UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN

"IMPLEMENTACIÓN DE SISTEMAS BASADOS EN LAMP"

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:
I N G E N I E R O E N C O M P U T A C I Ó N

P R E S E N T A :
M A R Í A D E L O S Á N G E L E S
G U T I É R R E Z H E R N Á N D E Z

ASESOR: M. EN C. MARCELO PÉREZ MEDEL

MÉXICO, 2007.

AGRADECIMIENTOS

- A la UNAM, porque no sólo me proporcionó las herramientas para desarrollarme en el ámbito profesional sino también en el personal, por ello me siento orgullosa de ser universitaria.

- A mi familia por su confianza, sus enseñanzas y su gran apoyo durante todo este tiempo, las cuales me permitieron llegar hasta este momento, especialmente a mi mamá la Lic. María Luisa Hernández Martínez.

- A mi hermana la Lic. Guadalupe Gutiérrez Hernández, por su gran apoyo durante todo este tiempo, especialmente mientras desarrollaba este proyecto de titulación; así mismo por sus enseñanzas y confianza en mí.

- A mi tío, el Ing. Luís Martín Hernández Martínez, por sus enseñanzas, apoyo y confianza, además de ser un gran ejemplo a seguir.

- A mi asesor, el Maestro Marcelo Pérez Medel, por sus enseñanzas, apoyo y confianza en mí.

- Al Lic. José Trinidad Campos Campos, por su gran apoyo y confianza.

- A la Lic. Gisella de León Cortes, por sus enseñanzas, confianza y apoyo.

- Al Ing. Narciso Acevedo Hernández, por sus enseñanzas, apoyo y confianza.

- A todas las personas que han hecho aportaciones a mi vida de distintas formas, como amigos, compañeros de clase o trabajo y profesores.

- A los autores de aquellos libros que tanto me han enseñado.

- A mis abuelos Ángela Martínez Maldonado, Delfino Hernández Corona y a mi tía Silvia Hernández Martínez por sus enseñanzas.

ÍNDICE

OBJETIVO GENERAL

Proponer una alternativa viable para desarrollar sistemas que permitan la automatización del control de procesos empleando software libre como: Linux, Apache, MySQL y PHP.

Explicar el manejo de las herramientas LAMP en el desarrollo e implementación de sistemas para el control de procesos e información que funcione en red o por Internet.

Justificar la importancia de la seguridad informática, los elementos que permiten proteger el sistema, el flujo de información, los vectores y mecanismos de seguridad, así como plantear la implementación de una política de seguridad aplicada al CONACYT, (basada en el RFC 2196).

JUSTIFICACIÓN

Actualmente es muy importante el uso de las páginas Web en Internet, así mismo en las empresas es necesario contar con sistemas que permitan la automatización del control de procesos.

Una alternativa viable para desarrollar estos sistemas es utilizar software libre como Linux, Apache-WWW-Server, MySQL y PHP; este conjunto de herramientas consigue reunir los requerimientos para una plataforma Web: un sistema operativo, un servidor Web, una base de datos y un lenguaje de programación.

Dichas herramientas se han hecho tan populares que han recibido el acrónimo de LAMP, las cuales nos proporcionan una visión más amplia acerca del desarrollo de sistemas en línea como en Internet o en cualquier otro tipo de red, y éstas fueron utilizadas en el diplomado “Desarrollo e Implementación de Sistemas con Software Libre.

LAMP está considerada como una de las mejores herramientas disponibles para que cualquier organización o individuo pueda emplear un servidor Web versátil y potente. Aunque creados por separado, cada una que lo forma dispone de una serie de características comunes. Estas cuatro herramientas pueden funcionar en una amplia gama de hardware, con requerimientos relativamente pequeños sin perder estabilidad.

INTRODUCCIÓN

Actualmente, en las empresas es necesario contar con sistemas que permitan la automatización del control de procesos, para lograr un uso más eficiente y un control más preciso de los recursos.

Para desarrollar estos sistemas se puede adquirir una licencia de software para satisfacer determinadas necesidades. Sin embargo, en ocasiones su precio es muy elevado.

Cuando contamos con el poder adquisitivo necesario para pagar el precio exigido, lo más factible es utilizar software comercial. Pero si sólo estamos en el proceso de aprendizaje, o deseamos modificar código para adaptarlo a nuestras necesidades; la solución más viable es que estos sistemas sean sustentados empleando software libre como: Linux, Apache-WWW-Server, MySQL y PHP; esta combinación es tan común en la actualidad que, incluso, ha recibido el acrónimo de LAMP, el cual traduce un conjunto de aplicaciones que permiten establecer una plataforma Web. Dichas herramientas han demostrado tener un alto desempeño, gran estabilidad y seguridad; además, fácilmente se pueden adquirir en la red sin costo alguno.

