master Boot Record).

Una vez que se instaló el cargador de Linux, continúa la configuración del mouse y del programa gpm que permite cortar y pegar texto en consolas virtuales usando el mouse. A continuación de esto, el programa de instalación pregunta si se desea configurar la red. A lo cual se responde que sí:

Lo primero que se pide es el nombre del host, es decir, el nombre del equipo. Además del dominio al cual va a pertenecer ese host (unam.mx). Después de esto pide la dirección ip para la tarjeta de red del host; ya sea que la dirección fuera estática, vaya a ser asignada por dhcp o sino se va a conectar a una red, la dirección de loopback. En este caso se escoge la asignación de ip por dhcp. Al escoger este tipo de ip el programa de instalación pide el nombre del dhcp; si se conoce el nombre del dhcp se proporciona. Con esto ya se encuentra configurada la red; se muestra una pantalla de confirmación de los parámetros que se configuraron anteriormente, si todo está correcto se acepta (Yes).

A continuación se deben de seleccionar los servicios que se requiere que se inicien al arrancar el sistema; servicios como el servidor de archivos y de impresión para redes Macintosh Netatalk Appletalk, el servidor de DNS BIND, el servidor de impresión CUPS, el servidor Web Apache, entre otros.

Después de los servicios anteriores, se debe de seleccionar el administrador de ventanas que va a utilizar el sistema X Window. Se selecciona el ambiente KDE, por ser el más estable.

Por último aparece un mensaje de advertencia que indica que no hay ninguna password del usuario root detectada. En este punto de debe de establecer una password para root. Se debe de saber que por motivos de seguridad no se debe de apuntar en ningún lado la password y además debe ser una password fuerte, es decir, hacer combinaciones de letras minúsculas con letras mayúsculas y números. Una vez establecida la password del usuario root, aparece un mensaje indicando que la instalación se ha completado. Se debe de reiniciar el equipo para que quede por completo instalado el sistema operativo.

## 1.9 ADMINISTRACIÓN DE LINUX

### 1.9.1 PERFIL Y ACTIVIDADES DEL ADMINISTRADOR

El Administrador del sistema debe de planear las actividades que se van a desarrollar. Por ejemplo, el hacer respaldos de información, apagar el sistema para darle mantenimiento al hardware, dar de alta cuentas de usuario, dar de baja cuentas de usuario, asignar espacio en disco, etc.

Una actividad importantísima del Administrador es hacer frecuentemente copias de seguridad del sistema; esto es vital, ya que si por algún motivo se llegan a perder archivos del sistema, recaerá en una pérdida de información, mal funcionamiento del sistema y posiblemente la caída por completo del sistema.

Como responsable de mantener a punto el sistema, al Administrador debe conocer las utilerías básicas de administración del sistema, tales como find, cron, df, du, entre otras. Además es necesario que el administrador conozca la documentación del sistema. Esta documentación se puede encontrar en libros, manuales, Internet y en el mismo sistema. De igual forma es indispensable que el Administrador conozca el hardware del sistema, ya que esto le permitirá configurar de manera adecuada de todos los componentes y dispositivos, utilizando los controladores apropiados para cada uno.

Para saber si el hardware que se pretende utilizar en el sistema es compatible con la distribución de Linux que se esté utilizando, en nuestro caso Slackware, es necesario acudir a la página web[4] que proporciona la HCL (Hardware Compatibility List), donde se enlistan todos los componentes de hardware que son compatibles con las diferentes distribuciones de Linux, agrupados por marcas.

[4] http://www.linuxquestions.org/hcl/

Para mantener un control del sistema, el Administrador debe también de establecer políticas de uso, así como políticas de administración. Dentro de las políticas de uso, debe de establecer puntos sobre cómo, desde dónde y en que horarios pueden los usuarios ingresar al sistema, por ejemplo. Dentro de las políticas de administración se deben de establecer puntos sobre la privacidad en la información de cada usuario, horarios para el mantenimiento del sistema, horarios para hacer respaldos de información, frecuencia con la cual se debe de actualizar el sistema o módulos de él, etc.

Por último para tener una administración completa, es importante dentro de las actividades del Administrador del sistema mantener un canal de comunicación con los usuarios. Puede ser vía correo, o mandándoles mensajes directamente a cada usuario. Esto con el fin de darles avisos acerca de servicios, o cuando, por ejemplo, algún usuario incurre en algo no permitido.

### 1.9.2 CONCEPTOS BÁSICOS DE ADMINISTRACIÓN

> Uso de LILO

El Linux Loader (Cargador de Linux), LILO es el arrancador más popular en los sistemas Linux. Es completamente configurable y puede ser fácilmente utilizado para arrancar otros sistemas operativos. Slackware viene con una utilidad configurable a través de menús llamada *liloconfig*. Este programa es primero ejecutado durante el proceso de arranque, pero se puede invocar después tecleando en la línea de comandos *liloconfig*.

LILO lee su configuración del archivo **/etc/lilo.conf**. Este archivo no es leído cada vez que se arranca la computadora, pero en cambio es leído cada vez que se instala LILO. LILO debe ser reinstalado en el sector de arranque cada vez que se hacen cambios de configuración.

> Comandos ***shutdown***, ***halt*** y ***reboot***.

El sistema siempre debe ser apagado apropiadamente. Si se apaga con el switch de energía se pueden ocasionar graves daños al sistema de archivos. Mientras el sistema está encendido hay archivos que están siendo usados aún si no se está haciendo nada. Hay que recordar que hay muchos procesos que están corriendo en segundo plano todo el tiempo. Estos procesos están manejando el sistema y mantiene muchos archivos abiertos. Cuando el sistema se apaga con el switch de energía, estos archivos no son cerrados apropiadamente y pueden corromperse.

Una de las formas apropiadas para apagar el sistema es con el programa ***shutdown*** (comando del mismo nombre). Este comando sirve para reiniciar o apagar el sistema en un tiempo dado y puede desplegar un mensaje a todos los usuarios que estén trabajando en el sistema diciéndoles que el sistema se está apagando.

El uso más básico para el comando ***shutdown*** es:

```
# shutdown –h now
```

En este caso no se va a mandar ningún mensaje a los usuarios, ellos sólo van a ver el mensaje default que muestra este comando. "now" es el tiempo en el que se quiere apagar el sistema, y la –h significa que se detenga el sistema. Esta es una forma poco amigable de apagar un sistema multiusuario, pero funciona bien en un sistema casero. Un mejor método para un sistema multiusuario sería mandarles a los usuarios un aviso anticipado de que se va a apagar el sistema

```
# shutdown –h +60
```

Este comando apagaría el sistema en una hora (60 minutos), que sería lo correcto en un sistema multiusuario. Para reiniciar el sistema con este comando se intercambia la opción "–h" por "–r".

```
# shutdown –r now
```

La segunda forma para detener o apagar el sistema es usando los comando ***halt*** y ***reboot***. ***halt*** inmediatamente va a detener el sistema operativo y ***reboot*** va a reiniciar el sistema (*reboot* es en realidad sólo una liga simbólica a *halt*). Se invocan de la siguiente manera:

```
# halt
# reboot
```

> El archivo inittab

Este archivo describe cómo el proceso INIT debe iniciar el sistema en el run-level indicado. Los run-levels van desde 0 hasta 6. En general este archivo no se debería de modificar, pero al echarle un vistazo puede indicar porque algunas cosas en el sistema están trabajando en la manera en la que lo hacen.

> Manejo de usuarios y grupos

+ Añadir usuarios

Como ya se mencionó anteriormente, para agregar un usuario al sistema se utiliza el comando:

# *adduser*

Cuando se requiere borrar un usuario se utiliza el comando:

# *userdel* <nombre_usuario>

+ Comandos ***chsh*** y ***chfn***

Estos comandos sirven para cosas específicas. El comando ***chsh*** sirve para cambiar el shell del usuario. Cuando se ejecuta este comando el sistema pide la password al usuario y después le pide que ingrese la ruta completa del nuevo shell que va a utilizar, p. ej. /bin/bash.

El comando ***chfn*** permite cambiar la información opcional introducida durante el proceso de creación de la cuenta del usuario, como es el nombre completo, números de teléfono y número de departamento. Como es usual root puede cambiar esta información de cualquier usuario con sólo indicar como argumento el nombre del usuario.

> Archivar y comprimir

+ Uso de ***tar***

***tar*** es el empaquetador de GNU. Este comando trabaja tomando varios archivos o directorios y crea un archivo más grande. Esto permite comprimir un árbol entero de directorios, lo cual es imposible sólo usando *gzip* o *bzip2* (los explico en la siguiente sección). El uso más común de ***tar*** es para descomprimir y desempaquetar un paquete que haya sido, por ejemplo, bajado de un sitio web o ftp. La mayoría de los archivos vienen con la extensión "*.tar.gz*". Esto normalmente es conocido como "tarball". Esto significa que varios archivos fueron empaquetados usando ***tar*** y luego comprimidos usando ***gzip***.

***tar*** tiene varias opciones de entre las cuales están:

| | |
|---|---|
| **-c** | crea archivo y sobrescribe |
| **-v** | muestra lo que está empaquetando |
| **-f** | manda a un archivo |
| **-x** | desempaqueta |
| **-z** | empaqueta y comprime |

# *tar* –cvf archivo.tar .

Este comando, con la opción –c, crea el archivo, lo hace forma verbosa con la opción –v y con la opción –f le indica que mande la salida a un archivo ya que por default *tar* manda la salida a la salida estándar.

# *tar* –zxvf respaldo.tgz

Este comando es un ejemplo de cómo tar realiza una descompresión y desempaquetamiento. Si se quiere empaquetar y comprimir se usaría el mismo comando para sin la opción –x.

+ Uso de ***gzip***

***gzip*** es el programa de compresión de GNU. Trabaja tomando un sólo archivo y lo comprime. Su uso básico es de la siguiente forma:

# *gzip* compresion compreson.gz

El archivo resultante tendrá el nombre compresion.gz y usualmente es más pequeño que el archivo inicial. El archivo compresion.gz reemplaza a compresión. Los archivos regulares de texto se comprimen bien, mientras que las imágenes jpg, mp3´s y otros archivos como estos no se comprimen bien ya que estos ya están comprimidos.

Para descomprimir archivos que fueron comprimidos con ***gzip*** se pueden utilizar dos comandos que en realidad son el mismo programa. Gzip descomprime cualquier archivo con una extensión de archivo reconocida. Algunas de estas extensiones son: .gz, -gz, .z, -z, .Z o -Z. El primer método es con el comando gunzip.

# *gunzip* respaldo.gz

Este comando descomprime el archivo y lo coloca en el directorio actual. ***gunzip*** realmente es parte de ***gzip*** y es idéntico a utilizar ***gzip*** –d.

### 1.9.3 CONFIGURACIÓN DE LA INTERFAZ GRÁFICA

Al instalar el sistema operativo, por defecto, el sistema inicia en modo texto, este modo de arranque es especialmente útil para la administración de Linux, ya que no se necesita un entorno gráfico para ello[5]. De cualquier manera si se quiere levantar el entorno gráfico, existe un comando que lo hace: *startx*. Este comando funciona siempre y cuando la configuración de la tarjeta de video sea la adecuada; Si no se conoce a detalle el hardware con que se cuenta es difícil la correcta configuración. Un método que se utiliza para lograr levantar la interfaz gráfica fue sobrescribiendo el archivo xorg.conf con el archivo xorg.conf-fdev. Aquí es donde se guarda la configuración de la interfaz gráfica.

Si se desea que el sistema arranque siempre en modo gráfico, existen varias maneras de configurar el sistema para ello. Una de ellas es ir al archivo /etc/rc.d/rc.local y agregar la siguiente línea:

*/opt/kde/bin/kdm*

Otra forma es ir al archivo inittab y cambiar el nivel de arranque (run-level) al nivel indicado.

[5]Op. Cit. p. 75

### 1.9.4 ADMINISTRACIÓN DE PAQUETES

Un paquete es un conjunto de programas relacionados listos para instalarse. Cuando se baja de Internet un archivo de código fuente, uno tiene que configurarlo, compilarlo e instalarlo a mano. Con un paquete, esto ya ha sido hecho. Lo único que se tiene que hacer es instalarlo. Otra buena característica de utilizar paquetes, es que son fáciles de remover y de actualizar.

Linux Slackware viene con programas para la administración de paquetes. Algunos de estos son: **Pkgtool** e **Installpkg**.

### 1.9.5 INSTALACIÓN DE PROGRAMAS

Si el programa fue bajado de Internet y se encuentra en un archivo con extensión .tgz primero se tiene que descomprimir:

# *tar* –zvxf nombre_archivo.tgz

Después se configura:

# ./*configure* //Revisión del sistema (librerías y programas necesarios)

Se compila:

# *make* //Compilación

Como paso opcional se puede depurar el código que se género en la compilación:

# *make check* //Depuración del código que generó make

Y finalmente se instala el programa:

# *make install* //Instalación

### 1.9.6 DAR FORMATO A UNA PARTICIÓN

Para dar formato a una partición que se hace en el disco duro para datos, por ejemplo, se cuenta con el comando ***mkfs***. Este comando puede formatear cualquier partición o archivos en los formatos: bfs, cramfs, ext2, ext3, minix, reiserfs y vfat. También se puede formatear un archivo para contener los mismos datos que una partición, gracias a que en Linux, Unix y derivados se pueden montar archivos como si fueran unidades de disco.

La sintaxis para este comando es la siguiente:

# *mkfs*.formato <opción> partición

### 1.9.7 ADMINISTRACIÓN DEL ÁREA DE SWAP[6]

Si se necesita agregar más memoria para swap (intercambio) al sistema, se puede hacer de dos formas. La primera es agregando una partición de la siguiente forma:

# *mkswap* <*partición*> <tamaño>

Es decir se crea primero la partición, con fdisk, por ejemplo, después se utiliza el comando *mkswap* indicándole el nombre de la partición y el tamaño. Además se debe agregar al archivo /etc/fstab la siguiente línea:

**/dev/*nombre_partición* swap swap defaults 0 0**

La línea anterior contiene información sobre la partición.

Después, para habilitar el espacio swap se utiliza el comando:

# *swapon* <*nombre_partición*>

La segunda forma de agregar espacio para intercambio es vía un archivo; de la siguiente forma:

# *dd* if=/dev/zero of=/*nombre_archivo* bs=1024 count=*tamaño*

[6] http://perso.magic.fr/caliman/Guia_del_enROOTador-8.html

Después, dar de alta la swap:

# ***mkswap*** <*nombre_archivo*> tamaño

En el archivo /etc/fstab se agrega la línea:

***nombre_archivo*** **directorio swap defaults 0 0**

Como una observación, al utilizar esta forma, cada vez que se vaya a salir de Linux se debe de dar de baja la swap con:

# *swapoff* -a

## 1.9.8 CONFIGURACIÓN DE LA TARJETA DE RED

Para configurar la tarjeta de red se cuenta con el comando ***ifconfig***. *ifconfig* es una utilidad que permite obtener y configurar las interfaces de red del equipo. Si no se proporcionan argumentos, *ifconfig* muestra el estado de las interfaces de red que se encuentran activas. Si se proporciona una interfaz como argumento, *ifconfig* muestra el estado de dicha interfaz. Si se utiliza con la opción -a, muestra el estado de todas las interfaces, incluso aquellas que se encuentren desactivadas. Para configurar una interfaz se debe utilizar la sintaxis:

***ifconfig*** `<interfaz> <familia> <dir_ip> netmask <máscara>`
`broadcast <dir_broadcast> up`

- En *interfaz* debe proporcionarse el nombre de la interfaz de red que se desea configurar. Generalmente el nombre de interfaz se forma a partir de un nombre de manejador seguido de un número de unidad. El nombre de manejador para redes Ethernet es eth y las unidades comienzan a numerarse a partir de 0 hasta el número de interfaces existentes del mismo tipo menos uno (eth0, eth1, etc.).

- Para el parámetro *familia* se debe proporcionar el nombre de una familia de direcciones soportada por el sistema. Este nombre se utilizará para decodificar y mostrar en un formato inteligible todas las direcciones de protocolo. Las familias de direcciones más comúnmente utilizadas son inet para TCP/IP, inet6 para IP versión 6 e ipx para Novell IPX.

- *dir_ip* es la dirección IP con que se desea configurar la interfaz de red.

- *máscara* establece la máscara de red que se desea utilizar para la interfaz. Si no se proporciona este valor, se utilizarán las máscaras de red por defecto para direcciones clase A, B o C en función de la dirección IP con que se esté configurando esta interfaz.

- *dir_broadcast* al proporcionar una dirección de broadcast con la opción broadcast, se indica a la interfaz de red que se desea que habilite el modo de broadcast dirigido y que contemple dicha dirección. La dirección de broadcast dirigido a una red se determina a partir de la dirección IP de cualquiera de los equipos pertenecientes a dicha red y su máscara de red, ya que se forma a partir la dirección de red poniendo '1's en la parte de dirección correspondiente a equipos.

### 1.9.9 PROGRAMACIÓN DE TAREAS

Linux dispone de una herramienta para programar tareas. ***Crontab***. *Crontab* es una tabla que contiene los comandos que deben ser lanzados a intervalos regulares. Se pueden hacer respaldos diarios, iniciar un servicio, etc.

El formato es bastante simple:

Hacer un respaldo todos los lunes a las 02:00.

**0 2 * * 1 /home/eddie/backup**

Arrancar la máquina los días 1 y 15 de cada mes a las 4:15

**15 4 1,15 * * /sbin/shutdown<item>r +3**

Las 5 primeras columnas son:

1. Minutos (0 a 59) ;
2. Hora (0 a 23) ;
3. Día del mes (1 a 31) ;
4. Mes (1 a 12) ;
5. Día de la semana (0 a 6: 0 = Domingo, 1 = Lunes).

En seguida viene el comando. Todo usuario puede crearse un crontab gracias al comando *crontab*. Este comando tiene las siguientes opciones:

**- e** edita
**- l** lista de tareas programadas
**- d** elimina todas las tareas

# CAPÍTULO II

# LENGUAJES DE PROGRAMACIÓN Y BASES DE DATOS

## 2.1 ¿QUÉ SON LOS LENGUAJES DE PROGRAMACIÓN?

Los lenguajes de programación son un conjunto de instrucciones que al acomodarse en un orden lógico le indican a una computadora que realice una o varias tareas o acciones específicas.

## 2.2 LENGUAJES DE PROGRAMACIÓN: LENGUAJES MÁQUINA, LENGUAJES ENSAMBLADORES y LENGUAJES DE ALTO NIVEL

Los programadores escriben instrucciones en diversos lenguajes de programación, algunos de los cuales los comprende directamente la computadora, mientras en otros requieren pasos intermedios de traducción. En la actualidad se utilizan cientos de lenguajes de computación, los cuales se dividen en tres tipos generales[7]:

**1.** Lenguajes Máquina
**2.** Lenguajes ensambladores
**3.** Lenguajes de alto nivel

Cualquier computadora puede entender de manera directa sólo su propio lenguaje máquina. El lenguaje máquina es el lenguaje natural de una computadora en particular y está definido por el diseño del hardware de dicha computadora. Por lo general los lenguajes máquina consisten en cadenas de números (que finalmente se reduce a unos [1] y ceros [0]) que instruyen a las computadoras para realizar sus operaciones más elementales, una por una. Estos lenguajes son dependientes de la máquina (es decir, un lenguaje máquina en particular puede usarse solamente en un tipo de computadora). Dichos lenguajes son difíciles de comprender para los humanos.

La programación en lenguaje máquina era demasiado lenta y tediosa para la mayoría de los programadores. En vez de utilizar las cadenas de números que las computadoras podían entender directamente, los programadores empezaron a utilizar abreviaturas del inglés para representar las operaciones elementales. Estas abreviaturas formaron la base de los lenguajes ensambladores. Los programas traductores conocidos como ensambladores se desarrollaron para convertir los programas en lenguaje ensamblador a lenguaje máquina, a la velocidad de la computadora.

---

[7] DEITEL, Harvey & DEITEL, Paul. Java, Como programar. México 2004 pp. 6-7

El uso de las computadoras se incrementó rápidamente con la llegada de los lenguajes ensambladores, pero la programación en estos lenguajes aún requería de muchas instrucciones para llevar a cabo incluso las tareas más simples. Para agilizar el proceso de programación se desarrollaron los lenguajes de alto nivel, en donde podían escribirse instrucciones individuales para realizar tareas importantes. Los programas traductores, denominados compiladores, convierten programas en leguaje de alto nivel a lenguaje máquina. Los lenguajes de alto nivel permiten a los programadores escribir instrucciones que son muy similares al inglés común, y contienen la notación matemática común. Obviamente, los lenguajes de alto nivel son mucho más recomendables, desde el punto de vista del programador, que los lenguajes máquina o ensamblador. C, C++ y Java son ejemplos de lenguajes de alto nivel y además son los más poderosos y más ampliamente utilizados.

El proceso de compilación de un programa escrito en lenguaje de alto nivel a un lenguaje máquina puede tardar un tiempo considerable. Los programas intérpretes se desarrollaron para ejecutar programas en lenguaje de alto nivel directamente, sin necesidad de compilarlos en lenguaje máquina. Aunque los programas compilados se ejecutan mucho más rápido que los programas interpretados, los intérpretes son populares dentro de entorno de desarrollo de programas, en los cuales los programas se recompilan frecuentemente, a medida que se agregan nuevas características y se corrigen los errores. Una vez que se desarrolla un programa, se puede producir una versión compilada para ejecutarse con la mayor eficiencia.

## 2.3 HTML

HTML (Hyper Text Markup Lenguage) no es un lenguaje de programación como tal; pero lo que sí, es un lenguaje de marcas, estas marcas son fragmentos de texto destacado de una forma especial que permiten la definición de las distintas instrucciones de HTML, tanto los efectos a aplicar sobre el texto como las distintas estructuras del lenguaje. A estas marcas se les denomina etiquetas y son la base principal del lenguaje HTML. Un documento HTML[8] es un archivo de texto con etiquetas que variarán la forma de su presentación. El programa que sirve documentos HTML se llama servidor Web. Y la manera en que se solicitan estos documentos o también llamadas páginas es a través de un browser o navegador.

[8] Alva Arguinzoniz, Sergio. Diseño de Sitios en HTML. DGSCA UNAM

Una etiqueta será un texto incluido entre los símbolos *menor que* < y *mayor que* >. El texto incluido dentro de los símbolos será explicativo de la utilidad de la etiqueta. Por ejemplo:

| | |
|---|---|
| **<b>** | Letra Negrita, del inglés bold (negrita). |
| **<table>** | Definirá una tabla. |
| **<img>** | Inclusión de una IMaGen. |

Existe normalmente una etiqueta de inicio y otra de fin, la de fin contendrá el mismo texto que la de inicio añadiéndole al principio una diagonal /. El efecto que define la etiqueta tendrá validez para todo lo que este incluido entre las etiquetas de inicio y fin, ya sea texto plano u otras etiquetas HTML.

```
<etiqueta>Elementos Afectados por la Etiqueta</etiqueta>
```

Por ejemplo, con la etiqueta siguiente:

```
<b>Texto que será en negrita</b>.
```

Se obtiene:

**Texto que será en negrita**

Algunas etiquetas no necesitan la de fin, son aquellas en las que el final está implícito, por ejemplo **<p>** párrafo, **<br>** salto de línea ó **<img>** inclusión de una imagen. Definen un efecto que se producirá en un punto determinado sin afectar a otros elementos.

El uso de mayúsculas o minúsculas en las etiquetas es indiferente, se interpretarán del mismo modo en ambos casos. Pero para efectos de compatibilidad con nuevas tecnologías se utilizan minúsculas.

> Atributos de las etiquetas:

Las etiquetas pueden presentar modificadores que se llaman atributos que permitirán definir diferentes posibilidades de la instrucción HTML. Estos atributos se definen en la etiqueta de inicio y consisten normalmente en el nombre del atributo y el valor que toma separados por un signo de igual. El orden en que se incluyan los atributos es indiferente, no afectando al resultado.

Cuando el valor que toma el atributo tiene más de una palabra deberá expresarse entre comillas, en otro caso no será necesario.

Un ejemplo de atributo será:

```
<a href="http://www.unam.mx">Pagina principal de la UNAM</a>
```

En este caso la etiqueta A presenta un atributo HREF cuyo valor es http://www.unam.mx.

Igualmente una etiqueta podría presentar varios atributos:

```
<hr align=LEFT NOSHADE SIZE=5 WIDTH=50%>
```

En este caso la etiqueta <**hr**> presenta cuatro atributos. El segundo atributo *NOSHADE* es un caso especial que no presenta valor.

> Estructura básica

Un documento HTML está definido por una etiqueta de apertura <**html**> y una etiqueta de cierre </**html**>. Dentro de este se dividen dos partes fundamentales, la cabecera, delimitada por la etiqueta <**head**> y el cuerpo, delimitado por la etiqueta <**body**>. Por tanto la estructura de un documento HTML será:

```
<html>
      <head>
            Definiciones de la cabecera
      </head>
      <body>
            Instrucciones HTML
      </body>
</html>
```

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

Para insertar comentarios dentro de un documento HTML se utiliza la etiqueta especial **<!--**, definiéndose un comentario de la forma:

```
<!-- Esto es un comentario -->
```

> Cabecera de un documento HTML

Como se mencionó anteriormente, la cabecera de un documento HTML está delimitada por las etiquetas **<head>** y **</head>**, en esta se incluirán las definiciones generales que afectarán a todo el documento. Todas sus etiquetas son opcionales y se utilizarán solo en casos muy determinados, solo la etiqueta **<title>** tiene un uso general y aunque es opcional se recomienda incluirla en todos los documentos que se creen.

> Cuerpo de un documento HTML

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

La etiqueta <**body**> tiene diferentes atributos de entre los que se destacan:

-> *background*: Define la imagen que se utilizará de fondo del documento.
-> *bgcolor*: Indicará el color del fondo del documento.
-> *text*: Especificará el color del texto normal dentro del documento. Por defecto es negro.
-> *link*: Indicará el color que tendrán los hiperenlaces que no han sido accedidos.
-> *vlink*: Color de los enlaces que ya han sido visitados.

La utilización de los colores utiliza el siguiente formato:

**#RRGGBB**

Donde se indica en formato hexadecimal la proporción de rojo**,** verde y azul que forma el color deseado. El símbolo # es opcional. Un número hexadecimal es un número en base 16, la base normal utilizada es base 10 o decimal del 0 al 9. En este caso los número validos serán del 0 al 9 añadiendo desde la a ó A a la f ó F. Por tanto el número 0F será el 15, 0E será 14.

En la especificación del color se utiliza para definir la proporción de cada color un número del 0 al FF (255), 0 indica nada de ese color y FF la mayor proporción del color.

Algunos ejemplos de colores son:

| | |
|---|---|
| #000000 | Negro |
| #FFFFFF | Blanco |
| #FF0000 | Rojo |
| #00FF00 | Verde |
| #0000FF | Azul |

Algunos colores están predefinidos y pueden ser referenciados por su nombre, solo serán validos para Netscape e Internet Explorer de Microsoft, estos colores predefinidos son:

| **Color Predefinido** | **Color** | | **Color Predefinido** | **Color** |
|---|---|---|---|---|
| black | negro | | olive | oliva |
| teal | teal | | red | rojo |
| blue | azul | | maroon | marrón |
| navy | azul marino | | gray | gris |
| lime | lima | | fuchsia | fucsia |
| white | blanco | | green | verde |
| purple | purpura | | silver | plata |
| yellow | amarillo | | aqua | agua |

> Espaciados y saltos de línea

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

Existen unas etiquetas especiales HTML para definir estos elementos de control de texto.

**<p>** Cambio de Párrafo: Define un párrafo, se usa al comienzo o al final de un párrafo de texto e introduce un espaciado de separación (normalmente dos líneas) con el próximo texto que se exprese. Esta etiqueta se puede utilizar para introducir un espaciado entre cualquier elemento HTML y no solo servirá para separar texto.

**<br>** Salto de línea: Su utilidad es similar al anterior pero el espaciado del texto es menor, se pasa a la línea siguiente, sin dejar una línea de separación. En este caso será un cambio de línea y no de párrafo. No es necesaria la etiqueta de fin </br>.

**<hr>** Regla horizontal: Se usa para dividir un documento en distintas secciones, muestra una línea horizontal de tamaño determinable. Se asemeja al salto de página dentro de un documento.

**<pre>** Texto preformateado: Muestra una porción de texto en el que se respetan los saltos de línea, tabuladores y espacios en blanco. Todo lo que se encuentre entre las etiquetas de inicio y fin de texto preformateado se mostrará tal y como se expresa en el fuente del documento html.

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

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

> Caracteres especiales

Los caracteres acentuados y algunos caracteres especiales que usa el lenguaje HTML para definir sus etiquetas no se pueden incluir en un documento de manera normal, se deben utilizar una serie de secuencias de escape que al mostrar el documento se sustituyen por el carácter deseado.