Asimismo, las herramientas LAMP nos proporcionan una visión más amplia acerca del desarrollo de sistemas que se encuentran en línea como en Internet o en cualquier otro tipo de red, ya que hoy en da estos sistemas y algunas otras aplicaciones son elementos críticos e indispensables en el funcionamiento exitoso de las compañías de la sociedad globalizada.

Este fue el motivo por el cual me interesé en tomar el diplomado, para poder ampliar mis conocimientos con respecto al desarrollo e implementación de sistemas, esto a causa de que tenía una idea general relacionada al acrónimo LAMP; sin embargo, necesitaba actualizarme, así como obtener más información referente a Linux, para continuar aprendiendo y poder emplearlo en futuros proyectos.

El presente trabajo se divide en cuatro capítulos, los cuales incluyen pequeñas partes de los proyectos realizados en cada módulo del diplomado:

Capítulo I:
Trata, a grandes rasgos, de exponer qué es Linux, sus características, su estructura, los comandos más utilizados, cómo proteger archivos, cómo crear ligas y cómo ver procesos en ejecución.

Se explica el funcionamiento de FTP, los aspectos que debemos considerar en la administración de Linux, el proceso de instalación basado en Slackware 10.0

(debido a que fue el utilizado en el diplomado de “Desarrollo e Implementación de Sistemas con software libre en Linux”), y, finalmente, la instalación y mantenimiento de dispositivos.

Capítulo II:
Presenta una visión general de los aspectos generales de las redes, los modos de transmisión y de operación, los niveles que componen el modelo OSI, las topologías de red, las etiquetas más comunes de Html, y la administración de un servidor Web.

Capítulo III:
Se muestra, en términos generales las ventajas y desventajas del lenguaje compilado e interpretado, la instalación y código de PHP, los elementos de las bases de datos, cómo crearlas y cómo interactuar con MySQL por medio de PHP.

Capítulo IV:
Ofrece, a grandes rasgos, información y algunas definiciones relativas a la seguridad en cómputo; así como los planes de contingencia, los mecanismos de seguridad y el análisis de riesgo.; todos los puntos de suma importancia debido a que las Tecnologías de la información se han convertido en el corazón de cualquier organización y ayudan a definir el rumbo que ésta debe seguir. Por ello, debemos saber más acerca de los mecanismos aplicados a las TI, los cuales forman la seguridad en cómputo. Además se propone la implementación de una política de seguridad de acuerdo con el RFC 2196.

CAPÍTULO 1

CARACTERÍSTICAS Y ADMINISTRACIÓN DE LINUX

OBJETIVOS ESPECÍFICOS

Explicar:

- Las características generales de Linux.

- Comandos generales, procesos y programas en Linux.
- Perfil y actividades del Administrador.
- El proceso de instalación y Administración de Linux.
- La instalación y mantenimiento de dispositivos.

JUSTIFICACIÓN

El sistema operativo Linux proporciona un ambiente multiusuario y multitareas. Una de las principales áreas de aplicación en las que destaca es en el desarrollo de sistemas en línea como Internet o en cualquier otro tipo de red.

Una de sus características es la facilidad en la configuración de nuestro propio servidor, destacando el costo de alojamiento y de operación de sitios Web dinámicos que puede ser prácticamente nulo, al menos en lo que a software se refiere, usando las herramientas LAMP (Linux, Apache-WWW-Server, MySQL y PHP).

En este capítulo se proporcionan aspectos y características generales de Linux, para dar una idea de los métodos de operación, así como la administración de este Sistema Operativo.

1.- CARACTERÍSTICAS Y ADMINISTRACIÓN DE LINUX

1.1 Sistema operativo Linux

La historia de Linux inicia en 1990, cuando un estudiante finlandés, de 23 años de edad, comenzó un proyecto personal basado en Minix[1]. Empieza creando drivers para dispositivos de hardware en Lenguaje C mejorando su plataforma. Para 1991 libera la primera versión de Linux sin ser una distribución oficial, la 0.01.

En el mismo año pone a disposición, a través de Internet, la primera versión oficial de Linux, la 0.02; después de esta publicación se distribuyó en forma gratuita el código de Linux e invitó a todo aquel que pudiera aportar nuevas ideas, además de mejorar el código vía Internet. Gracias a estas contribuciones Linux evolucionó rápidamente a las versiones 0.03, 0.10, 0.11 y 0.12. En marzo de 1992 fue creada la versión 0.95, y, hasta 1994, la versión 1.0 del Kernel de Linux fue liberada. En enero de 2001 se libera la versión 2.6. El proyecto aún no está terminado, lo actualizan continuamente cientos de personas alrededor del mundo.