Estas secuencias de escape comienzan todas con el símbolo ampersand (&), seguido de un texto (siempre en minúsculas) que define el carácter deseado y termina con el símbolo punto y coma (;). A continuación se muestran algunas de estas secuencias de escape:

> Elementos del lenguaje:

| Sec. Escape | Símbolo |
|---|---|
| < | Signo < (menor que) |
| > | Signo > (mayor que) |
| & | Signo & (ampersand) |
| " | Se mostrará el signo de comillas " |

> Caracteres acentuados:

| Sec. Escape | Letra | | Sec. Escape | Letra |
|---|---|---|---|---|
| á | á | | Á | Á |
| é | é | | É | É |
| í | í | | Í | Í |
| ó | ó | | Ó | Ó |
| ú | ú | | Ú | Ú |

> Algunos otros símbolos:

| Sec. Escape | Símbolo | | Sec. Escape | Símbolo |
|---|---|---|---|---|
| ç | ç | | Ç | Ç |
| ® | ® Símbolo de registrado | | © | © Símbolo de Copyright. |
| &#nnn; | Donde nnn es un número decimal, el carácter nnn del código ISO-8859-1 (ASCII). | | | |

> Listas

El lenguaje HTML proporciona un modo sencillo de representar elementos en forma de lista. Dentro de una lista se puede incluir cualquiera de los elementos HTML, e igualmente una lista puede incluirse dentro de formularios, tablas, etc.

Existen principalmente tres tipos de listas: las listas no ordenadas, las listas ordenadas y las listas de definiciones.

+ Listas no ordenadas

Este tipo de lista se usa para enumerar elementos que no tengan un orden definido. Se especifica con la etiqueta <**ul**>, todo lo que se incluya dentro de esta etiqueta y la de cierre formará la lista. Con las etiquetas <**li**> se indican cada uno de los componentes de la lista. El formato es el siguiente:

```
<ul type = DISK ó CIRCLE ó SQUARE >
        <lh> Titulo de la lista </lh>
        <li> Elemento 1
        <li> Elemento 2
        . . .
        <li> Elemento n
</ul>
```

Para marcar los distintos elementos de la lista se usarán unos símbolos, que pueden ser un disco (DISK), un círculo (CIRCLE) ó un cuadrado (SQUARE), seleccionables con el atributo TYPE.

+ Listas ordenadas

Estas listas son similares a las anteriores pero aquí se usa un número para indicar el orden de cada elemento de la lista. Al ser ordenadas no significa que ordenen los elementos alfabéticamente sino que los elementos guardan un orden que es el número que indexa la lista.

```
<ol start = n type = A ó a ó I ó i ó 0 >
        <lh> Titulo de la lista </lh>
        <li> Elemento 1
        <li> Elemento 2
                . . .
        <li> Elemento n
</ol>
```

El punto de comienzo siempre será el 1 si no se indica en start con otro valor de comienzo y el tipo de numeración puede seleccionarse con el atributo type. Sus posibles valores son:

- A: Letras mayúsculas.
- a: Letras minúsculas.
- I: Número romanos en mayúsculas.
- i: Número romanos en minúsculas.
- 0: Números (es por defecto por tanto no hay que indicarlo).

+ Listas de definiciones

En esta lista existen dos elementos la definición y el término, se usa principalmente para listas en las que se incluirán una palabra o frase y su definición (diccionario). El término aparecerá pegado en el borde izquierdo y la definición aparecerá ligeramente tabulada a partir del borde izquierdo.

Se puede usar de forma separada la definición y el término, pudiendo incluir sólo definiciones o solo términos. El incluir solo términos podría usarse para sangrar el comienzo de un párrafo.

```
<dl>
     <lh>Titulo de la lista </lh>
     <dt>Termino 1
     <dd>Definición 1
     <dt>Termino 2
     <dd>Definición 2
     . . .
     <dd>Termino N
     <dt>Definición N
</dl>
```

> Hiperenlaces

Es la utilidad básica del hipertexto, permite indicar zonas de texto o imágenes que si son seleccionados nos traslada a otros documentos HTML u otras zonas del documento actual.

Para definir un hiperenlace se puede utilizar cualquier elemento HTML, no debe ser texto necesariamente, pueden ser, cabeceras (<***hn***>), cualquiera de los estilos, una imagen, etc. Un hiperenlace igualmente podrá definirse dentro de cualquier elemento HTML: listas, párrafos de texto, tablas, formularios.

El texto del hiperenlace aparece normalmente resaltado sobre el texto normal, en azul y subrayado, en el caso de las imágenes, si tienen definido un borde éste aparecerá resaltado en color azul. La mayoría de los navegadores, cuando la zona por la que pasa el cursor es sensible, éste cambia de aspecto indicándolo y en la parte baja de la pantalla se indica el hiperenlace que se activa al pulsar en esa zona. Mediante los atributos de <**body**> es posible cambiar el color de los hiperenlaces.

Se utilizan dos tipos de etiquetas para los hiperenlaces:

<**a** href.>: Son los enlaces con documentos externos al actual. En este caso se indicará una URL que definirá el documento al que se accede si se sigue el enlace. La forma de indicarlo es:

```
<a href="URL a la que se accede">Texto del Hiperenlace</a>
```

Para hacer que una imagen sea un hiperenlace:

```
<a href="URL a la que se accede"><img src="Imagen"> y también texto</a>
```

<**a** name>: Sirve para indicar puntos de un documento que son accesibles directamente. Marcará las distintas zonas de un documento. La forma de indicarlo es:

```
<a name="Id. del ancla">Texto del ancla</a>
```

A cada ancla se le dará un nombre distinto que será el que se utilice luego para referenciarlo. El texto que define a la etiqueta no tendrá ningún tipo de resalte, y solo se utiliza como punto de destino del hiperenlace.

La forma de especificar un enlace que acceda a un punto determinado del mismo documento es:

```
<a href="#Id. del ancla">Texto del enlace al ancla</a>
```

Será un enlace a la zona del documento actual que se había marcado con la etiqueta indicada. También es posible acceder a un ancla de un documento externo de la forma:

```
<a href="Dirección URL#Id. del ancla">Texto del enlace al ancla</a>
```

De esta forma se podrá acceder directamente a puntos determinados o secciones de un documento. La utilidad principal es la creación de índices en documentos largos, al comienzo del documento se especifica el índice con enlaces a las distintas anclas y dentro del documento se indica el comienzo de cada apartado con el ancla que lo define.

> Tablas

Permite la representación de datos por filas y columnas, en forma tabular. La definición es muy flexible indicando sólo los elementos que forman cada fila y columna, calculándose de forma automática el tamaño que deben tener las celdas. En una tabla se pueden introducir todo tipo de elementos como imágenes, enlaces, texto, listas, cabeceras, formularios, etc.

No es necesario definir inicialmente el número de filas o columnas ya que éstas se calculan según se va definiendo la tabla. En el caso que una fila tenga más columnas que otra, en las otras filas las columnas se representarán vacías, no siendo necesario que todas las filas sean iguales.

Para definir una tabla se usa la etiqueta <**table**>, que tiene el siguiente formato:

```
<table border="tamaño del borde" >
        ... Definición de la tabla
</table>
```

En la etiqueta inicial <**table**> se definen los atributos que afectarán a toda la tabla, todos los atributos son opcionales. Todo lo que se incluya dentro de la instrucción de tabla se considerará como tal, pudiendo definir tablas anidadas (tablas dentro de tablas).

Algunos de los atributos que puede usarse para afectar la tabla son:

=> *Border*: Si se especifica se dibujará un borde alrededor de la tabla y separando los distintos campos que presenta. Se indica un número que especificará el tamaño del borde, por defecto será 0, es decir, no se dibujará ningún borde. Aunque no se dibuje el borde sí se mantendrá el espaciado de los elementos de la tabla.

=> *Cellspacing*: Indica el espacio que debe existir entre las distintas celdas de la tabla. Por defecto será 2. Si se indica 0 no habrá ningún espacio entre las celdas.

=> *Cellpadding*: Es la cantidad de espacio entre el borde de la celda y el contenido de ésta, por defecto es 1. Si se indica 0 las celdas aparecerán sin separación.

=> *Width*: Será el ancho de la tabla, se puede indicar como valor absoluto o como porcentaje del ancho del documento (tamaño de la ventana del visualizador). Lo recomendable es utilizar tamaño en porcentaje, ya que de esta forma la tabla queda perfectamente ajustada al área de visualización del documento.

=> *Height*: Definirá el alto de la tabla, a igual que Width, se puede indicar en valor absoluto o en porcentaje. En este caso es más recomendado en valor absoluto ya que el alto es más difícil que pueda coincidir con el tamaño de la ventana.

La etiqueta <**caption**> especifica el título de la tabla, se mostrará en la parte superior externa de la tabla.

<**tr**>, <**th**> y <**td**> son etiquetas que forman parte de la definición interna de la tabla. La etiqueta <**tr**> define cada una de las filas de la tabla y especifica los parámetros que afectarán a todas las celdas de la fila; no es necesario indicar etiqueta de cierre. Algunos atributos de esta etiqueta son:

=> *Align*: Indica la alineación del elemento dentro de la celda: izquierda, derecha o centro.

=> *Valing*: Indica la alineación vertical del dato dentro de la celda: superior, media o baja.

=> *Bgcolor*: Indica el color de fondo que tendrán todas las celdas de la fila.

Las etiquetas <**th**> y <**td**> definen cada una de las celdas de una fila, la diferencia consiste en que <**th**> se usa para definir una celda de tipo cabecera y <**td**> para definir una celda de datos.

Estas etiquetas, además de tener atributos como los de la etiqueta anterior, que afectan a toda la fila, estas últimas sólo afectan a cada celda, y tienen otros atributos:

=> *Rowspan*: Indica el número de filas que ocupará está celda en la misma fila. En este caso la celda se expandirá ocupando tantas celdas de la tabla inicial como se especifique.

=> *Colspan*: Indica el número de columnas que ocupará la celda actual, se obtendrá una celda que ocupa varias columnas.

=> *Nowrap*: Se usa para que no se divida la línea por defecto. Las líneas de texto no se dividirán dentro de la celda en varias líneas. Por tanto si la línea es muy larga la columna de la tabla será tan ancha como la línea, solo se dividirá si se usa el elemento

> Formularios

Los formularios son plantillas que permiten la creación de documentos HTML con peticiones de datos. La principal utilidad de los formularios es la posibilidad de crear cuestionarios, encuestas, páginas de comentarios o cualquier documento en la que se desee una interacción por parte del usuario.

Se pueden definir distintos tipos de recuadros de dialogo, botones de selección, menús de múltiples opciones, etc. Para permitir obtener los datos de una manera más intuitiva.

Existe una etiqueta ara la creación de formularios esta es <**form**>, que tiene la siguiente estructura:

```
<form action="archivo que trata el formulario" method= POST | GET>
        ...
        Elementos que forman el formulario
        ...
</form>
```

Dentro de la etiqueta de formulario se definen los distintos elementos de petición de datos. Estas instrucciones definen los tipos de botones, cajas de dialogo y ventanas para la introducción de datos. Y además definen las variables que almacenarán los datos introducidos por el usuario. Estas etiquetas se incluyen entre la de definición del formulario y la etiqueta de final de formulario.

Los siguientes atributos modifican el comportamiento de la etiqueta <**form**>:

=> *Action*: Indica el programa que se encargará de tratar los datos del formulario. Este programa debe encontrarse en el servidor y estar escrito en algún lenguaje de programación. A este programa se pasará como parámetros los datos introducidos en el formulario y regresará un código HTML que se mostrará tras procesar el formulario.

=> *Method*: Indica el protocolo usado para el envío de los datos. Con POST se envían los datos en la entrada estándar del programa que trata el formulario y con GET los datos se pasan por parámetro, en la línea de comandos, al programa. El usar uno u otro método vendrá determinado por cómo son tratados los parámetros del formulario en el programa.

Una vez definidas las características globales del formulario se incluyen los distintos botones y cajas de diálogo que lo van a constituir. Dentro de la instrucción del formulario podrán incluirse cualquier texto o instrucción HTML, siendo recomendado a fin de poder etiquetar las opciones de entrada y especificar cualquier dato importante relacionado con el formulario. Igualmente un formulario puede ser incluido en algunas instrucciones como las listas, tablas, etc.

> Entrada de datos

La etiqueta <**input**> se utiliza para definir gran variedad de tipos de campos de entrada de datos. El formato básico es el siguiente:

```
<input type = ( text | password | checkbox | radio | hidden | submit |
 image | reset ) name = "Variable que toma el valor" value = "Valor de
 Inicialización">
```

El atributo type se usa para determinar el tipo de recuadro de dialogo de entrada que se está definiendo. El atributo *name* especifica el nombre de la variable que se define. Este nombre será pasado al programa que trata el formulario junto con el valor que le asigno el usuario del formulario. El atributo value suele especificar el valor de inicialización, que será el valor por defecto.

> Tipos de entrada:

* **Text**: Se utiliza para la entrada de cadenas de texto corto, como por ejemplo nombre de personas, números, fechas o diversos datos que se puedan expresar en una línea de texto.

Nombre:

* **Password**: Es similar al anterior pero en este caso no se imprimen los caracteres según se van introduciendo, se muestra un asterisco en vez de los caracteres. Solo se puede ver el número de caracteres, pero no valor.

Password: **********

* **Checkbox**: El checkbox es un botón que puede presentar dos estados activado o desactivado.

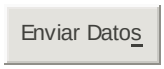
- [ ] Opción
- [x] Opción

* **Radio**: Se usa cuando la opción puede tomar un valor simple de una serie de alternativas. En este caso se presentan varios valores opcionales de los que sólo puede tomar un valor. Para especificar cada uno de estos valores se incluye una etiqueta radio por cada una de las posibles alternativas.

- ( ) Opción 1
- (•) Opción 2
- ( ) Opción 3
- ( ) Opción 4

* **Hidden**: En este caso no se muestra ningún campo para la entrada de datos al usuario, pero el par variable/valor especificado es enviado junto con el formulario. Se suele usar para transmitir información de estado ó control ó para enviar algún tipo de información que no debe ser variada en el formulario, pero sí debe ser enviada junto a este.

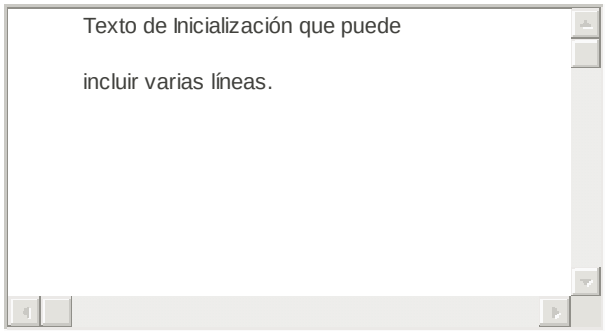
* **Submit**: Este botón se usa para enviar los datos del formulario, al pulsar el usuario este botón, se acaba la introducción del formulario y pasa el control al programa indicado en *action*. En todo formulario debe existir al menos un botón de submit, si solo incluye un recuadro del tipo text no será necesario incluirlo.




* **Image**: Su funcionalidad es similar al botón de submit, se usa igualmente para enviar los datos de un formulario, pero en este caso se presenta una imagen como botón.

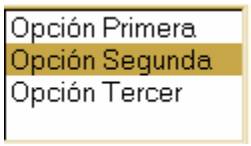
* **Reset**: Este botón se usa para volver a los valores por defecto todos los elementos del formulario, borrando todos los datos introducidos por el usuario.

Otro tipo de campo de datos que se utiliza para introducción de texto que se define con su propia etiqueta es <**textarea**>; permite la introducción de un texto que puede abarcar varias líneas, introduciendo éste en forma de párrafo.




Como se puede ver en este ejemplo, en la definición de este tipo de campo se puede especificar un texto de inicialización, al igual se puede definir con los campos text.

La etiqueta <**select**> se usa para menús simples o múltiples. Define menús de tipo pop-up (menú de barras) y ofrece una alternativa más compacta al uso de botones radio o checkbox.

> Frames

Con las frames es posible dividir la ventana del navegador en varias subregiones (*frames*), permitiendo mostrar una URL distinta en cada una de ellas. En cada frame se permite:

- Mostar su propia URL, diferenciada del resto de las frames de la pantalla, de esta forma un hiperenlace puede tener como destino un documento y la frame en el que se mostrará.

- Tendrán asociado un nombre, que las distinguirán del resto de las frames de la pantalla y permitirá usarlo en los hiperenlaces.

- En el caso que se cambie el tamaño de la ventana, se podrá determinar si la frame se ajusta a este tamaño o mantiene su tamaño intacto.

Esto permite crear nuevos tipos de documentos, en los que por ejemplo se mantendrá una región estática (lista de enlaces, barra de botones, formulario) y otra zona dinámica en la que se mostrará el resultado. De esta forma una de las principales utilidades de las frames es la creación de páginas con un índice (por ejemplo un manual) o una cabecera estática, consiguiendo así una mejora de la navegación al poder acceder al índice de una manera más rápida y efectiva.

Un documento con frames se define de manera diferente a un documento normal, siendo la estructura del documento distinta, en este caso no se define la etiqueta <**body**>. Su estructura es la siguiente:

```
<html>
      <head>
            Definiciones de la cabecera
      </head>
      <frameset>
            Definición de las frames que forman el documento y de los
            archivos que incluye cada una.
      </frameset>
      <noframes>
            Instrucciones que se mostrará en los navegadores que no
            soporten frames.
      </noframes>
</html>
```

Dentro de la etiqueta <**noframes**> se podrá incluir una explicación de que el documento sólo es visible con los navegadores que soporten frames.

<**frameset**> Con esta etiqueta se definen las frames que forman el documento, su sintaxis es similar a la de las tablas, permitiendo definir muy distintos tipos de frames. Su formato es el siguiente:

```
<frameset rows=Lista de las Filas cols=Lista de las Columnas>
      <frame src=url_1 name="Nombre de la frame1">
      <frame src=url_2 name="Nombre de la frame2">
      . . .
      <frame src=url_N name="Nombre de la frameN">
</frameset>
```

Se define solo uno de los atributos, o la lista de filas (rows) o la lista de columnas (cols).

=> **Rows**: Se define separado por comas el tamaño de cada una de las frames. De esta manera se dividirá la pantalla de forma horizontal, según cada una de las filas definidas. El tamaño de la frame, puede expresarse de las siguientes formas:

- En valor absoluto, que indicará el tamaño en puntos de la pantalla. En este caso si todas las frames se indican de este modo, los valores se ajustarán para que las frames ocupen la totalidad del espacio de la ventana del navegador, no guardando siempre la proporción con la que se definen las frames.

- En tanto por ciento sobre el tamaño de la ventana, en este caso si los porcentajes suman un valor distinto del 100%, se ajustarán para que coincidan con el tamaño de la ventana. Se podrá combinar con el apartado anterior de forma que algunas frames se definan en valor absoluto y otras en porcentaje.

- De forma relativa con el símbolo * que indica el tamaño restante de la ventana. Si se indica una frame como 2* y otra como *, la primera ocupará dos tercios del espacio restante y la segunda un tercio del espacio restante. Se puede combinar con las definiciones anteriores.

=> **Cols**: Toma los mismos posibles valores que rows, pero en este caso para las columnas, se definirán las frames de forma vertical.

Una vez definida el número de frames por fila o por columna se define el contenido de cada una de estas con la etiqueta <frame>, pero igualmente podrían definirse frames dentro de frames, de forma que cada una de las definidas anteriormente podría estar divida en varias frames, esto se hará incluyendo dentro de la instrucción <**frameset**>, nuevas instrucciones <**frameset**> que dividirán esta en las frames indicadas,

La etiqueta <**frame**> tiene los siguientes atributos:

=> *URL*: Especifica el documento HTML o archivo que se mostrará en la frame definida. Si no se especifica documento alguno se mostrará la frame vacía.

=> *NAME*: Indica el nombre de la frame, este nombre es importante ya que se usará en los hiperenlaces (normalmente en los documentos de las otras frames) para indicar la frame de destino del documento. Si no se indica el nombre sólo se podrá mostrar el documento actual, sin que sea posible cambiarlo mediante hiperenlaces.

=> *SCROLLING*: Indica si la frame tendrá o no una barra de scroll. Si toma el valor "yes", siempre se mostrará esta barra, para el valor "auto" sólo se mostrará si el documento no cabe en la frame, si es necesaria. Y por último "no" indica que en ningún caso se muestre la barra de scroll. Si no se indica nada se toma por defecto el valor auto.

=> *NORESIZE*: Indica que la frame no debe ser variada de tamaño por el usuario. Por defecto todas las frames pueden variar su tamaño.

Por último existe un atributo que se puede utilizar en la definición de frames, este atributo es "**target**", que indicará la frame de destino de un hiperenlace, por ejemplo. Normalmente, en páginas sin frames, cuando se sigue un hiperenlace, éste se muestra en la ventana del navegador sustituyendo el documento actual, con las frames se puede especificar que frame será la de destino, no siendo necesario que sea la frame del documento actual. Como nombre de la frame su usará el nombre que se especificó en el atributo *name* de la etiqueta <**frame**>

Existen unos valores especiales de TARGET que nos permitirán definir destinos distintos a las frames definidas. Estos valores son los siguientes:

- target="_blank": Indica que se muestre en una nueva ventana del navegador.
- target="_self": Se mostrará en la misma ventana o frame que lo referencia.
- target="_parent": Se muestra en la frame o estructura de frames que llamó al documento actual.
- target="_top": Indica que se muestre en la ventana completa, eliminando la estructura de frames que tenga la ventana.

.

## 2.4 EL LENGUAJE DE PROGRAMACIÓN PHP

### ¿Qué es PHP?

PHP[9] (acrónimo de "PHP: Hypertext Preprocessor") es un lenguaje de "código abierto" interpretado, de alto nivel, embebido en páginas HTML (Hyper Text Markup Lenguage) y ejecutado en el servidor. Para comprender esto véase el siguiente ejemplo:

```
<html>
    <head>
        <title>Ejemplo</title>
    </head>
    <body>

        <?php
        echo "Hola, &iexcl;soy un script PHP!";
        ?>

    </body>
</html>
```

[9]SÆTHER BAKKEN, Stig. PHP Manual. PHP Documentation Group 2004

Puede apreciarse que no es lo mismo que un script escrito en otro lenguaje de programación como Perl o C; En vez de escribir un programa con muchos comandos para crear una salida en HTML, escribimos el código HTML con cierto código PHP embebido (incluido) en el mismo, que producirá cierta salida (en el ejemplo, producirá un texto). El código PHP se incluye entre etiquetas especiales de comienzo y final que permiten entrar y salir del modo PHP.

Lo que distingue a PHP de tecnologías como Javascript, la cual se ejecuta en la máquina cliente, es que el código PHP es ejecutado en el servidor. Si se tuviera un script similar al del ejemplo en un servidor, el cliente solamente recibiría el resultado de su ejecución en el servidor, sin ninguna posibilidad de determinar qué código ha producido el resultado recibido. El servidor Web puede ser incluso configurado para que procese todos los archivos HTML con PHP.

Lo mejor de usar PHP es que es extremadamente simple para el principiante, pero a su vez, ofrece muchas características avanzadas para los programadores profesionales. Aunque el desarrollo de PHP está concentrado en la programación de scripts en el lado del servidor, se puede utilizar para muchas otras cosas.

## ¿Qué se puede hacer con PHP?

PHP puede hacer cualquier cosa que se pueda hacer con un script CGI, como procesar la información de formularios, generar páginas con contenidos dinámicos, o enviar y recibir cookies. Y esto no es todo; existen tres campos en los que se usan scripts escritos en PHP.

- Scripts del lado del servidor. Este es el campo más tradicional y el principal foco de trabajo. Se necesitan tres cosas para que esto funcione. El intérprete PHP (CGI ó módulo), un servidor Web y un navegador. Es necesario correr el servidor Web con PHP instalado. El resultado del programa PHP se puede obtener a través del navegador, conectándose con el servidor Web.

- Scripts en la línea de comandos. Se puede crear un script PHP y correrlo sin ningún servidor Web o navegador. Solamente se necesita el intérprete PHP para usarlo de esta manera. Este tipo de uso es ideal para scripts ejecutados regularmente desde cron (en *nix o Linux) o el Planificador de tareas (en Windows). Estos scripts también pueden ser usados para tareas simples de procesamiento de texto.

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

De modo que, con PHP se tiene la libertad de elegir el sistema operativo y el servidor. También tiene la posibilidad de usar programación procedimental o programación orientada a objetos. Aunque no todas las características estándares de la programación orientada a objetos están implementadas en la versión actual de PHP, muchas librerías y aplicaciones grandes (incluyendo la librería PEAR) están escritas íntegramente usando programación orientada a objetos.

Quizás la característica más potente y destacable de PHP es su soporte para una gran cantidad de bases de datos. Escribir un interfaz vía Web para una base de datos es una tarea simple con PHP. Las siguientes bases de datos están soportadas actualmente:

| | | |
|---|---|---|
| Adabas D | Ingres | Oracle (OCI7 and OCI8) |
| dBase | InterBase | Ovrimos |
| Empress | FrontBase | PostgreSQL |
| FilePro (read-only) | mSQL | Solid |
| Hyperwave | Direct MS-SQL | Sybase |
| IBM DB2 | MySQL | Velocis |
| Informix | ODBC | Unix dbm |

También se cuenta con una extensión DBX de abstracción de base de datos que permite usar de forma transparente cualquier base de datos soportada por la extensión. Adicionalmente, PHP soporta ODBC (el Estándar Abierto de Conexión con Bases de Datos), así que puede conectarse a cualquier base de datos que soporte tal estándar.

PHP también cuenta con soporte para comunicarse con otros servicios usando protocolos tales como LDAP, IMAP, SNMP, NNTP, POP3, HTTP, COM (en Windows) y muchos otros. También se pueden crear sockets puros. PHP soporta WDDX para el intercambio de datos entre lenguajes de programación en Web. Y hablando de interconexión, PHP puede utilizar objetos Java de forma transparente como objetos PHP Y la extensión de CORBA puede ser utilizada para acceder a objetos remotos.

Si se usa PHP en el campo del comercio electrónico, se encontrarán muy útiles las funciones Cybercash, CyberMUT, VeriSign Payflow Pro y CCVS para programas de pago.

## Instalación de PHP

La instalación de PHP se lleva a cabo instalando el módulo de PHP en el servidor Web Apache. Para instalar el servidor Apache con soporte para PHP se llevan a cabo los siguientes pasos:

**1**) Primeramente se obtiene PHP de la página http://www.php.net

**2**) Se descomprime con el comando: # tar -zvxf php-x.x.x.tar.gz. Al descomprimir este archivo se genera una carpeta, se ingresa a la carpeta: # *cd* php-x.x.x

**3**) Después se empieza con la instalación de Apache:

# ./*configure* --prefix=/usr/local/php --with-apxs2=/usr/local/apache2/bin/apxs

# *make*

**4**) Una vez instalado el modulo de PHP la siguiente línea debe de estar en el archivo /usr/local/apache2/modules:

libphp4.so //módulo de PHP

En el archivo de configuración de Apache (httpd.conf) debe de aparecer la siguiente línea:

LoadModule php4_module module/libphp4.so

Y se debe agregar la siguiente línea en el mismo archivo:

AddType application/x-httpd-php .php .phtml

**5**) Para comprobar si la instalación fue exitosa se checa con el siguiente script:

```
<? php
      print “Hello Everybody”;
?>
```

Se guarda este script en donde se guardan los archivos del servidor (en este caso /usr/local/apache2/htdocs) y de ejecuta desde el navegador de preferencia de la siguiente manera:

http://localhost/script.php o http://127.0.0.1/script.php

## Sintaxis básica

*Escapar de HTML*:

Para interpretar un archivo, php simplemente interpreta el texto del archivo hasta que encuentra uno de los caracteres especiales que delimitan el inicio de código PHP. El intérprete ejecuta entonces todo el código que encuentra, hasta que encuentra una etiqueta de fin de código, que le dice al intérprete que siga ignorando el código siguiente. Este mecanismo permite embeber código PHP dentro de HTML: todo lo que está fuera de las etiquetas PHP se deja tal como está, mientras que el resto se interpreta como código.

Hay cuatro conjuntos de etiquetas que pueden ser usadas para denotar bloques de código PHP. De estas cuatro, sólo 2 (`<?php. . .?>` y `<script language="php">. . .</script>`) están siempre disponibles; el resto pueden ser configuradas en el archivo de `php.ini` para ser o no aceptadas por el intérprete. Mientras que el formato corto de etiquetas (short-form tags) y el estilo ASP (ASP-style tags) pueden ser convenientes, no son portables como la versión de formato largo de etiquetas. Además, si se pretende embeber código PHP en XML o XHTML, será obligatorio el uso del formato <?php. . .?> para la compatibilidad con XML.

Las etiquetas soportadas por PHP son:

```
1.  <?php echo("si quieres servir documentos XHTML o XML, haz como
aqu&iacute;\n"); ?>

2.  <? echo ("esta es la m&aacute;s simple, una instrucci&oacute;n de
procesado SGML \n"); ?>
    <?= expression ?> Esto es una abreviatura de "<? echo expression ?>"

3.  <script language="php">
        echo ("muchos editores (como FrontPage) no
              aceptan instrucciones de procesado");
    </script>

4.  <% echo ("Opcionalmente, puedes usar las etiquetas ASP"); %>
    <%= $variable; # Esto es una abreviatura de "<% echo . . ." %>
```

La forma en la que se separan las distintas sentencias es mediante la utilización de “;”. En PHP cada sentencia debe finalizar con “**;**”.

*Comentarios*:

En PHP hay 3 formas distintas de incluir comentarios:

```
/* Al estilo de C
   en donde el comentario empieza y termina
   delimitado por barra asterisco y asterisco barra
*/
```

O bien usando

```
// Comentario
```

O por último

```
# Comentario
```

En las dos últimas variantes el comentario empieza en donde se encuentra el "//" o el "#" y termina cuando termina la línea; conocidos como comentarios de fin de línea.

*Tipos de Datos*:

PHP soporta los siguientes tipos de datos:

· **Enteros**
· **Binarios de punto flotante**
· **Strings**
· **Arreglos**
· **Objetos**

En general el tipo de dato de una variable no es decidido por el programador sino que lo decide el lenguaje en tiempo de ejecución, la instrucción *settype* puede usarse para forzar el tipo de dato de una variable en los raros casos en que esto sea necesario. Todas las variables en php se denotan utilizando el signo '**$**' precediendo al nombre de la variable.

· Enteros:

```
$a = 1234; # número decimal
$a = -123; # número negativo
$a = 0123; # número octal (83 en decimal)
$a = 0x12; # número en hexadecimal (18 decimal)
```

· Flotantes:

Los números de punto flotante pueden notarse de la siguiente manera:

```
$a = 1.234;
$a = 1.2e3;
```

· Strings:

En PHP los strings tienen un manejo similar al utilizado en C o C++, están predefinidos los siguientes caracteres especiales:

| |
|---|
| **\n** Nueva línea<br>**\r** Salto de carro (carring return)<br>**\t** Tabulación<br>**** Barra invertida<br>**\$** Signo pesos<br>**\"** Comillas doble |

Un string puede inicializarse usando comillas dobles o comillas simples. Cuando se utilizan comillas dobles el intérprete de PHP parsea previamente el string, es decir que reemplaza los nombres de variables que encuentra en el string por el contenido de la variable. Cuando se usan comillas simples el string se imprime tal y como figura sin ser parseado.

*Ejemplo*:

```
$x="Juan";
$s="Hola $x";
$t='Hola $x'
```

$s vale "Hola Juan" y $t vale "Hola $x".

Otra forma de inicializar un string es usando un string multilínea de la siguiente manera:

```
$str=<<<EOD
        Este es un ejemplo de un string
        que ocupa varias líneas y se puede
        definir así
EOD;
```

Se pueden concatenar strings usando el operador "**.**" de la siguiente manera:

```
$x=”hola”;
$y=”mundo”;
$s=$x.$y;          #$s es igual a “holamundo”.
$s=$x.” “.$y;      # Aquí $s vale “hola mundo”
```

· Arreglos:

Los arreglos en PHP actúan tanto como arreglos tradicionales (indizados por número) así también como arreglos asociativos (indizados por clave).

Los arreglos pueden crearse usando las instrucciones "list" o "array" o bien inicializando en forma explícita cada elemento del arreglo.

```
$a[0]=”hola”
$a[1]=”mundo;
$a[“clave”]=”valor”;
```

Utilizando la notación especial $v[]; se pueden agregar elementos a un arreglo.

```
$a[0]=”nada”;
$a[1]=”hola”;
$a[]=”mundo”;      #Asigna a $a[2] el valor “mundo”.
```

Existen funciones especiales para ordenar arreglos, contar la cantidad de elementos de los mismos, recorrerlos, etc.

*Matrices o Arreglos de 2Dimensiones:*

La definición, inicialización y uso de matrices en PHP es sencilla. Se puede pensar una matriz en PHP como un arreglo de arreglos, por lo que se puede utilizar la misma lógica que en los primeros.

```
$a[0][1]=”Hola”;
$a[0][“clave”]=”una cosa”;
$a[“clave1”][“clave2”][0][1]=”otra cosa”;
etc…
```

Para incluir el valor de un elemento de un arreglo en un string se deben usar llaves para indicar el alcance del nombre de la variable a reemplazar:

```
echo “Esta es una prueba {$a[0][1]}”;
```

Una forma útil de inicializar un arreglo asociativo es usando la siguiente notación:

```
$a=array(
      “color”  => “rojo”,
      “edad”   =>  23,
      “nombre” => “juan”
);
```

Para crear una matriz se pueden anidar las declaraciones de tipo array.

```
$a = array(
      "apple" => array(
            "color" => "red",
            "taste" => "sweet",
            "shape" => "round"
      ),
      "orange" => array(
            "color" => "orange",
            "taste" => "tart",
            "shape" => "round"
      ),
      "banana" => array(
            "color" => "yellow",
            "taste" => "paste-y",
            "shape" => "banana-shaped"
      )
);
```

*Constantes:*

Para definir una constante se utiliza la instrucción "**define**" de la forma:

```
define("PI",3.14151692);
```

Luego las constantes pueden usarse como variables tradicionales ($PI) con la salvedad de que no se les puede asignar un valor.

*Operadores:*

Los operadores aritméticos en PHP también se asemejan a C:

```
$a + $b;      //suma
$a - $b;      //resta
$a++;         //pos-incremento, esta sentencia devuelve el valor de $a y lo incrementa en 1.
++$a;         //pre-incremento, incrementa en 1 el valor de $a y devuelve el valor incrementado.
$a--;         //pos-decremento
--$a;         //pre-decremento
$a * $b;      //multiplicación
$a / $b;      //división
$a % $b;      //módulo
```

- Asignación:

La asignación se resuelve con el signo igual ("=").

```
$a=5;          //Asigna 5 a $a
$a=$b;         //Asigna el valor de $b a $a
$b=($c=6);     //Asigna 6 a $c y a $b
```

Y pueden combinarse asignación y operadores de la forma:

```
$a+=5;          //Suma y asigna 5 a $a
$x.="hola";  //Concatena $x con "hola"
```

- Operaciones con bits:

```
$a & $b;      //$a AND $b
$a | $b;      //$a OR $b
$a ^ $b;      //$a XOR $b
~$a;          //NOT $a
$a << $b;     //Shift hacia la izquierda $b posiciones
$a >> $b;     //Shift hacia la derecha $b posiciones
```

- Comparaciones:

```
$a == $b;     //true si $a igual a $b
$a === $b;    //true si $a igual a $b y además son del mismo tipo
$a >= $b;     //mayor o igual
$a <= $b;     //menor o igual
$a != $b;     //true si a y b son distintos
```

- Operador @

Cuando se antepone @ a una expresión se suprimen los errores que la expresión pudiera generar.

*Ejemplo*:

```
@($c=$a/$b);
```

- Operador de ejecución:

PHP soporta el uso de “backticks” (comillas invertidas) para ejecutar un comando desde el shell y devolver el resultado de la ejecución del comando en una variable:

```
$result=`ls –l`;
```

- Operadores lógicos:

```
$a && $b;      //True si $a es true y $b es true
$a || $b;      //True si $a es true o $b es true
$a xor $b;     //OR exclusivo
!$a;           //True si $a es falso (NOT)
```

*Estructuras de Control*:

Las estructuras de control son sentencias que modifican el comportamiento de cierto programa, a través de evaluar una condición; en algunas estructuras (if, if…else) si la condición a evaluar es verdadera se ejecuta el bloque siguiente de instrucciones, si no es verdadera se ejecuta otro de bloque de instrucciones o simplemente no se ejecuta el primer bloque. En otras estructuras (while, for, do…while) la condición sirve para ciclar un bloque de instrucciones para que se ejecute un número de veces deseado. La sintaxis de las estructuras de control se muestra a continuación:

***if:***

```
if (expresión) {sentencias;}

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

***while:***

```
while (expresión) {
      sentencias;
}

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

***for:***

```
for (expr1,expr2,expr3) {
      sentencias;
}
```

La primera expresión en el ciclo *for* cumple la función de inicializar las variables de control del FOR. Esta expresión se cumple incondicionalmente, más allá de que se entre dentro del ciclo o no. La expresión 2 se evalúa siempre que se esté por ingresar al ciclo del FOR, aún cuando se ingresa al *for* por primera vez.

La tercera expresión se ejecuta cada vez que se termina el ciclo. Por lo general se utiliza esta expresión para indicar el incremento de alguna variable que se este utilizando para el FOR. La ejecución de esta expresión es también incondicional y es la que se ejecuta inmediatamente antes de evaluarse la expresión 2.

*Ejemplo*:

```
for ($i=0; $i <5; $i++) {
      print(“$i”);
}
```

***Foreach:***

Para realizar ciclos con la cantidad de ocurrencias en un arreglo se utiliza la estructura *foreach*:

```
foreach ($vector as $variable) {
      sentencias;
}

foreach ($vector as $clave => $valor) {
sentencias;
}
```

Por cada iteración cada elemento de $vector es asignado a $variable.

El segundo caso es aplicable para recorrer vectores asociativos.

***Break:***

Break permite salir del ciclo actual "*for*", "*while*" o "*switch*"

*Ejemplo*:

```
for ($i=0; $i < 6; $i++) {
    if($i==$b) break;  //Sale del ciclo si $i es igual a $b
}
```

***Switch:***

La sentencia *switch* permite ejecutar un grupo de sentencias de acuerdo al valor de una variable:

```
switch ($variable) {
      case valor:
            sentencias;
      break;
      case valor2:
            sentencias;
      break;
      default:
            sentencias;
      break; }
```

Cuando el valor de la variable ($variable) coincide con el valor de algún "**case**", se ejecutan las sentencias que se encuentran a continuación.

En este caso se utiliza la sentencia "**break**" en forma prácticamente obligatoria, porque en caso de no existir esta sentencia se seguiría ejecutando linealmente todas las sentencias continuas, es decir las sentencias de los demás "cases" inferiores.

Por último, la opción "**default**" se utiliza generalmente para cuando el valor de la variable no coincide con ningún "case". Estas sentencias se ejecutan siempre, salvo en el caso de que se ejecute antes un "**break**".

*Funciones*:

Definir subrutinas (funciones) en PHP es sencillo:

```
function prueba($a, $b) {
      $r=$a + $b;
      return $r;
}
```

Para invocar a la función basta con hacer `$x = prueba(4,6);`

Los parámetros que recibe la función pueden ser enteros, flotantes, strings, arreglos u objetos es decir cualquiera de los tipos de datos soportado por PHP. El valor devuelto por la función también puede ser cualquier tipo de datos de PHP.

*Parámetros default:*

Es posible asignar un valor default a los parámetros que recibe una función de forma tal que cuando se invoca la función si se ignora el parámetro el mismo es asignado al default.

```
function prueba($a=2, $b=3, $c=5) {
      //código
}
```

Si se llama prueba(4)  // Entonces $a=4, $b=3 y $c=5

Se puede saber cuantos parámetros recibió una función usando **func_num_args**() y se puede obtener el i-esimo parámetro de una función con **func_get_arg**(número_de_parámetro);

Finalmente los nombres de funciones pueden guardarse en variables e invocarse las mismas usando el nombre guardado en una variable.

*Ejemplo*:

```
$nombre = sumar;
$nombre(4,5);              //Llama a la función sumar
```

Esto permite guardar nombres de funciones en tablas, archivos, arreglos etc. lo cual da lugar a ciertas maniobras interesantes.

## 2.5 CASOS DE USO: APLICACIONES EN PHP

Como se vio en el punto anterior, PHP incluye muchas funcionalidades estándar de la mayoría de los lenguajes de alto nivel. A continuación se presentan algunas aplicaciones que se desarrollaron con este lenguaje, para tener más claro el poder de programación que proporciona a los programadores.

### Calculadora

```
<?php

        $op1=$_GET["op1"];
        $op2=$_GET["op2"];
        $calc=$_GET["calc"];

      if ($calc == "suma")
        print "La suma de $op1 + $op2 es ".($op1+$op2);

      if ($calc == "resta")
        print "La resta de $op1 - $op2 es ".($op1-$op2);

      if ($calc == "multi")
        print "La multiplicación de $op1 * $op2 es ".($op1*$op2);

      if ($calc == "div")
        print "La división de $op1 / $op2 es ".($op1/$op2);

?>
```

Este código obtiene los operandos necesarios para realizar una de cuatro operaciones: suma, resta, multiplicación y división; que se seleccionan a través de estructuras de control *if*. Dichos operandos se obtienen del usuario a través de un formulario en una página HTML;

El código del formulario se muestra a continuación:

```
<form action="oper_calc.php">

      <br><br>

      Oper1: <input type="text" name="op1">
      <br><br>

      Oper2: <input type="text" name="op2">
      <br><br>

      <input type="radio" name="calc" Value="suma">SUMA
      <input type="radio" name="calc" Value="resta">RESTA
      <input type="radio" name="calc" Value="multi">MULTIPLICACION
      <input type="radio" name="calc" Value="div">DIVISION

      <br>

      <input type="Submit" value="Do it!">
</form>
```

El siguiente programa calcula el **factorial de un número**:

```
<?php
      $num=$_GET["num"];
      $x=1;
      $contador=1;

      while ($contador<=$num)
        {
         $x*=$contador;
         $contador++;
        }
      print "El factorial de $num es: ".$x;

?>
```

El código HTML que llama a al código anterior es el siguiente:

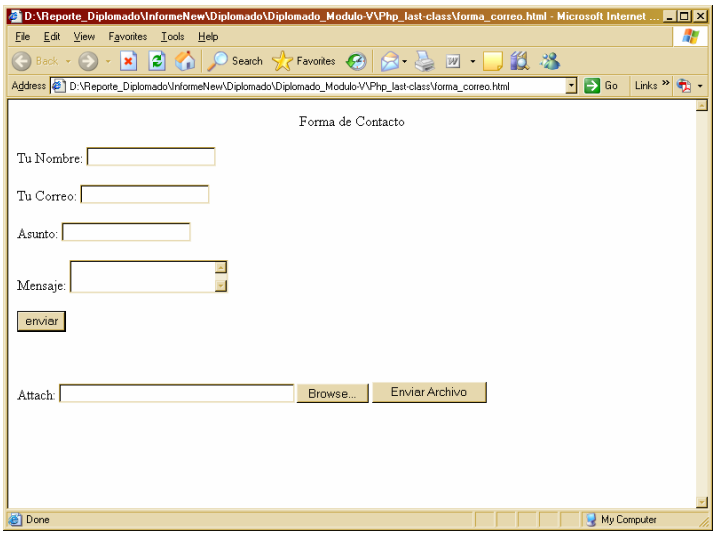
```
<?php
      print "<br><br><br><br>";
      print "<h2 align=center>Factorial</h2>";
?>
  <form align=center action="oper_fact.php">
       <br><br>
       Numero: <input type="text" name="num">
       <br><br>
       <input type="Submit" value="Calculate">
  </form>
```

Como se puede observar en este último código se utilizan etiquetas HTML dentro de código PHP mediante instrucciones *print*. El atributo '*action*' de la etiqueta <form> específica el archivo (*oper_fact.php*) al cual se van a enviar los datos de la forma para que realice los procesos especificados; en este caso calcular el factorial de un número.

## Formulario para envío de correo electrónico

El siguiente código corresponde a un formulario en una página HTML para el envío de correo electrónico, con la posibilidad de poder adjuntar un archivo al mensaje.

```
<html>
      <p align=center>Forma de Contacto</p>
      <form method="post" action="envia_correo.php">
      Tu Nombre: <input type="text" name="nombre">
      <br><br>
      Tu Correo: <input type="text" name="remitente">
      <br><br>
      Asunto: <input type="text" name="asunto">
      <br><br>
      Mensaje: <textarea name="mensaje">
            </textarea>
      <br><br>
      <input type="submit" value="enviar">
      </form>
      <form method="post" action="subir_archivo.php"
enctype="multipart/form-data">
      <br><br>
      Attach: <input name="archivito" type="file" size="40">
      <input type="submit" value="Enviar Archivo">
      </form>
</html>
```



Formulario de envío

El siguiente es el archivo *envia_correo.php* que se encarga de recibir los parámetros del formulario y mandar el correo de acuerdo a dichos parámetros.

```
<?php

      $nombre = $_POST["nombre"];
      $asunto = $_POST["asunto"];
      $mensaje = $_POST["mensaje"];
      $remitente = $_POST["remitente"];
      $headerFrom = "From: $remitente";

      //mail (string to, string subject, string message)
      $res = mail("4runner64@hotmail.com", "$asunto", "$mensaje",
                  "$headerFrom");
      if($res)
        print "Thanks!,..... I've received your comments";
      else
        print "Oppss!...Problems sending your comments";

?>
```

El siguiente código perteneciente al archivo *subir_archivo.php* ejecuta la acción de subir el archivo, que se desea enviar, al servidor.

```
<?php

     /*TODA la información de los archivos
       que se suben al servidor están en la
       variable $_FILES*/

     if(isset($_FILES))
       print "FILES es de tipo: ".gettype($_FILES);

     print "<pre>";
     print_r($_FILES);
     print "</pre>";
     $nombre = $_FILES["archivito"]["name"];
     $tipo = $_FILES["archivito"]["type"];
     $datos = $_FILES["archivito"]["tmp_name"];
     $tamanio = $_FILES["archivito"]["size"];

     print "<p>Info del Archivo</p>
           Nombre: $nombre<br>
           Tipo: $tipo<br>
           Datos: $datos<br>
           Tamanio: $tamanio";

     if(move_uploaded_file ($datos, "./attch/$nombre"))
       print "<br><br>El Archivo ha sido guardado";
     else
       print "Problemas.....no se ha guardado el archivo";

?>
```

## 2.6 LENGUAJE DE PROGRAMACIÓN JAVA

### ¿Qué es Java?

Java es un poderoso lenguaje de programación que fue creado como parte de un proyecto denominado Green[10], en Sun Microsystems para el desarrollo de aplicaciones para dispositivos electrónicos domésticos. Pero Sun se dio cuenta de que el mercado de dispositivos electrónicos inteligentes para el hogar no crecía como se esperaba. Así que al popularizarse la WWW en 1993, la gente de Sun puso su atención en la gran capacidad que tenía el lenguaje para agregar contenido dinámico y animaciones a las páginas Web.

[10]Op. Cit. p. 8

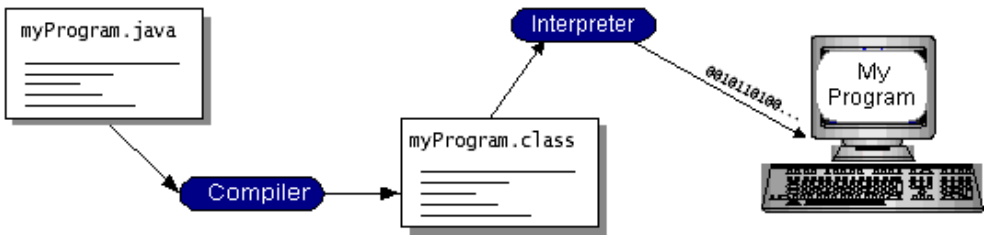
Finalmente en 1995 SUN anunció formalmente Java, que generó gran interés en la comunidad de negocios, debido al auge que estaba teniendo la WWW y las características del lenguaje hacia ésta. En la actualidad Java se utiliza para desarrollar aplicaciones empresariales a gran escala, para mejorar los servicio de World Wide Web, para proporcionar aplicaciones para los dispositivos domésticos (teléfonos celulares, radiolocalizadores y asistentes digitales personales [PDA's]) y muchos otros propósitos.

## El lenguaje de programación

El lenguaje de programación Java es de alto nivel y se caracteriza por ser:

- Simple
- Orientado a objetos
- Distribuido
- Interpretado
- Robusto
- Seguro
- De arquitectura neutral
- De alto rendimiento
- Multihilo
- Dinámico
- Portable

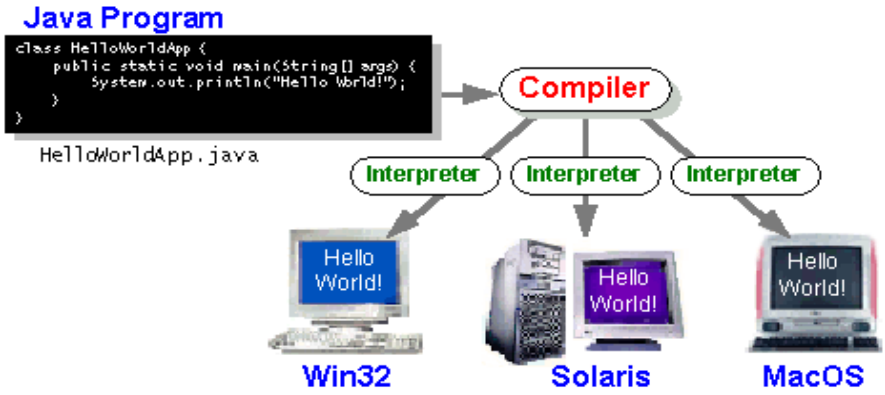
Con la mayoría de los lenguajes programación, los programas se compilan o interpretan para poder ejecutarlos. En este sentido Java es inusual ya que un programa es compilado e interpretado. Con el compilador primero se traslada el programa a un lenguaje intermedio llamado *Java bytecodes*[11]. Que son los códigos independientes de la plataforma y que interpreta el intérprete de la plataforma Java. El intérprete analiza y ejecuta cada instrucción *bytecode* en la computadora. La compilación ocurre una sola vez, y la interpretación ocurre cada vez que el programa es ejecutado. La siguiente figura muestra como funciona:




[11] The Java Tutorial, Sun MicroSystems

Se puede pensar de los Java bytecodes que son el código de instrucciones para la máquina virtual de Java (Java Virtual Machina JVM). Cada intérprete de Java, ya sea una herramienta de desarrollo o un Web browser que pueda correr applets, es una implementación de la máquina virtual de Java.

Los códigos de bytes de Java o Java bytecodes hacen que el lema “escribe una vez, corre en cualquier lado”, se haga posible. Se puede compilar un programa en bytecodes en cualquier plataforma que cuente con un compilador de Java. Los bytecodes pueden entonces ser ejecutados en cualquier implementación de la JVM. Esto quiere decir que si una computadora tiene la JVM, el mismo programa escrito en el lenguaje Java puede correr en Windows 2000, Solaris o en una Mac.




## La plataforma Java

Una plataforma es el entorno de hardware o software en el cual corre un programa. La mayoría de las plataformas pueden ser descritas como una combinación del sistema operativo y hardware. En cambio la plataforma Java difiere del resto de las plataformas en que es una plataforma sólo de software (software-only) que corre en la cima de otras plataformas basadas en hardware (hardware-based).

La plataforma Java tiene dos componentes:

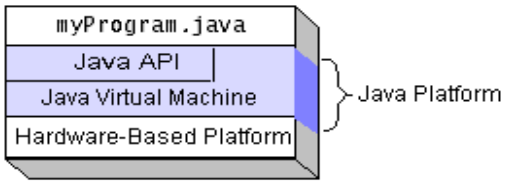
**- La Máquina Virtual de Java (Java Virtual Machina, JVM)**

**- La Interfase de Programación de Aplicaciones (Application Programming Interface, API)**

La Máquina Virtual de Java es la base de la plataforma Java y es transportada a varias plataformas de hardware.

La API de Java es una extensa colección de componentes de software que provee una gran capacidad de utilidad, como componentes de interfaces gráficas de usuario (GUI's). la API de Java está agrupada en bibliotecas de interfaces y clases relacionadas, estas bibliotecas son conocidas como paquetes.

A continuación se muestra una figura que describe un programa que está corriendo en la plataforma Java.




Como la figura muestra, la API y la Máquina Virtual de Java aíslan el programa de el hardware.

El código nativo, es código que después que se compila, dicho código compilado corre en una plataforma de hardware específica. Como un entorno independiente de la plataforma, la plataforma Java puede ser un poco más lenta que el código nativo. Pero aún así, pequeños compiladores, intérpretes bien afinados y compiladores bytecodes just-in-time, pueden mejorar el desempeño y ser muy cercano al del código nativo sin portabilidad.

## Estructura básica del Lenguaje

Considérese el siguiente programa:

```
public class BasicProgram {
    public static void main(String[] args) {
        int sum = 0;
        for (int current = 1; current <= 10; current++) {
            sum += current;
        }
        System.out.println("Sum = " + sum);
    }
}
```

Este programa suma los números del 1 al 10 y despliega el resultado; la salida de este programa es:

=> Sum = 55

Un programa como este hace uso de las características tradicionales del lenguaje Java, incluyendo variables, operadores y sentencias de control de flujo.

Variables: Una variable es un elemento de datos que es nombrado por un identificador. Se requiere especificar un nombre y un tipo para cada variable que se use en un programa. El nombre de la variable se utiliza para referirse a los datos que la variable contiene. El tipo de la variable determina qué valor se puede almacenar y que operaciones se pueden realizar con él. La sintaxis para declarar una variable es: *Tipo nombre*

El lenguaje de programación Java tiene dos categorías de tipos de datos: primitivos y de referencia. Una variable de tipo primitivo contiene un valor simple del tamaño y formato apropiado para su tipo: un número, un caracter o un valor booleano. La siguiente tabla muestra todos los tipos de datos primitivos soportados por la plataforma Java, sus tamaños y formatos y una breve descripción de cada uno:

| **Keyword** | **Description** | **Size/Format** |
|---|---|---|
| *(integers)* | | |
| `byte` | Byte-length integer | 8-bit two's complement |
| `short` | Short integer | 16-bit two's complement |
| `int` | Integer | 32-bit two's complement |
| `long` | Long integer | 64-bit two's complement |
| *(real numbers)* | | |
| `float` | Single-precision floating point | 32-bit IEEE 754 |
| `double` | Double-precision floating point | 64-bit IEEE 754 |
| *(other types)* | | |
| `char` | A single carácter | 16-bit Unicode character |
| `boolean` | A boolean value (`true` or `false`) | true or false |

Los arreglos, clases e interfaces son tipos de datos de referencia. El valor de una variable de tipo referencia, en contraste con el de una variable de tipo primitivo, es una referencia a (la dirección de) el valor o conjunto de valores representados por la variable (véase figura). Una referencia es llamada “puntero”, o una dirección de memoria en otros lenguajes de programación. Java no soporta el uso explícito de direcciones como algunos otros lenguajes. En lugar de la dirección explícita se usa sólo el nombre de la variable.


Operadores: Un operador ejecuta una función sobre uno, dos o tres operandos. Los operadores que requieren un solo operando son llamados operadores unarios. Por ejemplo, ++ es una operador unario que incrementa el valor de su operando en 1. Un operador que requiere dos operandos es un operador binario. Por ejemplo, = es un operador binario que asigna el valor del operando que se encuentra del lado derecho a el operando que se encuentra del lado izquierdo. Por último, un operador ternario es aquel que requiere, para ejecutar su función, de tres operandos. Java tiene un operador ternario, ?:, el cual es una abreviación de la sentencia IF-ELSE.

Los operadores en Java se dividen en varias categorías:

- Operadores aritméticos
- Operadores relacionales y condicionales
- Operadores de cambio y lógicos
- Operadores de asignación
- Otros operadores

## Expresiones, Sentencias y Bloques:

Las variables y operadores son bloques de construcción básicos en los programas. Se combinan literales, variables y operadores para formar expresiones (segmentos de código que realizan cálculos y regresan valores). Ciertas expresiones pueden estar incluidas en sentencias (unidades completas de ejecución). Agrupando un conjunto de sentencias entre {llaves}, se crean bloques de código.

+ Expresiones

Una expresión es una serie de variables, operadores y llamadas a métodos (construidos de acuerdo a la sintaxis del leguaje) que se evalúan para determinar un valor.

Las expresiones realizan el trabajo de un programa. Entre otras cosas, las expresiones son usadas para computar y asignar valores a variables y para ayudar a controlar el flujo de ejecución de un programa.

Cada expresión ejecuta una operación y regresa un valor como se muestra en la siguiente tabla:

| Expresión | Acción | Valor Regresado |
|---|---|---|
| `aChar = 'S'` | Asigna el caracter 'S' a la variable de tipo caracter `aChar` | El valor de `aChar` después de asignarle ('S') |
| `"The largest byte value is " + largestByte` | Concatena la cadena `"The largest byte value is "` y el valor de `largestByte` convertido a cadena | La cadena resultante `The largest byte value is 127` |
| `Character.isUpperCase(aChar)` | Llamada al método `isUpperCase` | El valor que retorna el método: `true` |

+ Sentencias

Las sentencias son equivalentes a las sentencias en lenguaje natural. Una sentencia conforma una unidad completa de ejecución. Los siguientes tipos de expresiones pueden ser creadas dentro de sentencias, terminando cada expresión con un (;):

- Expresiones de asignación
- El uso de ++ o --
- Llamada a métodos
- Creación de objetos

Este tipo de sentencias son llamadas sentencias de expresión. Estos son algunos ejemplos de este tipo de sentencias:

```
aValue = 8933.234;                              //sentencia de asignación
aValue++;                                       //sentencia de incremento
System.out.println(aValue);                     //llamada a método
Integer integerObject = new Integer(4);         //creación de objeto
```

Adicionalmente a estos tipos de sentencias de expresión, existen otros dos tipos de sentencias. Sentencias de declaración:

```
double aValue = 8933.234;                       //sentencia de declaración
```

Y sentencias de control de flujo, que regulan el orden con el cual las sentencias son ejecutadas. Cuando se escribe un programa, se escriben sentencias dentro de un archivo; sin las sentencias de control de flujo, el intérprete ejecuta estas sentencias en el orden en que aparecen dentro del archivo de izquierda a derecha de arriba abajo. De manera que se utilizan las sentencias de control de flujo para ejecutar condicionalmente las sentencias, para ejecutar repetidamente un bloque de sentencias y para cambiar de cualquier otra manera la secuencia normal del flujo de control. Por ejemplo, en el siguiente código, la sentencia if ejecuta condicionalmente la sentencia `System.out.println` que se encuentra dentro de los paréntesis, basándose en el valor que retorna `Character.isUpperCase(aChar)`:

```
char c;
if (Character.isUpperCase(aChar)) {
   System.out.println("The character " + aChar + " is upper case.");
}
```

Java provee varias sentencias de control de flujo, las cuales se muestran en la siguiente tabla:

| **Tipo de Sentencia** | **Palabra clave** |
|---|---|
| Ciclos | `while`, `do-while` , `for` |
| Toma de decisión | `if-else`, `switch-case` |
| Manejo de excepciones | `try-catch-finally`, `throw` |
| bifurcación | `break`, `continue`, `label:`, `return` |

+ Bloques

Un bloque es un grupo de cero o más sentencias escritas dentro de {llaves}, que puede ser usado en cualquier parte donde una sola sentencia es permitida. Las siguientes líneas de código muestran dos bloques con una sola sentencia cada uno:

```
if (Character.isUpperCase(aChar)) {
    System.out.println("The character " + aChar + " is upper case.");
} else {
    System.out.println("The character " + aChar + " is lower case.");
}
```

**Programación orientada a objetos (OOP)**

Java es un lenguaje orientado a objetos, por lo cual es importante conocer los conceptos de la programación orientada a objetos[12]. Algunos de estos conceptos son:

- objeto
- mensaje
- clase
- herencia
- interfase

[12] Idem

## Objeto

Un objeto es un conjunto de software que contiene variables y métodos relacionados. Si se echa un vistazo al mundo que nos rodea se pueden ver muchos ejemplos de objetos del mundo real: un perro, un escritorio, un set de televisión, una bicicleta, etc.

Estos objetos del mundo real comparten dos características: Tienen un estado y un comportamiento. Por ejemplo, los perros tienen un estado (nombre, color, raza) y un comportamiento (ladra, trae los juguetes y mueve la cola). Los objetos de software son modelados a partir de los objetos del mundo real, que al igual que estos tienen un estado y un comportamiento.

Un objeto de software mantiene su estado en una o más variables e implementa su comportamiento con métodos (funciones o subrutinas asociadas con el objeto).

## Mensaje

Con frecuencia un objeto sólo no es muy útil; normalmente aparece como un componente que forma parte de un programa más largo o de una aplicación que contiene muchos otros objetos. A través de la interacción entre objetos se pueden obtener funcionalidades de alto nivel y comportamientos más complejos. La comunicación entre objetos se realiza a través del envío de mensajes.

## Clase

En el mundo real se pueden encontrar varios objetos de la misma tipo. Por ejemplo, una bicicleta es sólo una de muchas bicicletas que hay en el mundo. Cuando se usa la tecnología orientada a objetos, se dice que el objeto bicicleta es una instancia de la clase de objetos conocidos como bicicletas.

En el software orientado a objetos, es posible tener varios objetos del mismo tipo que compartan características. Así como los fabricantes de bicicletas toman ventaja de que las bicicletas comparten características, y construyen muchas bicicletas del mismo molde, así mismo se puede tomar ventaja del hecho de que los objetos del mismo tipo son similares y se pueden crear moldes para esos objetos. Un molde de software para objetos es llamado una clase. En la clase se definen las variables y los métodos comunes a todos los objetos de ese tipo.

## Herencia

En general los objetos se definen en términos de clases. Se puede saber mucho de un objeto conociendo su clase. Los sistemas orientados a objetos toman esto y lo llevan un paso más allá permitiendo a clases ser definidas en términos de otras clases. Es decir, se pueden definir subclases a partir de una clase llamada superclase. Cada subclase hereda el estado (en forma de declaraciones de variables) de la superclase y también cada subclase hereda los métodos de la superclase.

Además las subclases no están limitadas al estado y comportamiento proveído por la superclase. Las subclases pueden agregar variables y métodos a aquellos heredados de la superclase.

## Interfase

Una interfase es un componente (de software), el cual, objetos que no están relacionados a través de una jerarquía de clases, pueden usar para interactuar entre ellos. Un objeto puede implementar múltiples interfases.

Se utiliza una interfaz para definir un protocolo de comportamiento que puede ser implementado por cualquier clase en cualquier parte de la jerarquía de clases. Las interfases son muy útiles para:

**::** Capturar similitudes entre clases no relacionadas sin forzar a una relación artificial de clases.
**::** Declarar métodos que una o más clases están esperando implementar.
**::** Revelar la interfase de programación de un objeto sin revelar su clase.

## **Interfaces Gráficas de Usuario (GUI's)**

Una interfaz gráfica de usuario presenta un mecanismo amigable al usuario para interactuar con un programa, ya que a este último le proporciona una "apariencia visual" única. Al proporcionar distintos programas en los que los componentes de la interfaz de usuario sean consistentes e intuitivos, los usuarios pueden familiarizarse con un programa incluso antes de utilizarlo. Esto, a su vez reduce el tiempo que requieren para aprender a usar un programa y aumenta su habilidad para usar el programa de una manera productiva.

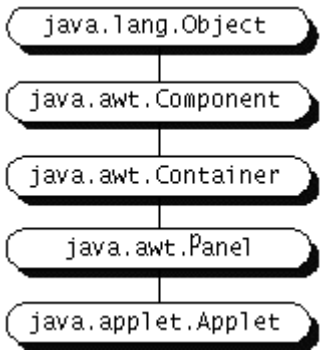
Las GUI's se crean a partir de componentes de la GUI (conocidos también como controles o widgets [accesorios de ventana]). Un componente de la GUI es un objeto con el cual interactúa el usuario mediante el ratón, el teclado u otra forma de entrada. Como el reconocimiento de voz. Algunos de los componentes de la GUI son los siguientes:

| | |
|---|---|
| JLabel | Áreas en dónde pueden mostrarse íconos o texto no editable. |
| JTextField | Área en la que el usuario introduce datos desde el teclado. Esta área también puede mostrar información. |
| JButton | Área que, cuando el ratón hace clic sobre ella, desencadena un evento. |
| JCheckBox | Componente que puede estar o no seleccionado. |
| JComboBox | Lista desplegable de elementos, de los cuales el usuario puede seleccionar uno haciendo clic en él o posiblemente escribiendo dentro del cuadro. |
| JList | Área que contiene una lista de elementos, de los cuales el usuario puede seleccionar cualquiera haciendo clic en él. Pueden seleccionarse varios elementos. |
| JPanel | Contenedor en el cual pueden colocarse y organizarse componentes. |

Las clases que crean los componentes de la GUI del listado anterior, son algunos de los componentes de la GUI de Swing del paquete javax.swing. Estos componentes se hicieron estándar en Java cuando se liberó la Plataforma Java 2 versión 1.2. La mayoría de los componentes de Swing, como se les llama comúnmente, están escritos, se manipulan y muestran, completamente en Java (los denominados componentes puros de Java). Los componentes de Swing son parte de la JFC (Java Foundation Classes), las bibliotecas de Java para el desarrollo de GUI's para múltiples plataformas.

**Applets**

Cada applet se implementa creando una subclase de la clase Applet. La siguiente figura muestra la jerarquía de herencia de la clase Applet. Esta jerarquía determina mucho de lo que un applet puede hacer y como lo hace.




Un applet es una clase extendida de la clase Applet que se incrusta en un documento HTML y se ejecuta en un Web Browser como Internet Explorer o Netscape. Cuando el Browser carga una página que contiene un applet, éste se descarga en el Browser y comienza a ejecutarse. Esto permite crear programas que cualquier usuario puede ejecutar con sólo cargar la página Web correspondiente en su Browser.

El Browser o navegador que ejecuta un applet se conoce en términos genéricos como el *contenedor de applets*.

## 2.7 CASOS DE USO: APLICACIONES EN JAVA

### Calculadora de números complejos

El siguiente programa realiza distintas operaciones con números complejos que obtiene a través de interactuar con el usuario. El código contiene comentarios para entender el propósito de las declaraciones.

```
import javax.swing.*;          //se importan todas las clases del paquete javax.swing
public class Complejo {        //declaración de la clase "Complejo"
    private int real;          //variables globales
    private int imaginario;    //
```

```
    private int sumar;        //
    private int sumaimg;      //con modificador de acceso privado

    public Complejo(int r, int img) {      //constructor de la clase
        real=r;
        imaginario=img;
    }

    public void imprimir(){                //método que imprime el número complejo
        System.out.println(real+ " + " + imaginario+"j");//imprime en línea de
                                                         //comando
       JOptionPane.showMessageDialog(null, real+ " + " + imaginario+"j");
            //imprime en una ventana
    }

  public Complejo sumar(Complejo comp){   //método que suma dos números complejos
    int r=real+comp.real;                 //realiza la suma de la parte real
    int i=imaginario+comp.imaginario;     //realiza la suma de la parte imaginaria
    return new Complejo(r,i);             //regresa el número complejo
    }

    public Complejo restar(Complejo comp){ //método que resta dos números complejos
    int r=real-comp.real;                  //realiza la resta de la parte real
    int i=imaginario-comp.imaginario;      //realiza la resta de la parte imaginaria
    return new Complejo(r,i);              //regresa el número complejo
  }

   public Complejo multiplicar(Complejo comp){ //método que multiplica dos números
                                               //complejos
    int r=(real*comp.real)-(imaginario*comp.imaginario); //lógica de la
                                                         //multiplicación
    int i=(comp.real*imaginario)+(real*comp.imaginario);
    return new Complejo(r,i);                  //regresa el número complejo
  }

  public static Complejo getComplejo(){   //método estático que obtiene los números
                                          // complejos del usuario
    String cadena=JOptionPane.showInputDialog(null,"Dame el valor real");
    int x=Integer.parseInt(cadena);  //convierte la cadena que se obtiene a entero
    String cadena1=JOptionPane.showInputDialog(null,
    "Dame el valor imaginario");
     int y=Integer.parseInt(cadena1);//convierte la cadena que se obtiene a entero
    return new Complejo(x,y);        //regresa el número complejo
  }

  public static int Menu(){   //método que presenta el menú para que el usuario elija
                              //la operación que quiere realizar
    String cadena=JOptionPane.showInputDialog(null,
    "Que Operacion quieres?:\n1Sumar\n2 Restar\n3 Multiplicar\n");
    int x=Integer.parseInt(cadena); //convierte la cadena que se otiene a entero
    return (x);                     //regresa la opción
  }
```

```
  public static void main(String[] args) {   //método main, aquí se inicia la ejecución
                                               //del programa
    Complejo a;                        //se declara una variable de tipo Complejo
    a=Complejo.getComplejo();          //se manda llamar a la función getComplejo
    Complejo b;                        //se declara otra variable de tipo Complejo
    b=Complejo.getComplejo();          //se manda llamar a la función getComplejo
    int op;                            //se declara una variable entera para almacenar la opción
    op=Complejo.Menu();    //se manda llamar a la función Menu
    switch(op){            //se ejecuta la opción elejida
      case 1:                    //si es 1
        res=a.sumar(b);    // se suman los números complejos
        res.imprimir();    //se imprime el complejo resultante
      break;

      case 2:              //si es 2
        res=a.restar(b);  //se restan los números complejos
        res.imprimir();   //se imprime el complejo resultante
      break;

     case 3:                     //si es 3
        res=a.multiplicar(b);  //se multiplican los números complejos
        res.imprimir();        //se imprime el complejo resultante
      break;
    } //cierre switch
  }    //cierre método main
}      //cierre clase
```

En el programa anterior se muestra un detalle importante a cerca de la utilización de *static*. Al utilizar *static* en la declaración de métodos, se les puede llamar directamente indicando el nombre de la clase en la que están declarados. Se puede llamar a cualquier método *static*, o referirse a una variable *static*, utilizando el operador punto con el nombre de la clase, sin necesidad de crear un objeto de ese tipo (new). En el programa se declaran dos métodos *static*: getComplejo y Menu.

Para mandar llamar a cualquiera de los dos métodos únicamente se declara una variable de tipo Complejo y se le asigna el valor que devuelven los métodos al llamarlos con la siguiente sintaxis: *Clase.nombre_método*; cabe mencionar que no es necesario tener que declarar una variable de tipo Complejo, únicamente se podría mandar llamar a los métodos *static* con la sintaxis mencionada.

El siguiente programa muestra como se instancia una clase para crear objetos. Se crean tres objetos: p, p2 y p3 (propiamente dicho estas variables contienen referencias a objetos de tipo Perro) los cuales mandan llamar a los métodos ladrar() y morder().

```
public class Perro {     //declaración de la clase “Perro”
    String nombre;       //variables globales
    String raza;         //…
    int edad;            //…
    float peso;          //…

    public Perro(String n, String r, int e, float p) {   //constructor de la clase
        nombre=n;
        raza=r;          //el constructor sirve para inicializar al objeto inmediatamente
        edad=e;          //después de haberlo creado
        peso=p;
    }

    public void ladrar(){      //método ladrar, no regresa ningún valor
        System.out.println("Perro "+nombre+" de raza "+raza+" ladrando");
    }    //imprime en pantalla un mensaje con el método println

    public void morder(){      //método morder, no regresa ningún valor
        System.out.println(nombre+" me mordiò");
    }

    public String getNombre(){       //método que regresa la variable nombre
        return nombre;
    }

    public static void main(String[] args) {            //método principal
        Perro p;  //declaración de la variable p de tipo Perro
        Perro p2; //declaración de la variable p2 de tipo Perro
        Perro p3; //declaración de la variable p3 de tipo Perro
        p=new Perro("Fido","Labrador",2,40.00f);     //se instancia la clase Perro,
                                                     //la variable p de tipo Perro
                                                     //contiene una  referencia (que
                                                     //regresa el operador new
                                                     //después de haber creado
                                                     //el objeto) hacía un
                                                     //objeto de tipo Perro
        p.ladrar();      //el objeto p manda llamar al método ladrar
        p2= new Perro("Cuco","Mastin",3,30.00f);       //se instancía la clase Perro
        p2.ladrar();     //el objeto p2 manda llamar al método ladrar
        p3=new Perro("Febbee","SharPei",1,20.00f);     //se instancía la clase Perro
        p3.morder();     //el objeto p3 manda llamar al método morder
        System.out.println(p.getNombre());
    }//fin método main
}//fin de la clase Perro
```

En el siguiente programa se muestra la forma de trabajar con archivos en Java, así como el manejo de excepciones:

```
import java.io.*;          //se importa el paquete java.io que contiene //las clases necesarias para el
                           //manejo de archivos

public class Persona {  //declaración de la clase "Persona"
    private String nombre;     //variables globales
    private String apellido;   //...
    private int edad;          //...

    //constructor de la clase
    public Persona(String name, String last_name, int age) {
        nombre=name;
        apellido=last_name;
        edad=age;
    }

    //método estático que escribe una cadena en el archivo personas
    public static void guardar(String cad)throws IOException{
        FileOutputStream k;    //ya que en Java toda la API está //construida en forma de
                               //distintas clases y cada clase a parte de tener una función,
                               //define un nuevo tipo de dato, se declara una variable
                               //de Tipo FileOutputStream, que se utiliza para guardar la
                               //referencia a un objeto de esa misma clase, al cual
                               //se le pasa como parámetro el nombre del archivo
                               //dónde se va a escribir la cadena
        try{       //esta sentencia indica qué bloque de código ejecutar
                   //cuando se presente una excepción
            cad="Setenta";
            k= new FileOutputStream("C:/personas.txt");
            for(int i=0; i < cad.length(); i++)
              k.write(cad.charAt(i));
            /*otra forma de escribir cadenas es: k.write(cad.getBytes());*/
            k.close();
        }
        catch(FileNotFoundException l){} //cacha la excepción que se
                                         //genera en el caso de que
                                         //no se encuentre el archivo
    }

    public static void main(String[] args)throws IOException {

        //lee archivo
        FileInputStream f;     //se crea una variable del tipo
                               //FileInputStream para guardar la
                               //referencia a un objeto que se instancia
                               //de esta clase, al cual se le pasa como
                               //parámetro el nombre y la ruta absoluta
                               //del archivo que se va a leer.
```

```
        try{
            f= new FileInputStream("C:/personas.txt");
            int car=f.read();
            while(car!= -1){ //mientras no sea fin de archivo
                             //imprime caracter por caracter

             System.out.println((char)car);
             car=f.read();

            }
            f.close();         //se cierra el archivo
        }
        catch(FileNotFoundException e){} //se cacha la excepción
                                         //probablemente producida por
                                         //no haberse encontrado el //archivo
        catch(IOException e){}       //se cacha cualquier otro tipo de
                                     //excepción
  }
}
```

## 2.8 ¿QUÉ SON LAS FAMOSAS BASES DE DATOS?

Antes de definir que es una base de datos es necesario definir los siguientes conceptos:

**Datos:** Son hechos, números, letras o símbolos en bruto que por si solos no tienen ningún significado.

**Información:** Es la interpretación de los datos. La información consiste en conocimientos producidos como resultado de las operaciones de procesamiento de datos.

**Esquema:** El diseño completo de la base de datos se llama el esquema de la base de datos.

Una **base de datos**[13] es una colección estructurada de datos que pertenecen al mismo contexto almacenados sistemáticamente para su uso posterior. Los datos que puede almacenar una base de datos conforman la información de algo tan simple como la de una agenda, un contador, o un libro de visitas, ó tan vasta como la de una tienda en línea, un sistema de noticias, un portal, o la información generada en una red corporativa. En este sentido, una biblioteca puede considerarse una base de datos compuesta en su mayoría por documentos y textos impresos en papel e indizados para su consulta.

[13] http://searchoracle.techtarget.com/sDefinition/0,,sid41_gci902816,00.html

Las aplicaciones más usuales son para la gestión de empresas e instituciones públicas. También son ampliamente utilizadas en entornos científicos con el objeto de almacenar la información experimental.

En la actualidad, y gracias al desarrollo tecnológico de campos como el cómputo y la electrónica, la mayoría de las bases de datos tienen formato electrónico, que ofrece un amplio rango de soluciones al problema de almacenar datos.

Existen los sistemas manejadores de bases de datos (DBMS), que permiten almacenar y posteriormente acceder a los datos de forma rápida y estructurada.

Un **DBMS** consiste en una colección de datos interrelacionados y un conjunto de programas para acceder a dichos datos. El objetivo principal de un DBMS es proporcionar una manera de almacenar y recuperar la información de una base de datos de manera que sea tanto práctica como eficiente.

Para definir una base de datos eficiente es necesario seguir un modelo de datos; dicho modelo es una colección de herramientas conceptuales para describir los datos, las relaciones, la semántica y las restricciones de consistencia.

El modelo más utilizado hoy en día es el Modelo Entidad-Relación. Este modelo de datos está basado en una percepción del mundo real que consta de una colección de objetos básicos, llamados entidades, y de relaciones entre estos objetos. Una entidad es una <<**cosa**>>u <<**objeto**>>en el mundo real que es distinguible de otros objetos. Las entidades se describen en una base de datos mediante un conjunto de **atributos**. Una **relación** es una asociación entre varias entidades. Las entidades son representadas con rectángulos, las elipses representan los atributos, y los rombos representan las relaciones.

Existe otro modelo conceptual llamado Modelo Relacional. En el modelo relacional se utiliza un grupo de tablas para representar los datos y las relaciones entre ellos. Cada tabla está compuesta por varias columnas, y cada columna tiene un nombre único. En este modelo se encuentran los conceptos de grado y cardinalidad. El **grado** de una tabla se refiere al número de atributos de una tabla. La **cardinalidad** es el número de tuplas que contiene una tabla.

Un sistema de base de datos proporciona un **lenguaje de definición de datos (DDL)** para especificar el esquema de la base de datos y un **lenguaje de manipulación de datos (DML)** para expresar las consultas a la base de datos y las modificaciones. En la práctica, los lenguajes de definición y manipulación de datos no son dos lenguajes separados; en su lugar simplemente forman parte de un único lenguaje de bases de datos, tal como SQL (Structured Query Language), que es actualmente un estándar en la industria.

**Ciclo de vida de una base de datos.**

Por regla general el ciclo de vida de una base de datos consta de seis fases:

**Análisis:** En la fase de análisis se entrevista a los accionistas y se examinan todos los sistemas existentes para identificar los problemas, las posibilidades y los límites. En esta fase se determinan los objetivos y el ámbito del nuevo sistema.

**Diseño:** La fase de diseño se encarga de la creación del diseño conceptual a partir de las necesidades determinadas previamente. También se crea un diseño lógico y físico para preparar la implantación de la base de datos.

**Implantación:** En esta fase se instala el sistema de administración de la base de datos (DBMS), se crea la base de datos y se cargan o importan los datos.

**Pruebas**: En la fase de pruebas se examina la base de datos y se ajusta, por lo general junto a las aplicaciones asociadas.

**Puesta en marcha:** En esta fase la base de datos opera normalmente, produciendo información para sus usuarios.

**Mantenimiento:** En la fase de mantenimiento se introducen cambios en la base de datos en respuesta a las nuevas necesidades o se modifican las condiciones operativas.

## 2.9 LA RELACIÓN ENTRE LAS BASES DE DATOS Y LOS LENGUAJES DE PROGRAMACIÓN

Las bases de datos al permitir almacenar conjuntos de datos, por sí mismas no representan información y tampoco las datos son meramente información. Para que los datos puedan tener un sentido, sean útiles y tengan algún valor para la toma de decisiones en las empresas e instituciones, es necesario procesar estos datos.

El procesarlos implica extraerlos de la base de datos, pero no sólo es extraerlos, se necesitan extraer los datos que se requieran únicamente; se deben de analizar y presentar en forma ahora si, de información. Los lenguajes de programación ayudan a crear las aplicaciones que interactuando con los sistemas manejadores de bases de datos, extraen los datos de las bases de datos, los analizan y los presentan como información. Esto permite que los datos sean completamente independientes de las aplicaciones que los utilizan, otorgando flexibilidad al desarrollo de sistemas, ya que se pueden utilizar diferentes lenguajes de programación para crear diversas aplicaciones que utilicen el mismo conjunto de datos.

## 2.10 EL MANEJADOR DE BASES DE DATOS MYSQL

MySQL es el servidor de bases de datos relacionales más popular, desarrollado y proporcionado por MySQL AB. MySQL AB es una empresa cuyo negocio consiste en proporcionar servicios en torno al servidor de bases de datos MySQL. Una de las razones para el rápido crecimiento de popularidad de MySQL, es que se trata de un producto Open Source, y por lo tanto, va de la mano con este movimiento.

Como se mencionó anteriormente, una base de datos al ser una colección estructurada de datos, se necesita un sistema de administración de bases de datos, tal como MySQL para agregar, acceder, y procesar los datos almacenados en una base de datos,

Una base de datos relacional almacena los datos en tablas separadas en lugar de poner todos los datos en un solo lugar. Esto agrega velocidad y flexibilidad. Las tablas son enlazadas al definir relaciones que hacen posible combinar datos de varias tablas cuando se necesitan consultar datos. La parte SQL de "MySQL" significa "Lenguaje Estructurado de Consulta" (Structured Query Language), y es el lenguaje más usado y estandardizado para acceder a bases de datos relacionales.

El servidor de bases de datos MySQL es muy rápido, seguro, y fácil de usar. Fue desarrollado originalmente para manejar grandes bases de datos mucho más rápido que las soluciones existentes y ha estado siendo usado exitosamente en ambientes de producción sumamente exigentes por varios años. Aunque se encuentra en desarrollo constante, el servidor MySQL ofrece hoy un conjunto rico y útil de funciones. Su conectividad, velocidad, y seguridad hacen de MySQL un servidor bastante apropiado para acceder a bases de datos en Internet.

MySQL consiste de un sistema cliente/servidor que se compone de un servidor SQL multihilo, varios programas clientes y bibliotecas, herramientas administrativas, y una gran variedad de interfaces de programación (APIs). Se puede obtener también como una biblioteca multihilo que se puede enlazar dentro de otras aplicaciones para obtener un producto más pequeño, más rápido, y más fácil de manejar. La manera oficial de pronunciar MySQL es "my ess que ell" (no "my sequel"), pero no existe ningún inconveniente en pronunciarlo como "my sequel", ó de alguna otra manera.

Inicialmente, MySQL carecía de elementos considerados esenciales en las bases de datos relacionales, tales como integridad referencial y transacciones. A pesar de ello, atrajo a los desarrolladores de páginas Web con contenido dinámico, justamente por su simplicidad; aquellos elementos faltantes fueron llenados por la vía de las aplicaciones que la utilizan.

Poco a poco los elementos faltantes en MySQL están siendo incorporados tanto por desarrollos internos, como por desarrolladores de software libre. Entre las características disponibles en las últimas versiones se puede destacar:

- Amplio subconjunto del lenguaje SQL. Algunas extensiones son incluidas igualmente.
- Disponibilidad en gran cantidad de plataformas y sistemas.
- Diferentes opciones de almacenamiento según si se desea velocidad en las operaciones o el mayor número de operaciones posibles.
- Transacciones y claves foráneas.
- Conectividad segura.
- Replicación.
- Búsqueda e indización de campos de texto.
- Procedimientos almacenados.
- Triggers

- Vistas
- Cursores
- Transacciones Distribuidas XA
- Esquema de Información

**Trabajando con MySQL**[14]

Para conectarse al servidor, se necesita de un nombre de usuario (login) y de una contraseña (password), y si el servidor al que se desea conectar está en una máquina distinta en la cual se esta trabajando, también se necesita indicar el nombre o la dirección IP de dicho servidor. Una vez que se conocen estos tres valores, la conexión se realiza de la siguiente manera:

```
shell> MySQL -h NombreDelServidor -u NombreDeUsuario -p
```

Cuando se ejecuta este comando, se pide también la contraseña para el nombre de usuario que se esta usando.

Si la conexión al servidor MySQL se pudo establecer de manera satisfactoria, se recibe el mensaje de bienvenida e inmediatamente aparece el prompt de MySQL:

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 5563 to server version: 3.23.41

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

MySQL>
```

Este prompt indica que MySQL está listo para recibir comandos.

Después de haber realizado la conexión de manera satisfactoria, la desconexión se puede efectuar en cualquier momento al escribir "*quit*", "*exit*", o presionar CONTROL+D.

[14] http://www.programacion.com/bbdd/tutorial/mysql_basico/

Una vez que se ha hecho una conexión con el servidor se puede empezar a trabajar. Veáse el siguiente ejemplo:

```
MySQL> SELECT VERSION(), CURRENT_DATE;
+-----------+--------------+
| VERSION() | CURRENT_DATE |
+-----------+--------------+
| 3.23.41   | 2002-10-01   |
+-----------+--------------+
1 row in set (0.03 sec)
```

Este comando ilustra distintas cosas acerca de MySQL:

- Un comando normalmente consiste de un sentencia SQL seguida por un punto y coma.

- Cuando se ingresa un comando, MySQL lo manda al servidor para que lo ejecute, muestra los resultados y regresa el prompt indicando que está listo para recibir más consultas.

- MySQL muestra los resultados de la consulta como una tabla (filas y columnas). La primera fila contiene etiquetas para las columnas. Las filas siguientes muestran los resultados de la consulta. Normalmente las etiquetas de las columnas son los nombres de los campos de las tablas que estamos usando en alguna consulta. Si lo que se está recuperando es el valor de una expresión (como en el ejemplo anterior) las etiquetas en las columnas son la expresión en sí.

- MySQL muestra cuántas filas fueron regresadas y cuanto tiempo tardó en ejecutarse la consulta, lo cual puede dar una idea de la eficiencia del servidor, aunque estos valores pueden ser un tanto imprecisos ya que no se muestra la hora del CPU, y porque pueden verse afectados por otros factores, tales como la carga del servidor y la velocidad de comunicación en una red.

- Las palabras clave pueden ser escritas usando mayúsculas y minúsculas.

Las siguientes consultas son equivalentes:

```
mysql> SELECT VERSION(), CURRENT_DATE;
mysql> select version(), current_date;
mysql> SeLeCt vErSiOn(), current_DATE;
```

Es posible escribir más de una sentencia por línea, siempre y cuando estén separadas por punto y coma:

```
mysql> SELECT VERSION(); SELECT NOW();
+-----------+
| VERSION() |
+-----------+
| 3.23.41   |
+-----------+
1 row in set (0.01 sec)

+---------------------+
| NOW()               |
+---------------------+
| 2002-10-28 14:26:04 |
+---------------------+
1 row in set (0.01 sec)
```

Un comando no necesita ser escrito en una sola línea, así que los comandos que requieran de varias líneas no son un problema. MySQL determinará en donde finaliza la sentencia cuando encuentre el punto y coma, no cuando encuentre el fin de línea.

La siguiente consulta simple está escrita en varias líneas:

```
mysql> SELECT
    -> USER(),
    -> CURRENT_DATE;
+----------------+--------------+
| USER()         | CURRENT_DATE |
+----------------+--------------+
| root@localhost | 2002-09-14   |
+----------------+--------------+
1 row in set (0.00 sec)

mysql>
```

En este ejemplo debe notarse como cambia el prompt (de MySQL> a ->) cuando se escribe una consulta en varias líneas. Esta es la manera en cómo MySQL indica que está esperando a que finalice la consulta. Sin embargo si se desea no terminar de escribir la consulta, se puede hacer al escribir \c como se muestra en el siguiente ejemplo:

```
mysql> SELECT
    -> USER(),
    -> \c
mysql>
```

En la siguiente tabla se muestran cada uno de los prompts que se pueden obtener al ir escribiendo comandos y una breve descripción de su significado para MySQL:

| Prompt | Significado |
|---|---|
| MySQL> | Listo para una nueva consulta. |
| -> | Esperando la línea siguiente de una consulta multi-línea. |
| '> | Esperando la siguiente línea para completar una cadena que comienza con una comilla sencilla ('). |
| "> | Esperando la siguiente línea para completar una cadena que comienza con una comilla doble ("). |

## Creación de una base de datos

En MySQL se cuenta con el comando *SHOW* para mostrar todas las bases de datos que se encuentran en el servidor:

```
mysql> SHOW DATABASES;
+----------+
| Database |
+----------+
| mysql    |
| test     |
+----------+
2 rows in set (0.00 sec)

mysql>
```

Dependiendo de las bases de datos que se encuentren en el servidor son las que se van a mostrar, además de las que se muestran en el ejemplo. La base de datos "MySQL" es requerida, ya que ésta tiene la información de los privilegios de los usuarios de MySQL. La base de datos "test" es creada durante la instalación de MySQL con el propósito de servir como área de trabajo para los usuarios que inician en el aprendizaje de MySQL.

Se usa la sentencia CREATE DATABASE para crear una nueva base de datos:

```
mysql> CREATE DATABASE ventas;
Query OK, 1 row affected (0.00 sec)
```

Al crear una base de datos no se selecciona ésta de manera automática; se debe hacerlo de manera explícita, por ello se usa la sentencia USE:

```
mysql> USE ventas
Database changed
 mysql>
```

La base de datos se crea una sola vez, pero debe ser seleccionada cada vez que se inicia una sesión en el servidor MySQL; por esto preferentemente debe indicarse la base datos sobre la cual se va a trabajar al momento de conectarse al servidor:

```
shell> mysql -h NombreServidor -u NombreUsuario -p BasedeDatos
```

Crear la base de datos no conlleva mucha ciencia, pero en este momento la base de datos está vacía, como lo indica el comando SHOW TABLES:

```
mysql> SHOW TABLES;
Empty set (0.00 sec)
```

- Creación de tablas

Para crear tablas en la base de datos ventas, se utiliza la sentencia CREATE TABLE de la siguiente manera:

```
CREATE TABLE vendedor
(
  id_vendedor int, nombre varchar(30),
  apellido_paterno varchar(30)
  CONSTRAINT pk_vendedor PRIMARY KEY (id_vendedor)
) Type=InnoDB;
```

Se crea una tabla en la base de datos que consta de tres campos: id_vendedor, nombre y apellido_paterno. Cada campo debe de ser definido de un tipo de dato, como se muestra. MySQL soporta un buen número de tipos de columnas en varias categorías: tipos numéricos, tipos de fecha y hora y tipos de cadena. Todos los tipos de datos que soporta MySQL se muestran en el Anexo I.

Se usa la sentencia CONSTRAINT para definir la restricción de llave primaria para el campo id_vededor. La llave primaria de una tabla sirve para poder acceder de manera única a los datos de esa tabla.

Se crean dos tablas más en la base de datos: producto y venta_producto. Las tres tablas creadas son del tipo InnoDB como se muestra en la sentencia que se usa junto antes del indicador de fin de comando (;): Type=InnoDB. InnoDB es un mecanismo que permite realizar transacciones seguras en una base datos, con capacidades de aplicar commit (realizar cambios), rollback (deshacer trabajo) y crash recovery (recuperación de datos).

```
CREATE TABLE producto
 (
   id_producto int,
   nombre_producto varchar(30),
   precio double,
   CONSTRAINT pk_producto PRIMARY KEY (id_producto)
 ) Type=InnoDB;

CREATE TABLE venta_producto
 (
   id_venta int,
   id_vendedor int,
   id_producto int,
   cantidad int,
   CONSTRAINT pk_vp PRIMARY KEY (id_venta),
   CONSTRAINT fk_vpv FOREIGN KEY (id_vendedor) REFERENCES vendedor
   (id_vendedor),
   CONSTRAINT fk_vpp FOREIGN KEY (id_producto) REFERENCES producto
   (id_producto)
 ) Type=InnoDB;
```

En la tabla venta-producto se definen dos llaves foráneas. Una llave foránea es un campo de una tabla, en este caso de la tabla venta-producto, que hace referencia a un campo de otra tabla, en este caso vendedor o producto, que se encuentra definido como llave primaria. Esto es útil para el cruce de tablas cuando se requieren extraer datos de varias tablas.

- Inserción de datos

Una vez creadas las tablas en la base de datos, se insertan datos en ellas haciendo uso de la sentencia INSERT, como se muestra a continuación:

```
INSERT INTO vendedor VALUES (1,'JUAN CARLOS','LEDEZMA');
```

Este comando inserta en la tabla vendedor los valores dentro de los paréntesis que corresponden en orden a los campos definidos de la tabla. De la misma forma se pueden insertar n registros dentro de la tabla, mientras no se exceda el tamaño permitido de la tabla que es limitado por el S.O donde se esté corriendo el servidor MySQL.

Se insertan más registros en la tabla vendedor y también en las tablas producto y venta_producto, sobre las cuales se van a realizar consultas.

```
INSERT INTO producto VALUES (1,'COCA COLA',6.5);

INSERT INTO venta_producto VALUES (1,1,1,5);
```

- Selección de datos

Para extraer los datos de las tablas se usa el comando SELECT:

```
SELECT nombre, nombre_producto, cantidad
  FROM vendedor, venta_producto, producto
 WHERE vendedor.id_vendedor = venta_producto.id_vendedor
   AND producto.id_producto=venta_producto.id_producto;
```

La estructura del comando se define de la siguiente manera:

```
SELECT nombre_campo[1], nombre_campo[2], …, nombre_campo[n]
  FROM nombre_tabla[1], nombre_tabla[2], …, nombre_tabla[n]
 WHERE condiciones
```

Se usa la sintaxis *nombre_tabla.nombre_campo* para referenciar los campos de una tabla. La sentencia WHERE hace uso de esta referenciación para establecer las condiciones de búsqueda de los datos. En el ejemplo, las condiciones se establecen de la siguiente manera:

Primeramente, se van a seleccionar los campos nombre, nombre_producto y cantidad de las tablas (FROM) ***vendedor***, ***venta_producto*** y ***producto***, dónde (WHERE) el campo id_vendedor de la tabla vendedor sea igual al campo id_vendedor de la tabla venta_producto y (AND) el campo id_producto de la tabla producto sea igual al campo id_producto de la tabla venta_producto. Estas condiciones se pueden establecer de manera distinta, para obtener como resultado la selección de los datos requeridos.

```
SELECT nombre_producto, precio, SUM(cantidad)
  FROM venta_producto, producto
 WHERE venta_producto.id_producto= producto.id_producto
 GROUP BY nombre_producto;
```

En este ejemplo se utiliza la función SUM(), para efectuar la suma de todos los campos de la columna cantidad. Las funciones pueden ser utilizadas como en este caso directamente en las consultas. MySQL soporta un gran número de funciones SQL, algunas de ellas se muestran en el Anexo II.

```
SELECT nombre_producto, precio, SUM(cantidad) AS productos_vendidos,
                                SUM(cantidad)*precio AS total
  FROM venta_producto INNER JOIN producto ON
       venta_producto.id_producto= producto.id_producto
 GROUP BY nombre_producto;
```

En este comando se realiza la unión de tablas con INNER JOIN. Se especifican los campos a seleccionar, en este caso nombre_producto y precio; además se utiliza la función SUM() para sumar los campos de la columna cantidad y se le pone un nombre a la columna resultante con la opción AS, productos_vendidos. Se realiza además la operación de sumar los campos de la columna cantidad y se multiplica el resultado con el operador de multiplicación * por los campos de la columna precio, y se le asigna un nombre a la columna, total.

Después se indican las tablas de las cuales se extraerán los campos, venta_producto INNER JOIN (en unión) con producto y la condición para traer los datos ON dónde el campo id_producto de la tabla venta_producto sea igual al campo id_producto de la tabla producto.

- Alterar y Borrar una tabla

Si se requiere modificar la estructura de la una tabla, existe la sentencia SQL ALTER. Esta sentencia permite agregar campos a las tablas (ADD), cambiar campos, borrar llaves primarias con ayuda de la sentencia DROP, renombrar una tabla o adicionar llaves primarias. A continuación se muestran algunos ejemplos:

```
ALTER TABLE vendedor ADD apellido_materno varchar(30);
ALTER TABLE vendedor CHANGE nombre nombre_pila varchar(30);
ALTER TABLE vendedor DROP PRIMARY KEY;
ALTER TABLE vendedor RENAME vendedores;
ALTER TABLE vendedor ADD CONSTRAINT PRIMARY KEY (id_vendedor);
```

Para borrar una tabla se usa la sentencia DROP de la siguiente manera:

```
DROP TABLE vendedor;
```

Y si se requieren borrar todos o algunos registros de una tabla se utiliza la sentencia DELETE:

```
DELETE FROM vendedor;     //borra todos los registros

DELETE FROM vendedor
WHERE id_vendedor = 45;   //borra el(los) registro(s)
                          //que cumplan con la condición
```

# CAPÍTULO III

# ADMINISTRACIÓN DE SERVIDORES WEB

## 3.1 ¿QUÉ ES UN SERVIDOR WEB?

Un servidor Web es un software que se encarga de manejar peticiones HTTP. Es decir ofrece comunicación a través de dicho protocolo. Cuando el servidor Web recibe una petición HTTP, responde con una respuesta HTTP, enviando de regreso una página HTML. Para procesar una petición, el servidor Web puede responder con una página estática o una imagen, enviar un redireccionamiento o delegar la generación dinámica de la respuesta a algún otro programa como un script CGI, un JSP (JavaServer Pages), un servlet, un ASP (Active Server Pages), un server-side JavaScript o alguna otra tecnología del lado del servidor. Cualquiera que sea su propósito, dichos programas del lado del servidor generan una respuesta, generalmente en HTML, para ser vista en un navegador Web.

## 3.2 LOS SERVIDORES WEB MÁS POPULARES

Existe un gran número de empresas y organizaciones que ofrecen Servidores Web en el mercado, cada uno de ellos con distintas características que cubren necesidades propias de cada usuario. Los más populares en la actualidad son:

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

Para inclinarse por el empleo de uno de ellos se recomienda evaluar los siguientes criterios de selección:

- Función del servidor Web. Es importante establecer qué función va a desempeñar el servidor en relación a la estrategia de negocios. Es decir, pudiera implantarse para que los empleados de una compañía accedan a diversas páginas, para consultar determinada información de la base de datos. O pudiera implantarse para que los clientes de la misma compañía, a través de Internet, conozcan los productos y servicios que ésta ofrece.

- Nivel de expertise que tiene(n) el(los) administrador(es). Es de vital importancia considerar que tan experto(as) es(son) la(las) persona(s) que va(n) a administrar el servidor Web; en función de sus conocimientos y experiencia.

- Plataforma disponible. Ya que, por ejemplo, en el caso del servidor IIS de Microsoft, éste sólo se puede correr en un servidor con sistema operativo de la misma empresa.

- Número de conexiones concurrentes. Se debe considerar la estimación del número de conexiones concurrentes que debe atender el servidor.

- Número de transacciones por segundo. Como un factor importante de la capacidad del servidor, se debe tomar en cuenta cuantas transacciones por segundo puede realizar.

- Costo computacional por transacción. Aunado al criterio anterior se debe considerar el costo computacional (tiempo de procesador, memoria, envío y recepción de datos) por cada transacción realizada.

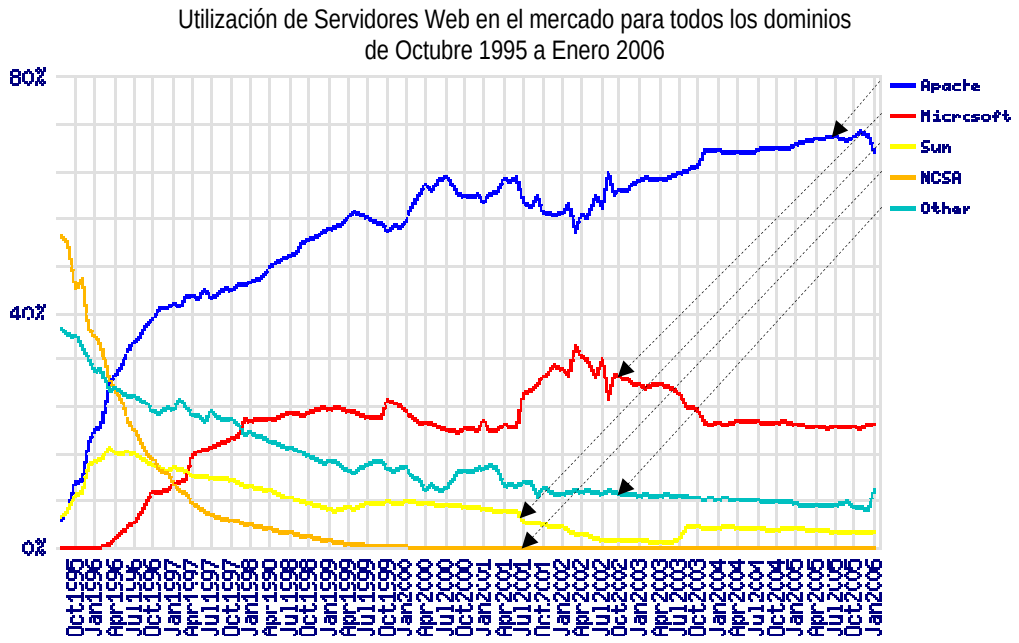
- Proyección del crecimiento esperado. Para saber si el servidor cubrirá las expectativas de crecimiento en el futuro, se debe hacer una proyección de dicho crecimiento ya que este factor es muy importante para determinar el empleo de un servidor u otro.

- Soporte para la tecnología utilizada para el desarrollo. Ya que no todos los servidores Web tienen compatibilidad con todas las tecnologías de desarrollo, es importante considerar que el servidor que se vaya a emplear sea compatible con la(as) tecnología(s) que se van a utilizar para el desarrollo.

- Análisis del retorno de inversión. Ya que el factor económico es muy importante a la hora de implantar un servidor Web, el análisis del ROI (Return On Investment) se vuelve obligatorio.

## 3.3 APACHE: EL SERVIDOR WEB LIBRE

Apache es el Servidor Web más utilizado en la actualidad debido a las características que ofrece y así lo muestra la siguiente gráfica[15]:




El Proyecto del Servidor Web Apache es un esfuerzo de colaboración para el desarrollo de un software que es la implantación de un Servidor HTTP o Web que apunta a ser robusto, de nivel comercial, altamente funcional y de código abierto. Este proyecto es juntamente manejado por un grupo de voluntarios que se encuentran alrededor del mundo, y que utilizan Internet y la Web para comunicarse, planear y desarrollar el servidor y su documentación relacionada. Este proyecto es parte de la Apache Software Foundation.

[15] http://news.netcraft.com/archives/web_server_survey.html

### 3.3.1 CARACTERÍSTICAS DE APACHE

Se ha escogido trabajar con Apache precisamente por sus características, de entre las cuales destacan:

• Robusto, Soporta un gran número de transacciones.

• Configurable para diferentes entornos de trabajo.

• Ofrece un alto nivel de seguridad.

• Disponible para una gran variedad de plataformas.

• Soporte para servicio de proxy

• Soporte para granjas de servidores

• Soporte para Scripting languages integrados como módulos (PHP, mod_perl)

• Incluye el código fuente del servidor

• Soporte para accesos restringidos

• Soporte para SSL (Secure Socket Layer)

• Es libre.

### 3.3.2 INSTALACIÓN Y CONFIGURACIÓN DE APACHE

**Instalación**[16]

Apache se puede obtener de http://www.apache.org de manera gratuita. Una vez que se ha bajado el archivo, se debe descomprimir con el siguiente comando:

```
# tar –zvxf httpd-xx.tar
```

Al descomprimirse se genera un directorio que contiene el código fuente de la distribución que se obtuvo. Se debe ingresar a ese directorio antes de seguir con la compilación.

[16] http://httpd.apache.org/docs/2.2/en/install.html#extract

El siguiente paso es configurar el árbol fuente para la plataforma particular y requerimientos individuales. Esto se hace con el script `configure`, incluido en el directorio raíz de la distribución que se obtuvo. Para configurar el árbol fuente de Apache con las opciones por defecto, únicamente se llama el script:

```
# ./configure
```

Este script acepta un gran número de opciones de configuración, pero la más importante, `--prefix`, es el directorio en el cual se va a instalar Apache, ya que es importante para que el servidor trabaje apropiadamente:

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

También en este punto se puede configurar Apache especificándole alguna característica que se quiera de las que incluye Apache a manera de habilitar o deshabilitar módulos. Apache viene con un set base de módulos incluidos por defecto. Otros módulos se habilitan usando la opción `--enable-module`, donde module es el nombre del módulo.

Se puede elegir también compilar módulos como Dynamic Shared Objects (DSO's), los cuales pueden ser cargados o descargados en tiempo de ejecución. Estos módulos a diferencia de los compilados estáticamente (dentro del binario del servidor), se compilan de manera que están separados del binario principal del servidor. Los módulos DSO pueden ser compilados al mismo tiempo que el servidor, o pueden ser compilados y adicionados después utilizando la Apache Extension Tool `apxs`.

```
# ./configure --prefix=/user/local/apache2 --enable-module=so
```

El módulo ***so*** permite cargar código ejecutable y módulos en el servidor al iniciarlo o reiniciarlo.

Una vez configuradas las opciones, se compila el servidor:

```
# make
```

En este paso se debe ser paciente, ya que desde una configuración base puede tardar varios minutos en compilar, y el tiempo puede variar ampliamente dependiendo del hardware y del número de módulos que se hayan habilitado.

Por último viene la instalación, bajo la configuración establecida con `--prefix`:

```
# make install
```

Para comprobar que se ha instalado correctamente el servidor se inicia con el comando:

```
# ./apachectl start
```

Una vez levantado el servidor, se abre un navegador Web y se escribe en la barra de direcciones *localhost* o la dirección de loopback *127.0.0.1*; aparecerá un mensaje de bienvenida, lo cual indica que se ha instalado exitosamente Apache.

## Configuración

### Directivas[17]

Apache es administrado por más de doscientas directivas, las cuales permiten que determinada funcionalidad pueda ser incluida. En Linux el administrador controla qué directivas estarán disponibles de a cuerdo a los módulos con los que se compila Apache.

En Apache la configuración se realiza mediante tres grupos de directivas:

- Global Environment
- Main Server
- Virtual Host

La sección Global Environment administra las directivas generales de operación para Apache.

La sección Main Server administra las directivas del servidor principal o estándar de Apache.

La sección Virtual Host administra las directivas donde los mismos procesos de Apache soportan diversas direcciones IP o diversos nombres de dominio.

[17] http://httpd.apache.org/docs/2.2/en/mod/directives.html#E

*Global Environment*

**ServerName**

Define el nombre y puerto para el servidor Apache para ser identificado. Es utilizado cuando se construyen los URL's en el momento que se solicitan recursos Web.

**User**

Esta directiva define el identificador de usuario (user ID), bajo el cual Apache operará. Para utilizar esta directiva, el servidor debe estar corriendo inicialmente como root.

Valor por defecto: User #-1

**Group**

Esta directiva define el identificador de grupo (group ID), bajo el cual Apache operará. Para utilizar esta directiva, el servidor debe estar corriendo inicialmente como root.

Valor por defecto: Group #-1

**Listen**

La directiva instruye a Apache el puerto por el cual estará escuchando por peticiones. Por defecto responde peticiones de todas interfaces IP.

Valor por defecto: 80

**ServerAdmin**

Define el correo electrónico del administrador del servidor Web e indica la dirección a incluirse en los mensajes de error que serán enviados al cliente.

Es necesario tener habilitada la directiva **ServerSignature Email**

**DocumentRoot**

Esta directiva define la ruta absoluta dónde se almacenarán los archivos HTML que se desean publicar.

Valor por defecto: Compilado: /usr/local/apache/htdocs

Instalado: /var/www/html

**ServerRoot**

La directiva ServerRoot define la ruta absoluta dónde reside el servidor. Generalmente contiene los directorios conf y logs.

Valor por defecto: /usr/local/apache

**MinSpareServers**

Esta directiva establece el número mínimo de procesos servidores ociosos que pueden estar corriendo. Estos procesos están a la espera de peticiones para atenderlas; si hay menos procesos servidores corriendo de los que establece MinSpareServers se crean nuevos procesos a razón de uno por segundo.

**MaxSpareServers**

Esta directiva establece el número máximo de procesos servidores ociosos que pueden estar corriendo. Si hay más procesos de los que esta directiva establece, dichos procesos se matan.

**StartServers**

Define el número de procesos servidores que se crearán cuando se inicia Apache

Valor por defecto: 5

**MaxClients**

La directiva MaxClients define el número máximo de peticiones que podrán ser atendidas simultáneamente.

Valor por defecto: 20

**ErrorDocument**

En caso de que un error ocurra con Apache cuando se solicite un recurso, éste puede ser configurado para que realice una de las siguientes acciones:

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

Ejemplos de errores son los siguientes:

```
ErrorDocument 500 http://foo.example.com/cgi-bin/tester
ErrorDocument 404 /cgi-bin/bad_urls.pl
ErrorDocument 401 /subscription_info.html
ErrorDocument 403 "Sorry can't allow you access today"
```

Algunos códigos de estatus para HTTP 1.1 se presentan a continuación:

**200** OK
**102** Processing
**204** No content
**400** Bad Request
**404** Not Found
**403** Forbidden
**405** Method not allowed
**505** HTTP version not supported
**507** Insufficient storage

**PidFile**

Esta directiva define el archivo dónde el servidor almacena el identificador del proceso padre o httpd
Archivo por defecto: ***logs/httpd.pid***

**Bitácoras – Logging**

Apache cuenta con dos archivos donde residen las bitácoras:

> Accesos: ***access.log***

Registra todos los accesos al servidor.

> Errores: **error.log**

Registra los errores que genere un acceso al servidor o que los procesos de Apache reporten.

**ErrorLog** <*filename*>

Esta directiva define el nombre del archivo en el cual el servidor registrará los errores que se presenten. Si la ruta del archivo no es absoluta (desde raíz /), se considera en relativa a ServerRoot.

**LogLevel** <*nivel de error*>

Define el nivel de mensajes de error que serán registrados en la bitácora definida por ErrorLog. Cuando un nivel es especificado, se reportan los mensajes de todos los niveles de mayor importancia incluyendo el nivel definido.

A menos que la bitácora de Apache sea manejada por el syslog, el nivel `notice` no puede ser suprimido y siempre será enviado a la bitácora.

Los niveles de mensajes de error se muestran en la siguiente tabla:

| Nivel | Descripción | Ejemplo |
|---|---|---|
| `emerg` | Emergencias – el sistema es inusable. | "Child cannot open lock file. Exiting" |
| `alert` | Una acción debe ser tomada inmediatamente. | "getpwuid: couldn't determine user name from uid" |
| `crit` | Condiciones críticas | "socket: Failed to get a socket, exiting child" |
| `error` | Condiciones de error | "Premature end of script headers" |
| `warn` | Condiciones de advertencia. | "child process 1234 did not exit, sending another SIGHUP" |
| `notice` | Normal pero condición significativa | "httpd: caught SIGBUS, attempting to dump core in ..." |
| `info` | Informativo. | "Server seems busy, (you may need to increase StartServers, or Min/MaxSpareServers)..." |
| `debug` | Mensajes de nivel de depuración | "Opening config file ..." |

Se recomienda utilizar cuando menos un nivel `crit`.

**CustomLog** *<filename>*

Esta directiva define el nombre del archivo en el cual el servidor registrará las peticiones que reciba.

**LogFormat** “format-string” <*nickname*>

Especifica el formato del archivo de registros de accesos al servidor. El formato del argumento es una cadena. Esta cadena es usada para registrar cada petición en el archivo de registros.

Para registrar las características de la petición se incluye un signo de “%” para cada una, en la cadena de formato, las cuales son reemplazadas en el archivo de registro por los valores siguientes:

**%h** = Registra la IP del cliente, hostname.

**%l** = Si el deamon (demonio) identd corre en el cliente, reporta la información que el identd devuelva.

**%u** = Si se requiere de un username y password para acceder, en este campo se registra.

**%t** = Fecha y hora de la petición en el formato: [día/mes/a_o:hora:minuto:segundo tzofffset].

**\"%r\"** = Recurso solicitado por el cliente, entre comillas, petición..

**%>s** = Un código de tres dígitos donde se muestra el valor devuelto al cliente, status.

**%b** = El número de bytes devueltos exceptuando encabezados.

**Formato por defecto*: “Format-string”: "%h %l %u %t \"%r\" %>s %b"**

Existe una serie de LogFormat preestablecidos:

º *Common* (por defecto)*

º *Refered*

º *Agent*

º *Combined*

**HostNameLookup [On|Off]**

Esta directiva habilita la resolución de nombres en el DNS para poder registrar nombres de hosts.

Valor por defecto: Off

**Directory & DirectoryMatch**

La directiva Directory permite aplicar otras directivas de forma específica a todos los recursos dentro de la ruta definida por "dir"; se deben de considerar rutas absolutas.

**Options**

Esta directiva controla qué características del servidor están disponibles para un directorio en particular.

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

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

=> *All*

Todas las opciones de activan excepto MultiViews (esta es la opción por defecto)

=> *ExecCGI*

La ejecución de scripts CGI's es permitida.

=> *FollowSymLinks*

Las ligas simbólicas son válidas dentro de este directorio.

**NOTA**: Aún cuando el servidor seguirá las ligas simbólicas, éste no cambiará de la ruta definida por la directiva Directory.

=> *Includes*

Permite el uso de Server-side includes.

=> *IncludesNOEXEC*

Server-side includes son permitidos, pero execCGI (ejecución de CGI's) son desactivados.

=> *Indexes*

Si una URL mapea a un directorio solicitado y no hay DirectoryIndex (index.html) en este directorio, entonces el servidor devolverá un listado de los archivos del directorio.

**CGI's**

La tecnología CGI "Common Gateway Interface" define un modelo de programación que puede ser implementado por múltiples lenguajes.

Los CGI's son programas que siguen un estándar definido, corren en el servidor, reciben parámetros desde el cliente y su salida es enviada al navegador. Fueron la primera alternativa para generar dinamismo a un sitio Web.

Existen un par de directivas para configurar la utilización de CGI's en el servidor Apache:

**ScriptAlias** <alias> <path-filesystem>

Convierte las solicitudes vía URL a la ruta absoluta donde residen los programas CGI. Esta directiva se utiliza tanto en ServerConfig como en VirtualHost.

**AddHandler** cgi-scripts .cgi .pl

Permite que determinada extensión sea relacionada a un evento en particular. En el caso de los CGI's lo que se indica es que las extensiones .cgi y .pl quedan identificadas como extensiones de scripts. Apache interpretará el archivo en lugar de sólo enviar el contenido al cliente.

**Server-Side Includes**

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

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

Las directivas que configuran esta opción son:

**AddType** text/html .shtml

Relaciona los archivos con extensión .shtml como aquellos que contienen etiquetas SSI.

**AddOutputFilter** INCLUDES .shtml

Habilita el filtro para los archivos .shtml.

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

Las directivas presentadas son sólo algunas con las que cuenta Apache, pero son las más importantes para poder configurar el servidor de forma básica.

## 3.4 CONTROL DE ACCESOS

Apache ofrece la funcionalidad al estilo "ACL", con lo cual es factible determinar las direcciones IP y los usuarios que tendrán acceso sobre recursos específicos del servidor Web. La configuración de acceso vía nombre de dominio o vía IP puede realizarse con ayuda de las siguientes directivas.

***Order***

Define el orden en que las directivas Allow y Deny serán implantadas.

***Allow***

Define los hosts que tendrán acceso a alguna área del servidor. El acceso puede ser controlado por nombre del host, dirección IP, rango de direcciones IP o por otras características de la petición del cliente guardadas en variables de entorno.

***Deny***

Define la restricción del acceso al servidor basado en nombre de host, dirección IP o variables de ambiente.

*Sintaxis:*

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

Permiten en conjunto definir un grupo de hosts a los que se les puede permitir o negar el acceso al servicio de Web.

La definición de los hosts que serán restringidos o permitidos se puede realizar mediante alguna de las siguientes formas:

- El nombre parcial de un dominio parcial.

```
.hackers.com.mx
```

- Una dirección IP.

```
200.38.166.1
```

- La pareja IP y mascara

```
10.2.0.0/255.255.255.0
```

- Una dirección de red definida por Classless Inter-Domain Routing (CIDR)

```
10.2.2.110/24
```

***order allow, deny***

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

***order deny, allow***

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

*Ejemplo*:

```
<Directory /usr/local/apache/htdocs/intranet>

      order allow,deny
      Allow from 192.168.0.0/255.255.255.0
      Deny from  all

</Directory>
```

En Apache es factible asignar un login y un password por usuario, quienes tendrán acceso sobre recursos específicos del servidor. Las directivas que se encargan del manejo de esta característica de Apache son las siguientes:

### *AuthUserFile <archivo de autenticación>*

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

*Ejemplo*: `/usr/local/apache/conf/.htpasswd`

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

### *AuthGroupFile <archivo de autenticación>*

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

*Ejemplo*: `/dev/null`

#### *AuthName "banner-string"*

Define el nombre de autorización para el recurso protegido, éste aparecerá como un mensaje en la caja de dialogo que solicita el password para acceder al recurso protegido.

#### *Require valid-user*

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

*Ejemplo*:

```
<Directory /usr/apache/htdocs/intranet >

       AuthUserFile /usr/apache/conf/claves/.htpasswd
       AuthGroupFile /dev/null
       AuthName "Bienvenidos a la Intranet"
       Require valid-user

</Directory>
htpasswd –C /usr/apache/conf/claves/.htpasswd  <username>
```

## 3.5 MANEJO DE SITIOS VIRTUALES.

El término Virtual Host[18] (Host Virtual o Sitio Virtual) se refiere a la práctica de correr más de un sitio Web (como www.miempresa1.com y www.miempresa2.com) en una sola máquina. Los sitios virtuales pueden ser basados en direcciones IP, es decir, se tienen diferentes direcciones IP para cada sitio Web, o basados en nombre, es decir, se tienen múltiples nombres corriendo en cada dirección IP. El hecho de que están corriendo en un mismo servidor físico no lo hace evidente para el usuario final.

Apache fue uno de los primeros servidores en dar soporte para sitios virtuales basados en IP y más tarde dio soporte para ambos, basados en IP y basados en nombre. Esta última variante es nombrada también basada en host (host-based) o no-IP (non-IP virtual hosts).

**Directivas de Configuración**

<**VirtualHost**> y <**/VirtualHost**>

Son usadas para encerrar un grupo de directivas que van a aplicar sólo para un sitio virtual en particular. Cualquier directiva que sea permitida en el contexto del sitio virtual puede ser utilizada.

```
<VirtualHost addr[:port] [addr[:port]] ...>
      ...
</VirtualHost>
```

Cuando el servidor recibe una petición de un documento, de un sitio virtual en particular, utiliza las directiva de configuración encerradas entre las directivas <VirtualHost> y </VirtualHost>. El parámetro *addr* puede ser:

+ La dirección IP del sitio virtual.

+ El nombre de dominio completo para la dirección IP del sitio virtual.

+ El carácter *, que es usado en combinación con NameVirtualHost (`NameVirtualHost *`) para relacionar todas las direcciones IP ó

[18] http://httpd.apache.org/docs/2.2/en/vhosts/

+ La cadena _default_, la cual es usada sólo para sitios virtuales basados en dirección IP para cachar todas las direcciones que no corresponden.

*Ejemplo*:

```
<VirtualHost 10.1.2.3>
      ServerAdmin webmaster@host.foo.com
      DocumentRoot /www/docs/host.foo.com
      ServerName host.foo.com
      ErrorLog logs/host.foo.com-error_log
      TransferLog logs/host.foo.com-access_log
</VirtualHost>
```

Para direcciones que utilicen el protocolo IP versión 6, éstas deben especificarse dentro de paréntesis cuadrados, porque de otra manera el número de puerto que es opcional no se podría determinar.

*Ejemplo*:

```
<VirtualHost [2001:db8::a00:20ff:fea7:ccea]>
      ServerAdmin webmaster@host.example.com
      DocumentRoot /www/docs/host.example.com
      ServerName host.example.com
      ErrorLog logs/host.example.com-error_log
      TransferLog logs/host.example.com-access_log
</VirtualHost>
```

**NameVirtualHost**

Esta directiva es requerida si se quiere configurar un sitio virtual basado en nombre. Aunque el parámetro *addr* pueda ser el nombre del host, se recomienda que siempre se utilice una dirección IP.

Con la directiva NameVirtualHost se especifica la dirección IP a través de la cual el servidor va a recibir peticiones para el sitio virtual basado en nombre. Usualmente esta será la dirección a la cual el sitio virtual basado en nombre será resuelto. En los casos en que un firewall u otro proxy recibe las peticiones y las reenvía a una dirección IP diferente a la del servidor, se debe especificar la dirección IP de la interfase física que se encuentra en la máquina que va a servir las peticiones.

**NOTA**: El argumento de la directiva <VirtualHost> debe de coincidir exactamente con el argumento de la directiva <NameVirtualHost>

```
NameVirtualHost 1.2.3.4
<VirtualHost 1.2.3.4>
    # ...
</VirtualHost>
```

**ServerName**

Esta directiva se usa tanto para el servidor principal como para los hosts virtuales. Si se están utilizando hosts virtuales basado en nombres, la directiva ServerName dentro de la sección de la directiva <VirtualHost> especifica que nombre de host debe aparecer en el campo `Host:` de la cabecera en las peticiones para que coincida con el host virtual.

**ServerAlias**

Esta directiva establece los nombres alternativos para un host. Su uso es para hosts virtuales basados en nombre.

```
<VirtualHost *>
      ServerName server.domain.com
      ServerAlias server server2.domain.com server2
      # ...
</VirtualHost>
```

**ServerPath**

Esta directiva establece la ruta-URL (URL-path) para un host. Para uso con hosts virtuales basados en nombre.

*Ejemplo*:

```
ServerPath /path/to/file.html
```

**NOTA**: La URL-path representa una vista Web de un recurso.

## 3.6 INTRODUCCIÓN A LA SEGURIDAD EN CÓMPUTO

La seguridad en un sistema de cómputo es la característica que permite garantizar:

° Que se opera como se espera que lo haga
° Que es ajeno a todo riesgo
° Que es ajeno a toda amenaza
° Que no posee vulnerabilidades
° Que es confiable
° Que funciones sin fallo

Una amenaza es la circunstancia o evento que puede causar daño a un sistema. Frecuentemente aprovecha una vulnerabilidad. Por su parte una vulnerabilidad es la ausencia de una contramedida o debilidad de la misma. La debilidad puede originarse en el diseño, la implantación o en los procedimientos para operar y administrar el sistema. En seguridad informática se denomina “hoyo”.

El riesgo es un factor que se compone de tres elementos:

+ *Peligro*: Es la probabilidad de que se presente una amenaza.

+ *Grado de Exposición*: Nivel de afectación (número de personas, bienes, módulos del sistema) que podrían ser afectados.

+ *Vulnerabilidad*: Predisposición de un sistema a ser afectado por un agente perturbador.

Es la probabilidad de que una vulnerabilidad sea explotada de acuerdo a su nivel de exposición y al peligro involucrado.

**Riesgo = Peligro * Exposición * Vulnerabilidad**

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

## Políticas y Servicios de Seguridad

Una política de seguridad especifica las características de seguridad que una organización debe observar y proveer con el fin de salvaguardar su información.

Un servicio de seguridad es una característica que debe tener un sistema para satisfacer una política de seguridad. Estándares como el BS 7799 o el ISO 17799 consideran a la información como un activo y definen servicios para su protección.

*Servicios de Seguridad*

+ **Confidencialidad**

Solo el propietario es capaz de descifrar la información.

+ **Autenticación**

Se asegura de la identidad de la persona del otro lado de la línea. Se puede realizar mediante alguno de los siguientes mecanismos:

> "algo que se sabe"

Primeros sistemas de autenticación se basan en claves de acceso: nombre usuario, claves de acceso, nips, passwords, etc.

> "algo que se tiene"

Sistemas que se basan en dispositivos de acceso como tokens, key generators, etc.

> "algo que se es"

Son los más recientes sistemas de autenticación y se basan en la biometría. Distinguen las características físicas únicas de cada individuo como son la huella digital, el iris, la forma del rostro o de la palma de la mano, etc.

**+ Integridad**

Se asegura que la información se mantenga inalterada desde su creación.

**+ Autorización (Control de Acceso)**

Se asegura que la información estará disponible solo para determinados usuarios con la posibilidad de manejar diferentes niveles de acceso para cada uno de ellos.

**+ No Repudio**

Se asegura que el emisor de la información no puede negar haberla enviado.

**+ Disponibilidad**

Asegurar que el sistema este disponible cuando se le requiera.

**+ Auditoria**

Asegurar que se defina un registro cronológico de los eventos exitosos y fallidos, que proporcionen evidencia de la actividad del sistema.

## Criptología

La seguridad en cómputo se apoya en la Criptografía que se encarga de convertir, a través de algoritmos, un texto normal y comprensible en un formato incomprensible a menos que se posea un cierto conocimiento secreto. En épocas recientes la Criptología se define como la ciencia de usar las matemáticas para cifrar y descifrar información. El criptoanálisis por su parte es la ciencia de analizar y romper la comunicación segura entre dos puntos mediante el análisis del algoritmo empleado.

La Criptología involucra tanto *Criptoanálisis* como *Criptografía*.

Los algoritmos criptográficos pueden estar basados en Criptografía Simétrica o Criptografía Asimétrica.

*Criptografía Simétrica*

Requiere que el emisor y el receptor compartan una clave secreta “Llave”, la cual es utilizada para cifrar el mensaje cuando se envía y descifrar el mensaje al recibirlo. El gran inconveniente se presenta en el proceso de intercambiar de forma segura la Llave. En este esquema de seguridad es importante que el emisor disponga de un canal seguro para realizar la entrega de la llave al inicio de la transmisión.
Ejemplos de estos algoritmos son: DES, 3DES, AES, Blowfish e IDEA.

*Criptografía Asimétrica (de Llave Pública)*

Resuelve el problema de intercambiar el secreto (Llave), utilizando para ello un algoritmo basado en dos llaves, tanto el emisor como el receptor utilizan un par de llaves una “pública” y otra “privada”. La privada es mantenida en secreto por un individuo, la pública esta disponible para que otro individuo sea capaz de cifrar la información importante con ella. Cuando el emisor desea enviar un mensaje lo cifra con la llave pública del receptor, siendo este el único capaz de descifrarlo mediante el uso de su llave privada. En este caso se consigue asegurar la confidencialidad del mensaje.

Si el emisor cifra con su llave privada, todo aquel que disponga de la llave pública puede conocer el mensaje, por lo que el objetivo en este caso no es asegurar confidencialidad, sino que se obtiene una Firma Digital. Bajo este escenario se obtiene:

+ *Autenticación*, dado que sólo el propietario de la llave privada pudo haber generado el mensaje.

+ *Integridad*, considerando que el mensaje requiere llegar sin cambios para que el algoritmo de descifrado pueda operar.

+ *No repudio*, ya que la llave privada está sólo en posesión del emisor, por lo que no puede negar su autoría.

Rivest Shamir Adleman (RSA) y Diffie-Hellman (D-H) son dos sistemas de llaves públicas utilizados actualmente.

*Message Digest*

Conocidos también como Hashes, checksums son el resultado de algoritmos unidireccionales que permiten generar una "huella digital" de un mensaje. Permiten obtener una cadena de longitud fija que es una representación condensada de un mensaje. Permite garantizar la integridad de un mensaje, si un solo carácter del mensaje original cambia el Hash será totalmente distinto.
Ejemplos de algoritmos que generan estas cadenas son: SHA1, SHA2, MD5.

En la actualidad existen herramientas de cifrado que emplean los algoritmos que se han mencionado como ejemplo. Algunas de estas herramientas son:

*GPG*

Es un sistema de cifrado de código libre. Es un reemplazo de PGP (Pretty Good Privacy) originalmente creado por Phil R. Zimmermann. Esta herramienta se puede obtener libremente de http://www.gnupg.org/(en)/download/index.html.

*OpenSSL*

El proyecto OpenSSL es un desarrollo de código abierto con el propósito de desarrollar un toolkit de SSL y TLS. Se basa en el código fuente de SSLeay, desarrollado por Eric A. Young y Tim J. Hudson. Esta herramienta se puede descargar de http://www.openssl.org/source/openssl-0.9.7e.tar.gz.

**Seguridad en Apache**

Para implantar un buen nivel de seguridad en el servidor se recomiendan los siguientes puntos:

=> Mantener en producción una versión actualizada.
=> Evitar el uso de permisos no necesarios en los directorios de apache.
=> Restringir el uso de SSI y de CGI's.
=> Revisar con frecuencia las bitácoras
=> Seleccionar passwords fuertes para los accesos restringidos.
=> Implementar el soporte para SSL (mod_ssl)
=> Agregar módulos orientados a incrementar el nivel de seguridad
(mod_access, mod_auth, mod_security)

*mod_security*

- Filtra las peticiones (request) que recibe el servidor de Web.
- Analiza el contenido de las peticiones
- Busca evitar ataques de SQL Injection.
- Permite correr Apache en un ambiente Jail.
- Analiza las peticiones e implanta técnicas anti-evasion.

Además se dispone de las directivas ***Order***, ***Allow*** y ***Deny*** para permitir o negar el acceso al servidor.

**SSL**

**SSL** (*Secure Sockets Layer*) es un protocolo diseñado por la empresa *Netscape Communications*, que permite cifrar la conexión, incluso garantiza la autenticación. Se basa en la criptografía asimétrica y en el concepto de los certificados. La versión estandarizada por el IETF se conoce como TLS.

Su mayor ventaja es que funciona entre la capa de transporte y la capa de aplicación, según el modelo OSI, por esto es muy fácil usarlo para proteger los protocolos de la capa de aplicación (por ejemplo FTP, gopher, HTTP) sin tener que realizar cambios importantes en los mismos.

***Objetivos de SSL:***

**Seguridad de la Información:**

- SSL garantiza que terceros no tengan acceso a la información mientras viaja por internet al encriptarla.

**Integridad de los datos:**

- La información recibida desde un servidor por SSL puede ser "validada" para comprobar que no ha sido alterada en la trayectoria.

**Autenticidad de los Datos:**

- Mediante los algoritmos de encriptación, es posible comprobar que los datos realmente han llegado del servidor que el cliente espera. Esto evita que alguien se haga pasar por un sitio para cometer fraudes. (evitando ataques como Phishing, Man in the Middle, etc.).

## HTTPS

Versión segura del protocolo HTTP. El sistema HTTPS utiliza un cifrado basado en las Secure Socket Layers (SSL) para crear un canal cifrado (cuyo nivel de cifrado depende del servidor remoto y del navegador utilizado por el cliente) más apropiado para el tráfico de información sensible que el protocolo HTTP. El puerto estándar para este protocolo es el 443.

### Certificados Digitales

Los certificados digitales son una forma de agregar un tercer "árbitro" a la cadena de confianza de la comunicación por SSL. Lo que un certificado digital hace es agregar el "endoso" de un tercero que garantiza la integridad, y existencia de la organización que envía los datos. Esto significa que una autoridad certificadora avala que la empresa que es dueña del sitio Web, por ejemplo, realmente existe.

### Ataques

Acciones que tienen por objetivo el que cualquier parte de un sistema de información automatizado, deje de funcionar de acuerdo con su propósito definido. Esto incluye cualquier acción que causa la destrucción, modificación o retraso del servicio no autorizado. Generalmente no se refieren a un ataque físico (aunque puede ser); no se realizan en un solo paso, pero esto depende de los objetivos del atacante.

Existen dos tipos de ataques, los **Ataques Pasivos** y los **Ataques Activos**. Los **Ataques Pasivos** se refieren a intervenciones que ponen en riesgo la confidencialidad de la información, a través de la observación o utilizando herramientas para recopilar información en bases de datos, para generar listados de direcciones IP y nombres de dominio, para el trazado de la ruta que sigue la información o para analizar el tráfico de una red.

Los Sniffers son herramientas para efectuar ataques pasivos y consisten en un proceso que "olfatea" el tráfico que se genera en una red; es capaz de leer toda la información que circule en el segmento de red en el que se ubique. Los datos que se obtienen de este proceso son tramas de red que transportan paquetes IP, IPX, etc. En estos paquetes se incluyen datos de capas superiores, entre ellos los de la capa de aplicación, que en un momento dado pueden ser claves de acceso.

Existe una variación de los sniffers llamado analizador de protocolos, el cual tiene implantada funcionalidad suficiente como para entender y traducir los protocolos que se están empleando en la red.

Dos tipos de sniffers destacan, los pasivos que no realizan actividad alguna además que capturar paquetes y los activos, que utilizan técnicas para apoderarse de las sesiones.

La detección y combate de sniffers es difícil ya que son programas pasivos que no generan bitácoras y que cuando de usan apropiadamente utilizan un mínimo de CPU y memoria. A nivel local es posible su localización verificando si se están ejecutando. A nivel red algunos pueden localizarse mediante herramientas como ***antisnif***.

Los **Ataques Activos** incluyen alguna modificación del mensaje (información) o la creación de mensajes falsos. Existen varios tipos de ataques activos:

- *Cambiar la identidad del emisor o receptor. Ocurre cuando una entidad pretende hacerse pasar por otra.*

- *Manipulación de datos.*

- *Repetición. Capturar una información, guardarla un tiempo y volverla a enviar, produciendo un efecto no autorizado.*

- *Denegación de servicio. Impedir una comunicación, una respuesta, causar un repudio de usuarios.*

- *Encaminamiento incorrecto. Atacan a los nodos dentro de la red*

El escaneo de puertos es un método de ataque activo que consiste en conectarse a los puertos donde trabajan los protocolos TCP y UDP del sistema objetivo de ataque con el propósito de determinar cuales se encuentran escuchando. Una herramienta libre para realizar este ataque es ***Nmap***.

El ataque de fuerza bruta (Brute-Force Attack) busca probar todas las posibles opciones, se recorre por completo el universo con la intención de tener acceso a un sistema o descifrar cierta información. ***Hydra*** es una herramienta para realizar ataques de fuerza bruta.

Los ataques por diccionario sólo prueban aquellas opciones con mayor probabilidad de éxito, las cuales se encuentran en un diccionario. El primer script para ataques por diccionario fue ***John The Ripper***; su principal objetivo es obtener passwords débiles del archivo *passwd* de Unix.

El secuestro de sesiones es un tipo de ataque en el que el atacante toma control de una comunicación, tal y como sería, por ejemplo, en un secuestro aeronáutico. Interfiere cuando dos entidades se están comunicando haciendo pasar por una de ellas. Este es un tipo de ataque "man-in-the-middle". El objetivo es robar una conexión generada por una aplicación de red iniciada por un cliente (p. ejem. una sesión telnet).

Otro ejemplo de ataque es *DoS* (Denial of Service); busca generar una alta demanda sobre un servicio válido con el objetivo de saturar la capacidad de respuesta y propiciar una caída del sistema victima. Una variante de este ataque es el *DDoS* que es un ataque *DoS* distribuido. En el año dos mil varias empresas que apoyan su estrategia de negocios en Internet sufrieron un ataque *DDoS*. Algunas de estas empresas fueron CNN (agencia de noticias), Amazon (Venta de artículos diversos), e-Bay (venta de artículos en remate), e-Trade (compra y venta de acciones) y Yahoo (correo); donde esta última estimó pérdidas por medio millón de dólares al dejar de dar servicio por tres horas.

Los siguientes ejemplos son tipos de ataques *Dos*:

+ *Ping de la muerte*

Se manda un paquete que excede el tamaño máximo (65,535 bytes de datos), permitido por la especificación del protocolo IP, a un equipo victima; si el equipo es susceptible a este ataque puede sufrir un reinicio.

+ *Inundación Sync*

Se lanzan una serie de paquetes SYNC hacia el equipo destino, el cual responde con un SYNC-ACK y espera un ACK de parte del equipo origen, el atacante continua mandando SYNC, pero no manda ACK con lo cual sobrepasa la capacidad del equipo victima de recibir nuevas conexiones.

+ *Spoofing*

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

+ *Smurf*

Es uno de los ataques Dos más temidos. Requiere tres actores: La victima, el atacante y la red amplificadora.

El atacante origina un paquete ICMP hacia la dirección de broadcast de la red amplificadora, haciendo aparecer que su origen es una interfaz de la red de la víctima. Cada interfaz de la red amplificadora enviará respuestas a la supuesta interfaz de origen originando un trafico que la victima no es capaz de soportar.

Los virus forman parte de los ataques activos. Se definen como una porción de código de programación cuyo objetivo es implementarse a si mismo en un archivo ejecutable y multiplicarse sistemáticamente de un archivo a otro.

Además de esta función primaria de "invasión" o "reproducción", los virus están diseñados para realizar una acción concreta en los sistemas informáticos sin la autorización del usuario. Existen variantes relacionadas con los virus, pero que en realidad son conceptualmente diferentes a estos. Algunos antivirus pueden detectarlos. Estas variantes son:

| | | |
|---|---|---|
| ° Troyanos | ° Gusanos | ° Bomba lógica |
| ° Spyware | ° Adware | ° Exploit |

**Firewalls**

De manera general el firewall examina el tráfico de la red tanto entrante como saliente y en base a ciertos criterios determina si deja o no pasar los paquetes de información. Si detecta algo anormal puede tener procedimientos a seguir o poner en aviso al administrador.

Existen dos tipos básicos de firewalls: Hardware Firewall y Software Firewall.

El firewall por Hardware usualmente es un ruteador o switch capa3 y dispone de ciertas reglas para dejar o no dejar pasar los paquetes.

El firewall por software es un programa que corre preferentemente en un bastioned host o en un appliance, que verifica los paquetes con diferentes criterios para dejarlos pasar o descartarlos.

Los siguientes son algunos tipos de firewalls:

+ *Packet Filters*

En este tipo de firewall el tráfico es filtrado en base a reglas específicas, incluyendo direcciones fuente y destino, tipo de paquete, número de puerto, etc.

+ *Circuit Level Gateways*

El tráfico en este tipo de firewall es filtrado en base a reglas de sesión específicas, por ejemplo, cuando una sesión es iniciada por una computadora reconocida. El trafico que no es reconocido sólo se le permite el paso hasta la capa 4 del modelo TCP/IP.

+ *Application Level Gateways*

Para este tipo de firewall el tráfico es filtrado en base a reglas específicas de aplicación, es decir, aplicaciones como un navegador Web; o un protocolo como FTP o combinaciones.

+ *Stateful Multilayer Inspection Firewall*

En este tipo de firewall el tráfico es filtrado en tres niveles, basado en un amplio rango de reglas específicas de aplicación, sesión y filtrado de paquetes.

**Implantación de un Firewall en Linux con IPTables**

Para implantar un firewall en Linux se debe determinar:

+ El nivel de seguridad requerido.
+ El tráfico que va a entrar en la red.
+ El tráfico que va a salir de la red.

IPTables y Netfilter son las dos piezas principales de productos firewall disponibles gratuitamente para distribuciones Linux. IPTables es usado para construir las reglas, en tanto que Netfilter es puente entre el kernel de Linux e IPTables.

Con IPTables se puede realizar:

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

IPTables se puede utilizar para robustecer la seguridad de un servidor Web Apache.

**Proxy Server**

Un servidor proxy es un intermediario entre red la interna de una institución, por ejemplo, e Internet; puede implantar ciertos criterios de seguridad, así como también caché, lo cual significa que si el Proxy recibe una petición lo primero que hace es buscar en su caché y si encuentra el recurso que se le solicitó responde a la petición.

**Seguridad en PHP**

PHP es un poderoso lenguaje e intérprete, ya sea incluido como parte de un servidor Web en forma de módulo o ejecutado como un binario CGI separado, es capaz de acceder a archivos, ejecutar comandos y abrir conexiones de red en el servidor. Estas propiedades hacen que cualquier cosa que sea ejecutada en un servidor Web sea insegura por naturaleza. PHP está diseñado específicamente para ser un lenguaje más seguro para escribir programas CGI que Perl o C, y con la selección correcta de opciones de configuración en tiempos de compilación y ejecución, y siguiendo algunas prácticas correctas de programación, PHP puede dar la combinación precisa de libertad y seguridad que se necesite.

Ya que hay muchas maneras de utilizar PHP, existen varias opciones de configuración diseñadas para controlar su comportamiento. Un amplio rango de opciones garantizan que se pueda usar PHP para muchos propósitos distintos, pero también quiere decir que hay combinaciones de éstas opciones y configuraciones de servidor que pueden resultar en un entorno inseguro.

El nivel de flexibilidad en la configuración de PHP se compara quizás solo con su flexibilidad de desarrollo. PHP puede ser usado para escribir aplicaciones completas de servidor, con todo el poder de un usuario de un intérprete de comandos, o puede ser usado para inclusiones simples del lado del servidor con muy poco riesgo en un entorno minuciosamente controlado. Cómo construir ese entorno, y qué tan seguro es, básicamente depende del desarrollador PHP.

Para comprender mejor la manera de aplicar seguridad en PHP se toma ejemplo de dos directivas de configuración: *register_globals* y *magic_quotes_gpc*.

*register_globals* permite que las variables recibidas por el código PHP desde un formulario HTML, automáticamente se conviertan en variables. Desde PHP 4.2.0 está directiva está deshabilitada (off).

Facilita el uso del lenguaje ya que es posible acceder a los valores del formulario con sólo hacer referencia al nombre de la forma ($<nombreForma). Presenta una alta inseguridad, por ejemplo, en el caso de que un usuario malicioso envíe un parámetro al navegador, éste lo reciba y se declare por consiguiente una variable en el código PHP con ese nombre.

*magic_quotes_gpc*, con esta directiva, PHP de forma automática escapa todas las comillas simples -'-, comillas dobles -"-, backslashs -\- y caracteres nulos que reciba de parte del cliente Web. Con esto se busca evitar que se inserte código dañino a un sistema Web. Si el administrador deshabilita la funcionalidad (off), podría hacer sumamente vulnerable el código del sistema.

**Ataques en Web**

Entre los principales ataques sobre sistemas Web se encuentran:

+ *SQL Injection*

+ *Cross Site Scripting – XSS*

SQL Injection es un ataque que aprovecha la mala validación de las entradas del usuario para inyectar instrucciones SQL que no corresponden a las realizadas de forma normal por el sistema. Esto da un acceso no permitido sobre la base de datos. El problema consiste en que se atenta contra la integridad y confidencialidad de la información que reside en la base de datos víctima.

Cross Site Scripting busca presentar un contenido distinto al original a un visitante específico de un sitio. No modifica realmente el sitio, solo lo muestra de forma distinta, es decir, permite mostrar información falsa de un sitio auténtico. Y esto atenta contra la integridad de la información que publica el sitio.

**Seguridad en MySQL: Cifrado de passwords**

Con gran frecuencia los administradores olvidan configuraciones importantes de MySQL, lo cual puede poner en riesgo la información almacenada. Un problema común es cuando se le asigna un password a la cuenta de root:

```
mysql> UPDATE mysql.user SET password=password('nuevo_password')
       WHERE user ='root';

mysql> flush privileges
```

MySQL dispone de soporte para cifrar la información que se almacena, ya sea passwords o incluso toda la base de datos.

+ La función *password* genera una cadena de 41 bytes de longitud basada en un algoritmo doble SHA-1

```
mysql> select password('secreto'),
+-------------------------------------------------------------------+
| password('secreto')                                               |
+-------------------------------------------------------------------+
| *B03CEEAC29B017382BA994FCF6D62F6823D1E034                         |
+-------------------------------------------------------------------+
1 row in set (0.00 sec)
```

+ La función md5 genera una cadena de128 bits de longitud basada en el algoritmo de cifrado MD5.

```
mysql> select md5('secreto');
+----------------------------------+
| md5('secreto')                   |
+----------------------------------+
| e201994dca9320fc94336603b1cfc970 |
+----------------------------------+
1 row in set (0.08 sec)
```

+ La función *sha1* genera una cadena basada en el algoritmo SHA-1.

```
mysql> select sha1('secreto');
+------------------------------------------+
| sha1('secreto')                          |
+------------------------------------------+
| 0a4f8b93faad504007df78c9acb6f93ea6cc8c53 |
+------------------------------------------+
1 row in set (0.32 sec)
```

# CAPÍTULO IV

# DESARROLLO DEL SISTEMA DE ADMINISTRACIÓN DE CITAS

## 4.1 PLANTEANDO LA NECESIDAD DEL SISTEMA

Internet se desarrolló hace más de cuatro décadas, y su patrocinio estuvo a cargo del Departamento de Defensa de los Estados Unidos. Esta red fue diseñada en un principio para conectar los sistemas de cómputo principales de aproximadamente una docena de Universidades y organizaciones de Investigación. Hoy en día Internet está siendo utilizada por cientos de millones de computadoras alrededor del mundo.

Con la introducción de World Wide Web (que permite a los usuarios localizar y ver documentos basados en multimedia, sobre casi cualquier tema a través de Internet), Internet se ha convertido explosivamente en uno de los principales mecanismos de comunicación en todo el mundo.

Internet y la World Wide Web se encuentran, sin duda, entre las creaciones más importantes de la humanidad. Años atrás la mayoría de las aplicaciones de computadora se ejecutaban en equipos que no estaban conectados entre sí. Las aplicaciones en la actualidad pueden diseñarse para intercomunicarse entre los cientos de millones de computadoras en todo el mundo. Internet mezcla las tecnologías de la computación y las comunicaciones. Ofrece tener información accesible de forma instantánea y conveniente en todo el mundo. Hace posible que los individuos, empresas y pequeños negocios locales obtengan una exposición mundial; con esto se está cambiando la manera en que se hacen los negocios. La gente puede buscar los mejores precios de casi cualquier producto o servicio. Los investigadores pueden estar inmediatamente al tanto de los últimos descubrimientos.

En el mundo actual tanto las pequeñas, medianas y grandes empresas necesitan estar cerca de sus clientes y proveedores, en comunicación continua con ellos a través de medios eficaces y eficientes. Uno de estos medios es Internet, que a lo largo de sus años de crecimiento se ha convertido en una herramienta en la cual basar las estrategias de negocio. Es por esto que el proyecto se enfoca en un sistema Web que se encargará de ofrecer una solución flexible y eficiente para los negocios. Su función se basa en proporcionar un esquema de administración de citas con el cual los clientes que necesiten solicitar una cita para recibir asesoría de su empresa preferida de Tecnología, por ejemplo, no necesiten levantar la auricular de su teléfono para hacer una llamada, ahorrando tiempo y costos de telefonía.

## 4.2 OBJETIVOS DEL SISTEMA

- ✓ ***Ofrecer una herramienta eficiente para la comunicación de las empresas con sus clientes y proveedores.***

- ✓ ***Promover el uso de la Web para los negocios.***

- ✓ ***Promover el uso de Linux y software libre como alternativas viables al software propietario que implica altos costos por licenciamiento.***

- ✓ ***Demostrar el nivel de robustez al que se puede llegar en aplicaciones desarrolladas con software libre.***

- ✓ ***Desarrollar un sistema profesional a bajo costo.***

## 4.3 REQUERIMIENTOS DEL SISTEMA

El sistema requiere ser implantado en un servidor Web para ser accedido por cualquier computadora a través de Internet. Se necesitará contar con una conexión de banda ancha. El equipo en el que se implantará el servidor Web tendrá las siguientes características:

- Procesador: AMD Athlon XP 2200+ a 1800 MHz.

- Disco duro: 40 GB de capacidad a 7200 RPM

- RAM: 768 MB DDR con ECC.

- Tarjeta de Red: 10/100 Integrada a la tarjeta Principal

- S.O.: Slackware Linux kernel 2.4.26

Cabe destacar que las características de este equipo son adecuadas para pruebas durante el desarrollo y para su puesta en marcha. Sin embargo en el caso de llevar el sistema a un ambiente empresarial donde el número de accesos al servidor sea mucho mayor, entonces se necesitará emplear un equipo con mayores capacidades de almacenamiento, procesamiento y memoria principal.

El sistema deberá contar con una interfaz amigable para el usuario, que se desarrollará con algún lenguaje de programación libre. Para almacenar los datos de los clientes que hagan sus citas se necesitará emplear un manejador de bases de datos.

## 4.4 DANDO SOLUCIÓN AL SISTEMA

El sistema va a estar disponible para los usuarios en un sitio Web. Los usuarios se podrán registrar en el sistema, debiendo proporcionar un nombre de usuario (*login*) y contraseña (*password*). Esto con el fin de llevar un control de los accesos al sistema. Una vez que se han registrado los usuarios pueden acceder a la parte del sistema donde podrán configurar los datos de su cita, estos datos serán los siguientes:

- Depto. con el cual quiere hacer su cita
- Persona a la que va a visitar
- Asunto de la cita.

Una vez configurados estos datos, se le muestra al usuario una pantalla con el calendario del mes en curso para que seleccione la fecha de la cita y a continuación la hora de la cita.

El sistema permitirá registrar 3 citas por usuario y será capaz de mandar un aviso al usuario indicándole que su cita ha sido registrada y proporcionándole el ID de su cita, además de enviarle un correo electrónico de confirmación. Adicionalmente, el sistema se encargará de alertar de las citas del día a los empleados involucrados de cada departamento, a través de un correo interno.

Cualquier usuario del sistema puede hacer consultas de sus citas en donde se le mostrarán todas las que ha registrado.

## 4.5 ¿CUÁLES HERRAMIENTAS LIBRES SE VAN A EMPLEAR?

- Sistema Operativo Slackware Linux.
- Servidor Web Apache.
- Herramientas de desarrollo HTML y PHP.
- Manejador de Bases de datos MySQL.

## 4.6 INTEGRANDO LAS HERRAMIENTAS

Como se ha mencionado, las herramientas que se van a emplear para desarrollar e implantar este sistema son libres, desde el sistema operativo el cual va a manejar los recursos de hardware hasta las herramientas de desarrollo pasando por el servidor Web que permitirá acceder al sistema desde cualquier computadora a través de Internet y por el manejador de bases de datos que administrará el almacenamiento de los datos de los usuarios.

Ahora bien, todas estas herramientas han sido desarrolladas a través de los años para trabajar prácticamente juntas, de manera que la compatibilidad entre ellas es casi nata. Aunque de cualquier forma existen distribuciones del servidor Web Apache o del manejador de bases de datos MySQL para trabajar bajo ambientes Windows® pero se ha demostrado que su mejor desempeño de estas herramientas se logra bajo ambientes Linux.

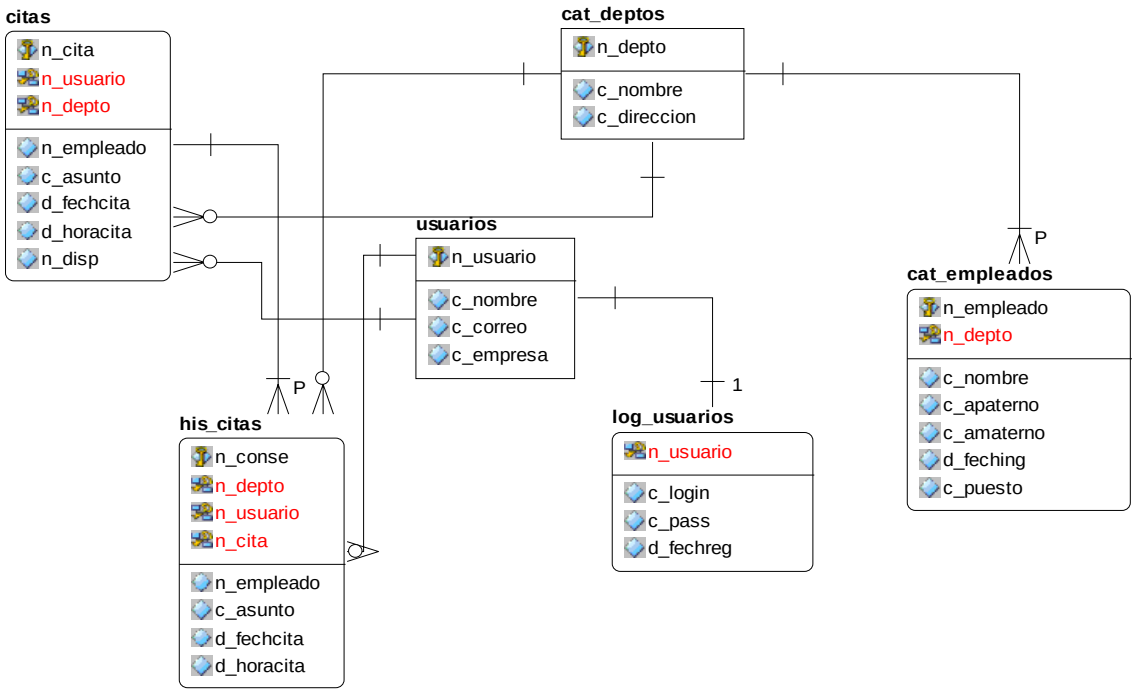
El primer paso de Integración de las herramientas será la instalación del Sistema Operativo en la máquina que va a fungir como el servidor. Se ha escogido la distribución Linux Slackware ya que esta es la que se ha venido empleando durante el desarrollo del presente trabajo, donde se han resaltado sus características. La instalación y configuración se llevará a cabo según lo visto en el Capítulo I. Una vez que se halla instalado el sistema operativo, se integrará el servidor Web Apache, el cual debe ser instalado con soporte para PHP y MySQL (Capítulo III).

## 4.7 DESARROLLO DEL SISTEMA

### Base de datos

La base de datos va a almacenar los datos de los usuarios como su nombre, correo electrónico y empresa, datos de las citas como fecha y hora, departamentos de la empresa a la que visita, login y contraseñas de los usuarios y algunos otros datos referentes a la citas.

*Modelado de la Base de Datos:*




*Creación de la Base de datos*

La base de datos se llamará ***db_citas***. Constará de las siguientes tablas:

- usuarios
- citas
- log_usuarios
- cat_deptos
- cat_empleados
- his_citas

**Estructura de Tablas:**

**usuarios:**

| **nombre_campo** | **tipo** | **null** | **llave** |
|---|---|---|---|
| n_usuario | integer | no | primary |
| c_nombre | varchar(50) | no | -- |
| c_correo | varchar(30) | no | -- |
| c_empresa | varchar(35) | yes | -- |

**log_usuarios:**

| **nombre_campo** | **tipo** | **null** | **llave** |
|---|---|---|---|
| n_usuario | integer | no | primary |
| c_login | varchar(15) | no | -- |
| c_pass | varchar(25) | no | -- |
| d_fechreg | datetime | no | -- |

**citas:**

| nombre_campo | tipo | null | llave |
|---|---|---|---|
| n_cita | integer | no | primary |
| n_usuario | integer | no | foreign |
| n_depto | integer | no | foreign |
| n_empleado | integer | no | foreign |
| c_asunto | varchar(30) | no | -- |
| d_fechcita | date | no | -- |
| d_horacita | time | no | -- |
| n_disp | integer | no | -- |

**cat_deptos**

| nombre_campo | tipo | null | llave |
|---|---|---|---|
| n_depto | integer | no | primary |
| c_nombre | varchar(15) | no | -- |
| c_direccion | varchar(25) | no | -- |

**cat_empleados**

| nombre_campo | tipo | null | llave |
|---|---|---|---|
| n_empleado | integer | no | primary |
| c_nombre | varchar(25) | no | -- |
| c_apaterno | varchar(30) | no | -- |
| c_amaterno | varchar(25) | no | -- |
| d_feching | date | no | -- |
| n_depto | integer | no | foreign |
| c_puesto | varchar(30) | no | -- |

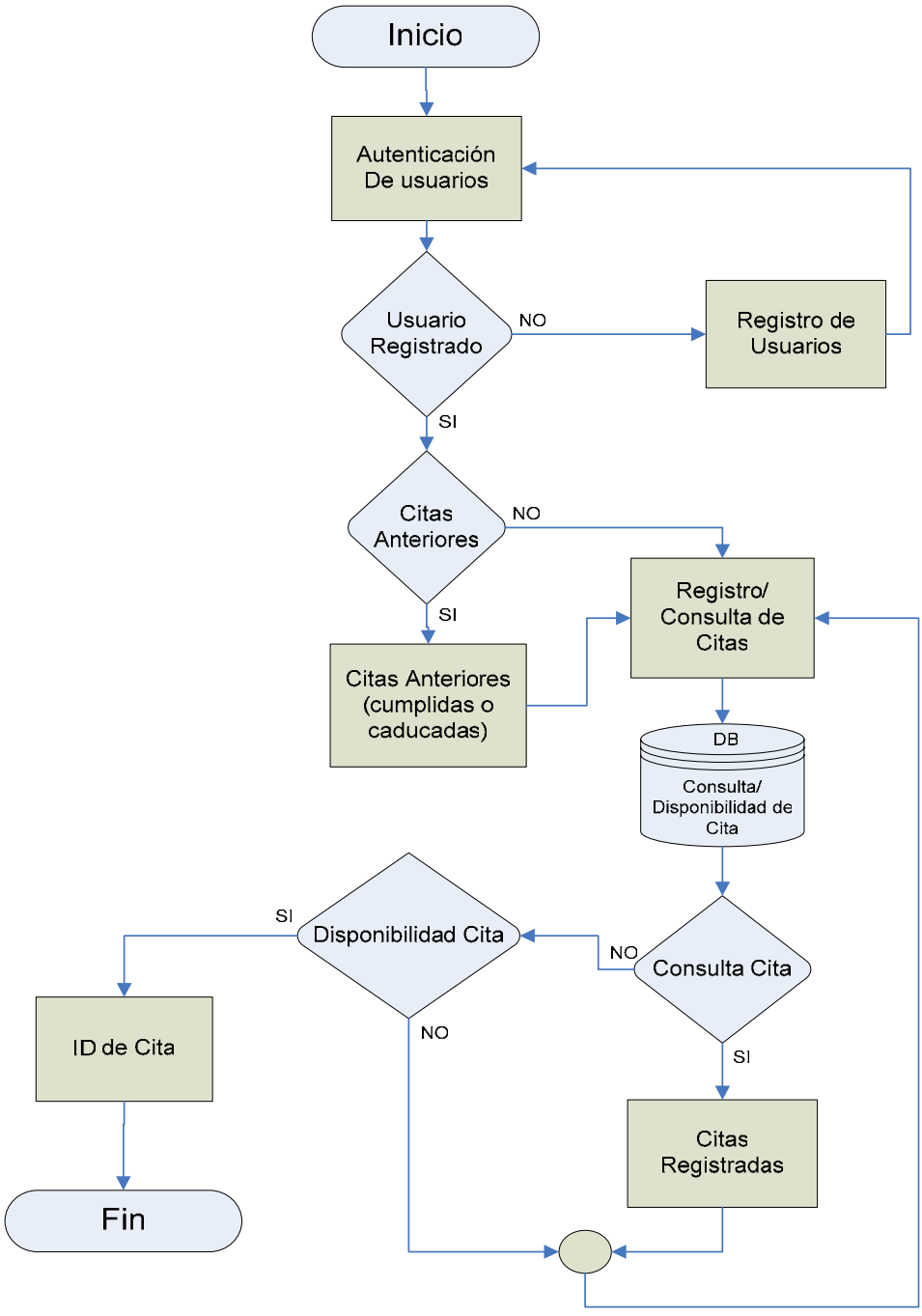
**his_citas**

| nombre_campo | tipo | null | llave |
|---|---|---|---|
| n_conse | integer | no | primary |
| n_cita | integer | no | foreign |
| n_usuario | integer | no | foreign |
| n_depto | integer | no | foreign |
| n_empleado | integer | no | foreign |
| c_asunto | varchar(30) | no | -- |
| d_fechcita | date | no | -- |
| d_horacita | time | no | -- |

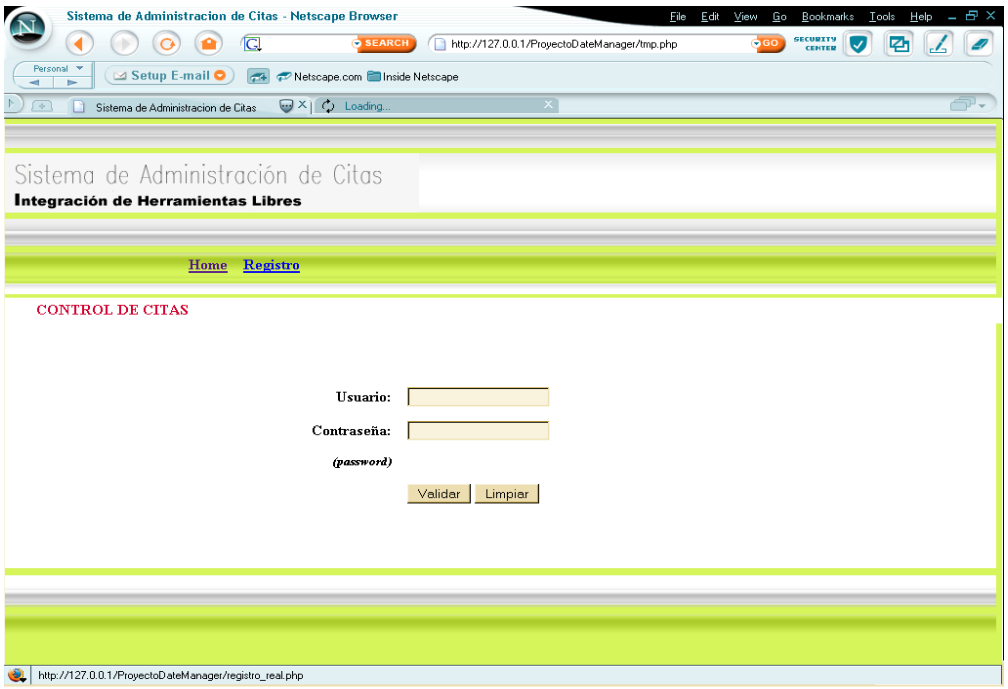
La nomenclatura que se utiliza para nombrar los campos de cada tabla está en función del tipo de dato que se almacenará en cada campo:

***tipodato**_nombre*

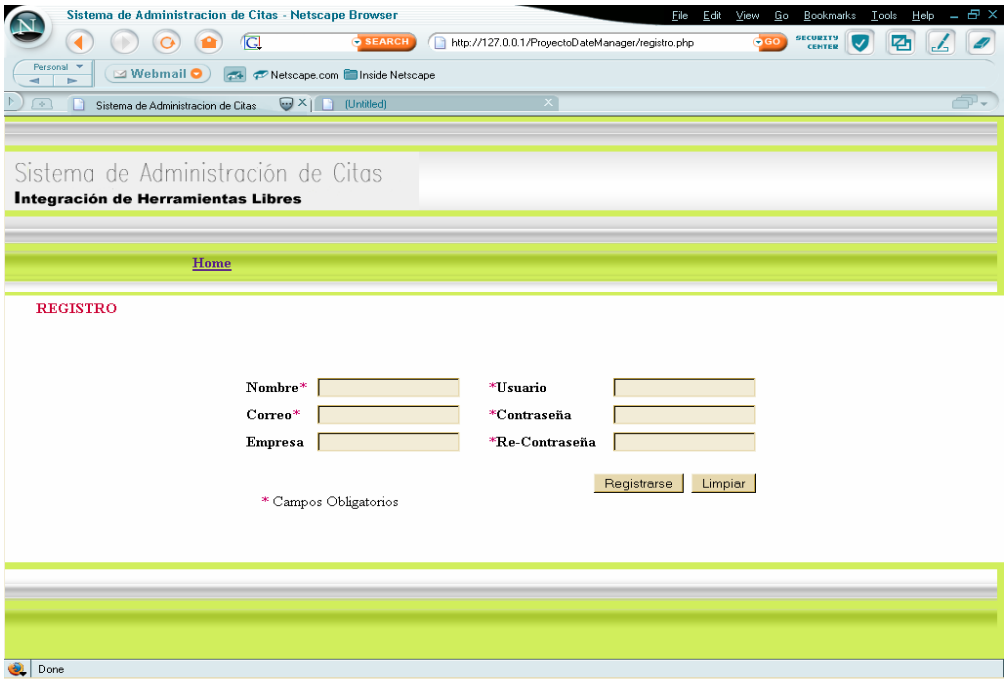
**Flujo del Sistema:**



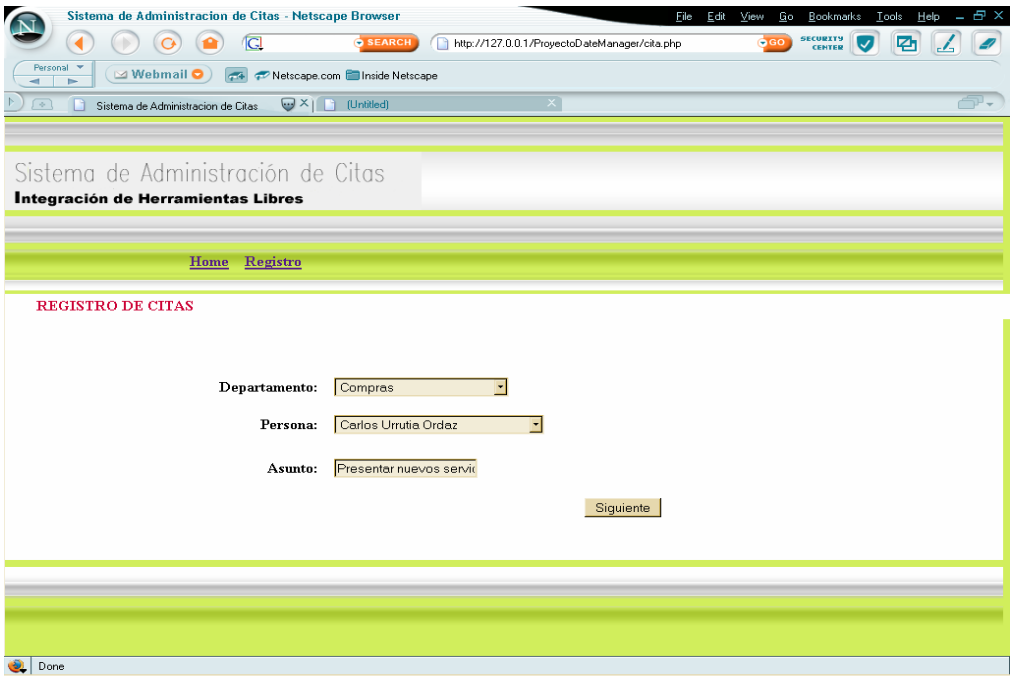
## Interfaz de Usuario




Esta es la pantalla inicial del sistema, donde los usuarios se autenticarán. En caso de que no se hayan registrado aún, la pantalla siguiente muestra el registro de usuarios:

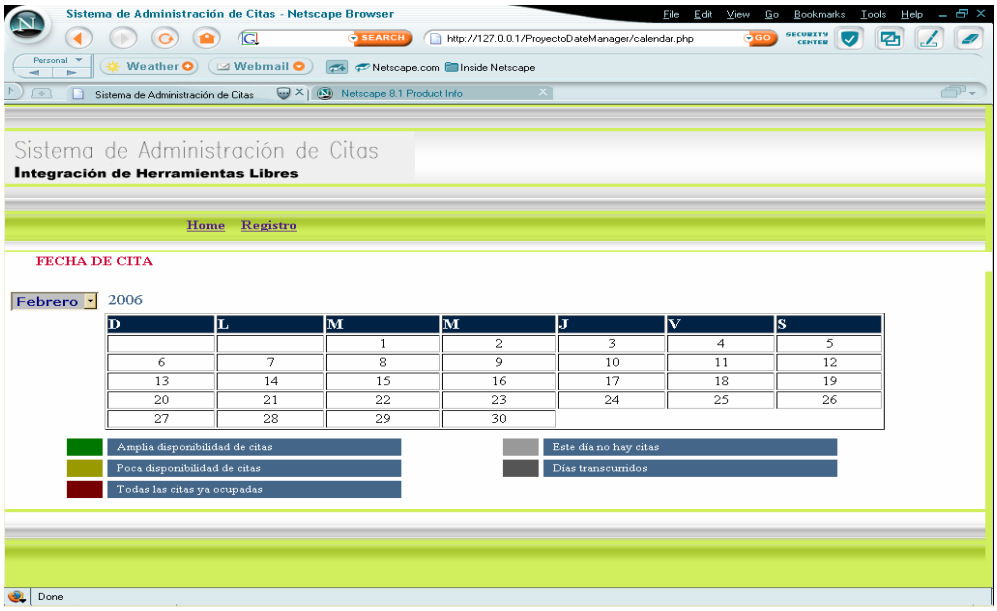


Una vez que se ha registrado el usuario, se le despliega de nuevo la pantalla inicial para que se autentique con su nombre de usuario y contraseña. Habiendo ingresado el usuario, se le muestra un formulario donde empezará a configurar su cita, eligiendo el departamento de la empresa a donde quiere hacer su cita, y la persona a la cual va a visitar, además del asunto por el cual desea hacer su cita:




El siguiente paso para el registro de la cita es el que se muestra en la siguiente pantalla, donde se muestra al usuario un calendario del mes en curso donde cada recuadro correspondiente a cada día del mes y se encuentra en un color dependiendo de las siguientes condiciones:

- Días transcurridos del mes
- Días en que no se otorgan citas
- Días con alta disponibilidad de citas
- Días con poca disponibilidad de citas
- Días sin disponibilidad de citas.




Una vez que el usuario he seleccionado la fecha para su cita, deberá seleccionar la hora para su cita, donde de igual manera cada recuadro tiene un color distinto indicando la disponibilidad de citas de acuerdo al horario:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 09:00 | 09:20 | 09:40 | 10:00 | 10:20 | 10:40 | 11:00 | 11:20 |
| 11:40 | 12:00 | 12:20 | 12:40 | 13:00 | 13:20 | 13:40 | 14:00 |
| 14:20 | 14:40 | 15:00 | 15:20 | 15:40 | 16:00 | 16:20 | 16:40 |

**NOTA:** Las horas que no están subrayadas no tienen disponibilidad de cita.

Ya que se haya registrado la cita, el usuario recibirá su IDentificador de cita, para que pueda el usuario reconfigurar más tarde su cita e incluso borrarla. En adición a esto se le enviará una correo electrónico al usuario indicándole todos los datos de sus cita, incluyendo su ID de cita.

**NOTA**: Por efectos de impresión las pantallas del sistema no presentan los colores que se utilizan para identificar la disponibilidad de citas.

La Interfaz de Usuario se apoya en la programación PHP que comprende varios archivos. Algunos de estos son: ***conexión_mysql.php***, ***funciones.php***, ***registro.php***, ***registro_real.php***, ***identificación.php***, ***dm1.php*** y ***cita.php***.

La conexión a la base de datos se realiza a través del archivo *conexión_mysql.php*:

```
<?
      $dbd = mysql_connect('localhost', 'root');

      if(!$dbd)
            echo "ERROR AL CONECTARSE";

      // Selección de la base de datos
      if(!mysql_select_db("db_citas", $dbd))
            echo "ERROR AL SELECCIONAR LA BASE DE DATOS";

?>
```

La librería de funciones se encuentra en el archivo funciones.php, al cual hacen llamar todos los demás archivos de la siguiente manera:

```
include ("funciones.php");
```

## CONCLUSIONES

Analizados los temas de los tres capítulos anteriores y el desarrollo del sistema en el cuarto capítulo, se puede concluir lo siguiente:

Las herramientas de Software Libre se encuentran a la par en características, funcionalidad, uso e implantación de aquellas propietarias, esto es, por las que se debe pagar un costo por su licencia de uso e implantación.

Al igual que con herramientas propietarias se desarrollan sistemas robustos y eficientes, con las herramientas libres también se pueden desarrollar estos y aún con mayores características en beneficio de la seguridad, confiabilidad, escalabilidad, compatibilidad y proyección.

Mientras siga habiendo un impulso continuo al movimiento de Software Libre, habrá mayor confianza en el uso de estas tecnologías, ya que la continuidad de este movimiento conlleva implícitamente al mejoramiento de estas herramientas.

Es importante que en nuestra casa de estudios se impartan cursos y diplomados como aquel en el cual se basa el presente trabajo, ya que al difundirse el uso de nuevas tecnologías se asegura que la Universidad Nacional Autónoma de México se encuentre a la vanguardia en cuanto conocimiento, técnica e investigación.

El Diplomado en Desarrollo e Implantación de Sistemas con Software Libre en Linux me aportó los conocimientos necesarios y junto con una profundización de cada tema me han permitido aplicarlos en el campo laboral, aunque sin duda hay aspectos del Diplomado que se pueden mejorar; el primordial de ellos es la actualización de contenidos, en donde a la par que se vaya impartiendo el diplomado, se desarrollen uno o varios sistemas enfocados al manejo real de información para los negocios.

## BIBLIOGRAFÍA


Hicks, Alan. Slackware Linux Essentials, Second Edition.
Brentwood, CA. USA 2005 Ed. Slackware Linux Inc.

Sæther Bakken, Stig. PHP Manual. PHP Documentation Group 2004

MySQL Reference Manual MySQL AB.

The Java Tutorial, Sun MicroSystems

Deitel, Harvey y Deitel, Paul. Java, Como Programar. 5a Ediciòn
Mèxico 2004 Ed. Pearson Educación

Laurie Ben and Laurie Peter, Apache: The Definitive Guide, Third Edition.
Sebastopol CA, USA 1999 Ed. O'Reilly & Associates Inc.

Argerich, Luis et al. Professional PHP4,
New York, N.Y. USA 2003 Ed. Apress

Kofler, Michael. The Definitive Guide to MySQL, Second Edition
New York, N.Y. USA 2004 Ed. Apress.

Converse, Tim; Park, Joyce; Morgan, Clark. PHP5 and MySQL Bible
Indianapolis, IN USA 2004 Ed. Wiley Publishing


## FUENTES


- http://www.linuxcordoba.org/node/73
- http://tldp.org/index.html
- http://perso.magic.fr/caliman/Guia_del_enROOTador-8.html
- http://es.tldp.org/Manuales-LuCAS/ENROOTADOR/html/Guia-del-enROOTador--2.8.html#toc14
- http://laurel.datsi.fi.upm.es/~rpons/personal/trabajos/lpractico/node153.html
- http://www.linuxplanet.com/linuxplanet/tutorials/2926/4/
- http://es.wikipedia.org/wiki/MySQL
- http://www.mysql-hispano.org/page.php?id=2
- http://www.mysql.com/products/what_is_mysql.html
- http://pisuerga.inf.ubu.es/lsi/Invest/Java/Tuto/II_7.htm
- http://www.desarrolloweb.com/articulos/1054.php?manual=34#numericos
- http://databases.about.com/cs/specificproducts/g/foreignkey.htm
- http://www.programacion.com/bbdd/tutorial/mysql_basico/

- http://searchoracle.techtarget.com/sDefinition/0,,sid41_gci902816,00.html
- http://www.netcraft.com.au/geoffrey/postgresql/fkey.html
- http://mysql.conclase.net/curso/index.php?cap=011
- http://www.javaworld.com/javaqa/2002-08/01-qa-0823-appvswebserver.html
- http://web-hosting.candidinfo.com/server-operating-system.asp
- http://news.netcraft.com/archives/web_server_survey.html
- http://httpd.apache.org/docs/2.2/en/install.html#extract
- http://httpd.apache.org/docs/2.2/en/mod/directives.html#E
- http://httpd.apache.org/docs/2.2/en/vhosts/
- http://www.iec.csic.es/criptonomicon/ssl.html
- http://www.pcm.gob.pe/portal_ongei/seguridad2_archivos/Lib5010/cap0203.htm
- http://www.php.net/docs.php

## GLOSARIO

***ACL***.- (*Access Control List*). Método para limitar el uso de recursos específicos a usuarios autorizados.

***APACHE***.- Servidor Web de libre distribución y es el más empleado actualmente en Internet para servir páginas.

***APPLET***.- Es un pequeño programa hecho en lenguaje Java que se descarga y ejecuta en un navegador Web.

***CGI***.- (*Common Gateway Interfase*). Esta tecnología define un modelo de programación que puede ser implementada en múltiples lenguajes. Son programas que siguen un estándar definido, corren en el servidor recibiendo parámetros desde el cliente y su salida es enviada al navegador.

***COOKIE***.- Pequeño archivo de texto (ilegible si se abre con algún procesador de textos) que se almacena en el disco duro del visitante de una página Web a través de su navegador a petición del servidor de la página. La información que contiene este archivo es recuperada por el servidor en posteriores visitas a la página.

***CPU***.- (*Central Processing Unit*). Es la Unidad Central de Procesamiento y se encarga de realizar todos los cálculos de una computadora incluyendo el manejo de algunos dispositivos. Se le denomina comúnmente como "*procesador*"

***DBMS***.- (*DataBase Management System*). Los Sistemas manejadores de bases de datos son los encargados de almacenar y posteriormente permitir el acceso a los datos de forma rápida y estructurada.

***DDL***.- (*Data Definition Language*). Es un lenguaje que permite especificar el esquema de la base de datos, a través de definir tablas, vistas, procedimientos almacenados y triggers.

***DML***.- (*Data Manipulation Language*). Es un lenguaje que permite la consulta y manipulación de datos. Este lenguaje al igual que el DDL son proporcionados por el sistema manejador de bases de datos.

***DoS***.- (*Denial of Service*). Es un tipo de ataque de seguridad en cómputo cuyo objetivo es generar una alta demanda sobre un servicio válido con el fin de saturar la capacidad de respuesta y propiciar la caída del sistema.

***DSO***.- (*Dynamic Shared Objects*). Es un mecanismo que provee una forma de construir parte del código de un programa en un formato especial para que pueda ser cargado en tiempo de ejecución en el espacio de direcciones de un programa ejecutable.

***FIREWALL***.- Es un mecanismo que puede ser físico o lógico que examina el tráfico de una red, y en base a ciertos criterios permite o no el paso de los paquetes de información.

***GNU***.- (*GNU No es Unix*). Fue el proyecto iniciado por Richard Stallman en 1984 con el objetivo de crear un sistema operativo libre que fuera compatible con UNIX.

***GPG***.- Es un sistema de cifrado de código libre que viene a ser un reemplazo de PGP creado por Phil R. Zimmermann

***GUI***.- (*Graphical User Interfase*). Es el método para facilitar la interacción de los usuarios con un programa o aplicación.

***HCL***.- (*Hardware Compatibility List*). Lista en la que se encuentran los dispositivos de hardware que son compatibles con las diferentes distribución de Linux.

***HTML***.- (*HyperText Markup Language*). Lenguaje estándar para la creación de documentos o páginas Web. Este lenguaje consta de etiquetas las cuales definen un comportamiento específico a la hora de ser interpretadas por un navegador Web.

***HTTP***.- (*HyperText Transfer Protocol*). Este protocolo permite la transferencia de hipertexto a través de la Web.

***HTTPS***.- Es la versión segura del Protocolo HTTP, utiliza un cifrado basado en SSL(ver SSL) para crear un canal cifrado apropiado para el tráfico de información sensible.

***INTERNET***.- Es el nombre que se le ha otorgado a la red mundial de redes que interconecta millones de computadoras entre si.

***IP, Dirección***.- (*Internet Protocol*). Las direcciones IP son un conjunto de números que definen la identidad única de una computadora en una red. Su estructura se establece por cuatro octetos separados por puntos.

***JAVA***.- Plataforma de software desarrollada por Sun Microsystems, que en un principio se pensó para dispositivos domésticos inteligentes, pero más tarde se observó su versatilidad para la Web.

***JVM***.- (*Java Virtual Machine*). La Máquina Virtual de Java es parte fundamental de la plataforma Java; se encarga de traducir los códigos de bytes (bytecodes) que se generan de la compilación de un programa a código que puede ser reconocido por cada plataforma de hardware en particular.

***LILO***.- (*Linux Loader*). Se encarga de arrancar el sistema operativo Linux e inclusive otros sistemas operativos que se encuentren en la misma máquina.

***MINIX***.- Sistema Operativo basado en Unix que fue escrito por el profesor Andrew Tanembaum

***MYSQL***.- Manejador libre de bases de datos que se ha popularizado por sus características incluyendo funcionalidad, compatibilidad y estabilidad.

***OOP***.- (*Object Oriented Programming*). La programación orientada a objetos es una metodología de diseño de software y un paradigma de programación que define un programa en términos de clases de objetos. Los objetos son entidades que combinan estado (datos) y comportamiento (procedimientos o métodos).

***OPENSSL***.- Es una implantación de código abierto de los protocolos SSL y TLS. El núcleo de librerías implantan las funciones básicas de criptografía y proveen varias funciones útiles

***PHP***.- (*PHP: Hypertext Preprocessor*). Es un lenguaje interpretado de alto nivel embebido en páginas HTML que se ejecuta en el servidor.

***PID***.- (*Process ID*). Es un identificador que distingue a cada proceso que se ejecuta en un S.O.; generalmente todos los sistemas operativos basados en UNIX utilizan este identificador.

***PROXY***.- Es un intermediario entre una intranet e Internet; puede implantar criterios de seguridad así como caché.

***RUN-LEVEL***.- El nivel de arranque es utilizado por el sistema operativo Linux para cargarse con determinadas características o bajo cierto modo. Los niveles de arranque van desde el nivel 0 hasta el 6.

***SISTEMA OPERATIVO***.- Conjunto de programas que se encargan de administrar los recursos de un sistema de cómputo; estos recursos incluyen hardware, software, comunicaciones, entre muchos otros.

***SNIFFERS***.- Herramientas para efectuar ataques pasivos, en donde un proceso "olfatea" el tráfico que se genera en una red, siendo capaz de leer toda la información que circule en el segmento de red en el que se ubique.

***SQL***.- (*Structured Query Language*).- Lenguaje estructurado de consulta que se ha convertido en un estándar en la industria. Está formado por un conjunto de sentencias que permiten la definición y la manipulación de datos.

***SSL***.- (*Secure Sockets Layer*). Protocolo diseñado por Netscape Communications que permite cifrar una conexión, incluso garantiza la autenticación. Se basa en la criptografía asimétrica y en el concepto de los certificados.

***SWAP***.- Es un área de intercambio de datos que se establece en el disco duro con el fin de mantener en memoria principal sólo los datos más utilizados de un programa y el resto de ellos almacenarlos en esta área.

***TCP/IP***.- (*Transmisión Control Protocol / Internet Protocol*).- Es un conjunto de protocolos de red implantados en una pila que permite la comunicación entre computadoras conectadas entre sí. Es la base sobre la que trabaja Internet, enlazando millones de computadoras que utilizan diferentes sistemas operativos. Fue desarrollado por el Departamento de Defensa de los Estados Unidos.

***UNIX***.- Sistema Operativo desarrollado por los laboratorios Bell en los años 60; sobre este sistema están finalmente basadas todas las distribuciones de Linux.

***URL***.- (*Uniform Resource Locator*). Es una secuencia de caracteres, de acuerdo a un formato estándar, que se usa para nombrar recursos, como documentos e imágenes en Internet, por su localización.

***WWW.-*** (*World Wide Web*). Servicio de Internet que permite la transferencia de documentos de hipertexto y multimedia.

***X WINDOW***.- Es el sistema que maneja la interfaz gráfica en un sistema operativo Linux.

# ANEXO I

## Tipos de datos en MySQL

## Tipos numéricos:

**TinyInt:** Es un número entero con o sin signo. Con signo el rango de valores válidos va desde -128 a 127. Sin signo, el rango de valores es de 0 a 255

**Bit ó Bool:** Un número entero que puede ser 0 ó 1

**SmallInt:** Número entero con o sin signo. Con signo el rango de valores va desde -32768 a 32767. Sin signo, el rango de valores es de 0 a 65535.

**MediumInt:** Número entero con o sin signo. Con signo el rango de valores va desde -8.388.608 a 8.388.607. Sin signo el rango va desde 0 a16777215.

**Integer, Int:** Número entero con o sin signo. Con signo el rango de valores va desde -2147483648 a 2147483647. Sin signo el rango va desde 0 a 429.4967.295

**BigInt:** Número entero con o sin signo. Con signo el rango de valores va desde -9.223.372.036.854.775.808 a 9.223.372.036.854.775.807. Sin signo el rango va desde 0 a 18.446.744.073.709.551.615.

**Float:** Número pequeño de coma flotante de precisión simple. Los valores válidos van desde -3.402823466E+38 a -1.175494351E-38, 0 y desde 1.175494351E-38 a 3.402823466E+38.

**xReal, Double:** Número de coma flotante de precisión doble. Los valores permitidos van desde -1.7976931348623157E+308 a -2.2250738585072014E-308, 0 y desde 2.2250738585072014E-308 a 1.7976931348623157E+308

**Decimal, Dec, Numeric:** Número de coma flotante desempaquetado. El número se almacena como una cadena

La siguiente tabla muestra una relación del tipo de campo con el tamaño de almacenamiento que permite:

| Tipo de Campo | Tamaño de Almacenamiento |
|---|---|
| TINYINT | 1 byte |
| SMALLINT | 2 bytes |
| MEDIUMINT | 3 bytes |
| INT | 4 bytes |
| INTEGER | 4 bytes |
| BIGINT | 8 bytes |
| FLOAT(X) | 4 ú 8 bytes |
| FLOAT | 4 bytes |
| DOUBLE | 8 bytes |
| DOUBLE PRECISION | 8 bytes |
| REAL | 8 bytes |
| DECIMAL(M,D) | M+2 bytes sí D > 0, M+1 bytes sí D = 0 |
| NUMERIC(M,D) | M+2 bytes if D > 0, M+1 bytes if D = 0 |

## Tipos de fecha y hora:

A la hora de almacenar fechas, Mysql no comprueba de una manera estricta si una fecha es válida o no. Simplemente comprueba que el mes esté comprendido entre 0 y 12 y que el día esté comprendido entre 0 y 31.

**Date:** Tipo fecha, almacena una fecha. El rango de valores va desde el 1 de enero del 1001 al 31 de diciembre de 9999. El formato de almacenamiento es de año-mes-día

**DateTime:** Combinación de fecha y hora. El rango de valores va desde el 1 de enero del 1001 a las 0 horas, 0 minutos y 0 segundos al 31 de diciembre del 9999 a las 23 horas, 59 minutos y 59 segundos. El formato de almacenamiento es de año-mes-día horas:minutos:segundos

**TimeStamp:** Combinación de fecha y hora. El rango va desde el 1 de enero de 1970 al año 2037. El formato de almacenamiento depende del tamaño del campo:

| Tamaño | Formato |
|---|---|
| 14 | AñoMesDiaHoraMinutoSegundo aaaammddhhmmss |
| 12 | AñoMesDiaHoraMinutoSegundo aammddhhmmss |
| 8 | ñoMesDia aaaammdd |
| 6 | AñoMesDia aammdd |
| 4 | AñoMes aamm |
| 2 | Año aa |

**Time:** Almacena una hora. El rango de horas va desde -838 horas, 59 minutos y 59 segundos a 838, 59 minutos y 59 segundos. El formato de almacenamiento es de 'HH:MM:SS'

**Year:** Almacena un año. El rango de valores permitidos va desde el año 1901 al año 2155. El campo puede tener tamaño dos o tamaño 4 dependiendo de si se quiere almacenar el año con dos o cuatro dígitos.

| Tipo de Campo | Tamaño de Almacenamiento |
|---|---|
| DATE | 3 bytes |
| DATETIME | 8 bytes |
| TIMESTAMP | 4 bytes |
| TIME | 3 bytes |
| YEAR | 1 byte |

Tabla de relación tipo de campo-almacenamiento que permite

## Tipos de cadena:

**Char(n):** Almacena una cadena de longitud fija. La cadena podrá contener desde 0 a 255 caracteres.

**VarChar(n):** Almacena una cadena de longitud variable. La cadena podrá contener desde 0 a 255 caracteres.

Diferencia de almacenamiento entre los tipos Char y VarChar

| Valor | CHAR(4) | Almacenamiento | VARCHAR(4) | Almacenamiento |
|---|---|---|---|---|
| '' | '' | 4 bytes | '' | 1 byte |
| 'ab' | 'ab ' | 4 bytes | 'ab' | 3 bytes |
| 'abcd' | 'abcd' | 4 bytes | 'abcd' | |
| 'abcdefgh' | 'abcd' | 4 bytes | 'abcd' | 5 bytes |

Dentro de los tipos de cadena se pueden distinguir otros dos subtipos, los tipo Test y los tipo *BLOB* (Binary Large Object)

La diferencia entre un tipo y otro es el tratamiento que reciben a la hora de realizar ordenamientos y comparaciones. Mientras que el tipo Test se ordena sin tener en cuenta las mayúsculas y las minúsculas, el tipo *BLOB* se ordena teniéndolas en cuenta.

Los tipos *BLOB* se utilizan para almacenar datos binarios como pueden ser ficheros.

**TinyText** y **TinyBlob:** Columna con una longitud máxima de 255 caracteres.

**Blob** y **Text:** Un texto con un máximo de 65535 caracteres.

**MediumBlob** y **MediumText:** Un texto con un máximo de 16.777.215 caracteres.

**LongBlob** y **LongText:** Un texto con un máximo de caracteres 4.294.967.295. Hay que tener en cuenta que debido a los protocolos de comunicación los paquetes pueden tener un máximo de 16 Mb.

**Enum:** Campo que puede tener un único valor de una lista que se especifica. El tipo Enum acepta hasta 65535 valores distintos

**Set:** Un campo que puede contener ninguno, uno ó varios valores de una lista. La lista puede tener un máximo de 64 valores.

| Tipo de campo | Tamaño de Almacenamiento |
|---|---|
| CHAR(n) | n bytes |
| VARCHAR(n) | n +1 bytes |
| TINYBLOB, TINYTEXT | Longitud+1 bytes |
| BLOB, TEXT | Longitud +2 bytes |
| MEDIUMBLOB, MEDIUMTEXT | Longitud +3 bytes |
| LONGBLOB, LONGTEXT | Longitud +4 bytes |
| ENUM('value1','value2',...) | 1 ó dos bytes dependiendo del número de valores |
| SET('value1','value2',...) | 1, 2, 3, 4 ó 8 bytes, dependiendo del número de valores |

Tabla de relación tipo de campo-almacenamiento que permite

# ANEXO II

## Funciones SQL

## Funciones de control de Flujo:

| | |
|---|---|
| **IF** | Elección en función de una expresión booleana |
| **IFNULL** | Elección en función de si el valor de una expresión es *NULL* |
| **NULLIF** | Devuelve *NULL* en función del valor de una expresión |

## Funciones matemáticas:

| | |
|---|---|
| **ABS** | Devuelve el valor absoluto |
| **ACOS** | Devuelve el arco-coseno |
| **ASIN** | Devuelve el arco-seno |
| **ATAN** y **ATAN2** | Devuelven el arco-tangente |
| **CEILING** y **CEIL** | Redondeo hacia arriba |
| **COS** | Coseno de un ángulo |
| **COT** | Cotangente de un ángulo |
| **CRC32** | Cálculo de comprobación de redundancia cíclica |
| **DEGREES** | Conversión de grados a radianes |
| **EXP** | Cálculo de potencias de *e* |
| **FLOOR** | Redondeo hacia abajo |
| **LN** | Logaritmo natural |
| **LOG** | Logaritmo en base arbitraria |
| **LOG10** | Logaritmo en base 10 |
| **LOG2** | Logaritmo en base dos |
| **MOD o %** | Resto de una división entera |
| **PI** | Valor del número π |
| **POW o POWER** | Valor de potencias |
| **RADIANS** | Conversión de radianes a grados |
| **RAND** | Valores aleatorios |
| **ROUND** | Cálculo de redondeos |
| **SIGN** | Devuelve el signo |
| **SIN** | Cálculo del seno de un ángulo |
| **SQRT** | Cálculo de la raíz cuadrada |
| **TAN** | Cálculo de la tangente de un ángulo |
| **TRUNCATE** | Elimina decimales |

## Funciones para el tratamiento de cadenas de caracteres:

| | |
|---|---|
| **ASCII** | Valor de código ASCII de un carácter |
| **BIN** | Conversión a binario |
| **BIT_LENGTH** | Cálculo de longitud de cadena en bits |
| **CHAR** | Convierte de ASCII a carácter |
| **CHAR_LENGTH** o **CHARACTER_LENGTH** | Cálculo de longitud de cadena en caracteres |
| **COMPRESS** | Comprime una cadena de caracteres |
| **CONCAT** | Concatena dos cadenas de caracteres |
| **CONCAT_WS** | Concatena cadenas con separadores |
| **CONV** | Convierte números entre distintas bases |
| **ELT** | Elección entre varias cadenas |
| **EXPORT_SET** | Expresiones binarias como conjuntos |
| **FIELD** | Busca el índice en listas de cadenas |
| **FIND_IN_SET** | Búsqueda en listas de cadenas |
| **HEX** | Conversión de números a base hexadecimal |
| **INSERT** | Inserta una cadena en otra |
| **INSTR** | Busca una cadena en otra |
| **LEFT** | Extraer parte izquierda de una cadena |
| **LENGTH** u **OCTET_LENGTH** | Calcula la longitud de una cadena en bytes |
| **LOAD_FILE** | Lee un fichero en una cadena |
| **LOCATE** o **POSITION** | Encontrar la posición de una cadena dentro de otra |
| **LOWER** o **LCASE** | Convierte una cadena a minúsculas |
| **LPAD** | Añade caracteres a la izquierda de una cadena |
| **LTRIM** | Elimina espacios a la izquierda de una cadena |
| **MAKE_SET** | Crea un conjunto a partir de una expresión binaria |
| **OCT** | Convierte un número a octal |
| **ORD** | Obtiene el código ASCII, incluso con caracteres multibyte |
| **QUOTE** | Entrecomilla una cadena |
| **REPEAT** | Construye una cadena como una repetición de otra |
| **REPLACE** | Busca una secuencia en una cadena y la sustituye por otra |
| **REVERSE** | Invierte el orden de los caracteres de una cadena |
| **RIGHT** | Devuelve la parte derecha de una cadena |
| **RPAD** | Inserta caracteres al final de una cadena |
| **RTRIM** | Elimina caracteres blancos a la derecha de una cadena |
| **SOUNDEX** | Devuelve la cadena "soundex" para una cadena concreta |
| **SOUNDS LIKE** | Compara cadenas según su pronunciación |
| **SPACE** | Devuelve cadenas consistentes en espacios |
| **SUBSTRING** o **MID** | Extraer subcadenas de una cadena |

| | |
|---|---|
| **SUBSTRING_INDEX** | Extraer subcadenas en función de delimitadores |
| **TRIM** | Elimina sufijos y/o prefijos de una cadena. |
| **UCASE** o **UPPER** | Convierte una cadena a mayúsculas |
| **UNCOMPRESS** | Descomprime una cadena comprimida mediante COMPRESS |
| **UNCOMPRESSED_LENGTH** | Calcula la longitud original de una cadena comprimida |
| **UNHEX** | Convierte una cadena que representa un número hexadecimal a cadena de caracteres |

## Funciones de comparación de cadenas:

| | |
|---|---|
| **STRCMP** | Compara cadenas |

## Funciones para el manejo de fechas:

| | |
|---|---|
| **ADDDATE** | Suma un intervalo de tiempo a una fecha |
| **ADDTIME** | Suma tiempos |
| **CONVERT_TZ** | Convierte tiempos entre distintas zonas horarias |
| **CURDATE o CURRENTDATE** | Obtener la fecha actual |
| **CURTIME o CURRENT_TIME** | Obtener la hora actual |
| **DATE** | Extraer la parte correspondiente a la fecha |
| **DATEDIFF** | Calcula la diferencia en días entre dos fechas |
| **DATE_ADD** | Aritmética de fechas, suma un intervalo de tiempo |
| **DATE_SUB** | Aritmética de fechas, resta un intervalo de tiempo |
| **DATE_FORMAT** | Formatea el valor de una fecha |
| **DAY o DAYOFMONTH** | Obtiene el día del mes a partir de una fecha |
| **DAYNAME** | Devuelve el nombre del día de la semana |
| **DAYOFWEEK** | Devuelve el índice del día de la semana |
| **DAYOFYEAR** | Devuelve el día del año para una fecha |
| **EXTRACT** | Extrae parte de una fecha |
| **FROM_DAYS** | Obtener una fecha a partir de un número de días |
| **FROM_UNIXTIME** | Representación de fechas UNIX en formato de cadena |
| **GET_FORMAT** | Devuelve una cadena de formato |

| | |
|---|---|
| **HOUR** | Extrae la hora de un valor time |
| **LAST_DAY** | Devuelve la fecha para el último día del mes de una fecha |
| **MAKEDATE** | Calcula una fecha a partir de un año y un día del año |
| **MAKETIME** | Calcula un valor de tiempo a partir de una hora, minuto y segundo |
| **MICROSECOND** | Extrae los microsegundos de una expresión de fecha/hora o de hora |
| **MINUTE** | Extrae el valor de minutos de una expresión time |
| **MONTH** | Devuelve el mes de una fecha |
| **MONTHNAME** | Devuelve el nombre de un mes para una fecha |
| **NOW** o **CURRENT_TIMESTAMP** o **LOCALTIME** o **LOCALTIMESTAMP** o **SYSDATE** | Devuelve la fecha y hora actual |
| **PERIOD_ADD** | Añade meses a un periodo (año/mes) |
| **PERIOD_DIFF** | Calcula la diferencia de meses entre dos periodos (año/mes) |
| **QUARTER** | Devuelve el cuarto del año para una fecha |
| **SECOND** | Extrae el valor de segundos de una expresión time |
| **SEC_TO_TIME** | Convierte una cantidad de segundos a horas, minutos y segundos |
| **STR_TO_DATE** | Obtiene un valor DATETIME a partir de una cadena con una fecha y una cadena de formato |
| **SUBDATE** | Resta un intervalo de tiempo de una fecha |
| **SUBTIME** | Resta dos expresiones time |
| **TIME** | Extrae la parte de la hora de una expresión fecha/hora |
| **TIMEDIFF** | Devuelve en tiempo entre dos expresiones de tiempo |
| **TIMESTAMP** | Convierte una expresión de fecha en fecha/hora o suma un tiempo a una fecha |
| **TIMESTAMPADD** | Suma un intervalo de tiempo a una expresión de fecha/hora |
| **TIMESTAMPDIFF** | Devuelve la diferencia entre dos expresiones de fecha/hora |
| **TIME_FORMAT** | Formatea un tiempo |
| **TIME_TO_SEC** | Convierte un tiempo a segundos |
| **TO_DAYS** | Calcula el número de días desde el año cero |
| **UNIX_TIMESTAMP** | Devuelve un timestamp o una fecha en formato UNIX, segundos desde 1070 |
| **UTC_DATE** | Devuelve la fecha UTC actual |
| **UTC_TIME** | Devuelve la hora UTC actual |
| **UTC_TIMESTAMP** | Devuelve la fecha y hora UTC actual |
| **WEEK** | Calcula el número de semana para una fecha |

| | |
|---|---|
| **WEEKDAY** | Devuelve el número de día de la semana para una fecha |
| **WEEKOFYEAR** | Devuelve el número de la semana del año para una fecha |
| **YEAR** | Extrae el año de una fecha |
| **YEARWEEK** | Devuelve el año y semana de una fecha |

## Funciones de casting (conversión de tipos):

| | |
|---|---|
| **CAST o CONVERT** | Conversión de tipos explícita |

## Funciones de encriptado:

| | |
|---|---|
| **AES_ENCRYPT y AES_DECRYPT** | Encriptar y desencriptar datos usando el algoritmo oficial AES |
| **DECODE** | Desencripta una cadena usando una contraseña |
| **ENCODE** | Encripta una cadena usando una contraseña |
| **DES_DECRYPT** | Desencripta usando el algoritmo Triple-DES |
| **DES_ENCRYPT** | Encripta usando el algoritmo Triple-DES |
| **ENCRYPT(str)** | Encripta str usando la llamada del sistema Unix *crypt()* |
| **MD5(string)** | Calcula un checksum MD5 de 128 bits para la cadena string |
| **PASSWORD** u **OLD_PASSWORD** | Calcula una cadena contraseña a partir de la cadena en texto plano |
| **SHA o SHA1** | Calcula un checksum SHA1 de 160 bits para una cadena |

## Funciones de Información:

| | |
|---|---|
| **BENCHMARK** | Ejecuta una expresión varias veces |
| **CHARSET** | Devuelve el conjunto de caracteres de una cadena |
| **COERCIBILITY** | Devuelve el valor de restricción de colección de una cadena |
| **COLLATION** | Devuelve la colección para el conjunto de caracteres de una cadena |
| **CONNECTION_ID** | Devuelve el ID de una conexión |
| **CURRENT_USER** | Devuelve el nombre de usuario y el del host para la conexión actual |
| **DATABASE** | Devuelve el nombre de la base de datos actual |

| | |
|---|---|
| **FOUND_ROWS** | Calcular cuántas filas se hubiesen obtenido en una sentencia SELECT sin la cláusula LIMIT |
| **LAST_INSERT_ID** | Devuelve el último valor generado automáticamente para una columna AUTO_INCREMENT |
| **USER** o **SESSION_USER** o **SYSTEM_USER** | Devuelve el nombre de usuario y host actual de MySQL |
| **VERSION** | Devuelve la versión del servidor MySQL |

## Funciones de grupos:

| | |
|---|---|
| **AVG** | Devuelve el valor medio |
| **BIT_AND** | Devuelve la operación de bits AND para todos los bits de una expresión |
| **BIT_OR** | Devuelve la operación de bits OR para todos los bits de una expresión |
| **BIT_XOR** | Devuelve la operación de bits XOR para todos los bits de una expresión |
| **COUNT** | Devuelve el número de valores distintos de NULL en las filas recuperadas por una sentencia SELECT |
| **COUNT DISTINCT** | Devuelve el número de valores diferentes, distintos de NULL |
| **GROUP_CONCAT** | Devuelve una cadena con la concatenación de los valores de un grupo |
| **MIN** | Devuelve el valor mínimo de una expresión |
| **MAX** | Devuelve el valor máximo de una expresión |
| **STD o STDDEV** | Devuelve la desviación estándar de una expresión |
| **SUM** | Devuelve la suma de una expresión |
| **VARIANCE** | Devuelve la varianza estándar de una expresión |

## Funciones generales:

| | |
|---|---|
| **DEFAULT** | Devuelve el valor por defecto para una columna |
| **FORMAT** | Formatea el número según la plantilla '#,###,###.## |
| **GET_LOCK** | Intenta obtener un bloqueo con el nombre dado |
| **INET_ATON** | Obtiene el entero equivalente a la dirección de red dada en formato de cuarteto con puntos |
| **INET_NTOA** | Obtiene la dirección en formato de cuarteto con puntos dado un entero |
| **IS_FREE_LOCK** | Verifica si un nombre de bloqueo está libre |
| **IS_USED_LOCK** | Verifica si un nombre de bloqueo está en uso |
| **MASTER_POS_WAIT** | Espera hasta que el esclavo alcanza la posición especificada en el diario maestro |
| **RELEASE_LOCK** | Libera un bloqueo |
| **UUID** | Devuelve un identificador único universal |