Linux es un sistema operativo, tipo Unix, es libre y viene acompañado del código fuente. Está formado por el núcleo del sistema (kernel) más un gran número de programas/librerías. Se distribuye bajo la *GNU Public License*[2], por lo tanto, el código fuente tiene que estar siempre accesible.

El sistema ha sido diseñado y programado por una multitud de programadores alrededor del mundo. El núcleo del sistema sigue en continuo desarrollo bajo la coordinación de Linus Torvalds. Día a día, más y más programas están disponibles para este sistema, y la calidad de los mismos aumenta de versión a versión. La gran mayoría de los mismos vienen acompañados del código fuente y se distribuyen bajo los términos de la Licencia General Publica de GNU, lo cual significa que dicho sistema puede ser distribuido, copiado y modificado gratuitamente, a condición de no imponer ninguna restricción en sucesivas distribuciones.

Plataformas

El tipo de plataforma se refiere a la arquitectura del hardware sobre el que va a correr nuestro Linux, para ello, se debe conocer para adquirir el tipo de distribución apropiada de acuerdo al equipo donde se va a instalar.

---

[1] Minix.- Sistema operativo tutorial, escrito por Andrew Tanebaum.

[2] GNU Public License.- Licencia Pública General creada por la Fundación de software libre, orientada principalmente a proteger a la libre distribución, modificación y uso del software.

Características

Multitarea:

Presenta las tareas de forma intercalada para que se ejecuten varias simultáneamente. Por lo tanto, en Linux es posible ejecutar varios programas a la vez sin tener que obstaculizar la ejecución de cada aplicación. Figura 1.1.

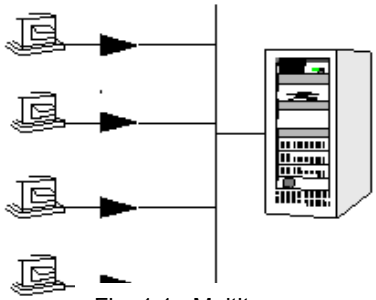


Fig. 1.1 Multitarea

Multiusuario:

Más de un usuario puede trabajar con la misma versión de un mismo programa al mismo tiempo, y actualizar inmediatamente cualquier cambio que se produzca en la base de datos, quedando reflejado para todos. Figura 1.2.

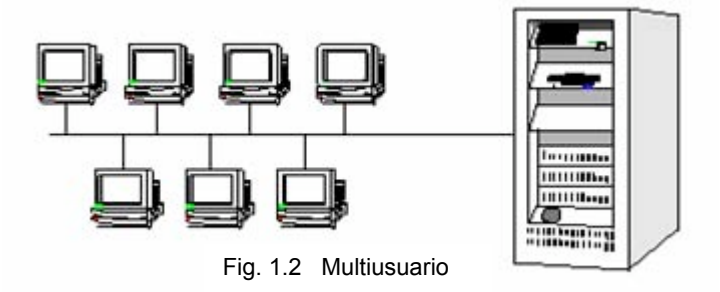


Fig. 1.2 Multiusuario

Multiplataforma:

Puede correr en muchas CPU distintas (Intel, AMD, Motorola, Sun, Sparc, etc.)

Multiproceso:

Varios procesos pueden usar la misma zona de memoria para ejecutarse.
Modular:
Podemos mezclar elementos y crear nuestra propia versión del Sistema Operativo, de esta forma disponemos de una multitud de programas y distribuciones. Se toma como base el kernel y luego se añaden sólo los programas que queremos.

Portable en sistemas abiertos:

Tiene la capacidad de transportar un Sistema Operativo de una plataforma a otra para que siga funcionando del mismo modo que lo hacia.

Shells programables:

Es la característica que hace a Linux más flexible. Casi todas las utilerías para emplear y administrar Linux se ejecutan mediante la escritura de comandos. En Linux al intérprete de la línea de comandos se le conoce como shell[3].

[3] Shell.- Programa diseñado para aceptar comandos y ejecutarlos.

Estructura

Linux está compuesto básicamente de cuatro capas:

- La capa más interna está conformada por el hardware, que es el conjunto de piezas físicas del equipo de cómputo.

- La segunda capa es el Kernel o núcleo del sistema, su función principal es interpretar las instrucciones proporcionadas por el usuario y convertirlas en lenguaje de máquina e indicarle al hardware lo que tiene que realizar con dicha información.

- La tercera capa está constituida por el grupo de los shells o intérpretes de comandos, los cuales funcionan como la interfaz entre el usuario y el Kernel (proporcionan las herramientas para que el usuario se pueda comunicar con el núcleo del sistema de Linux).

- La cuarta y última capa es donde se encuentra el usuario junto con los programas y aplicaciones que se le han agregado al sistema como hojas de cálculo, lenguajes de programación, manejadores de bases de datos, procesadores de texto, etc. En la figura 1.3 se muestra la estructura de Linux.

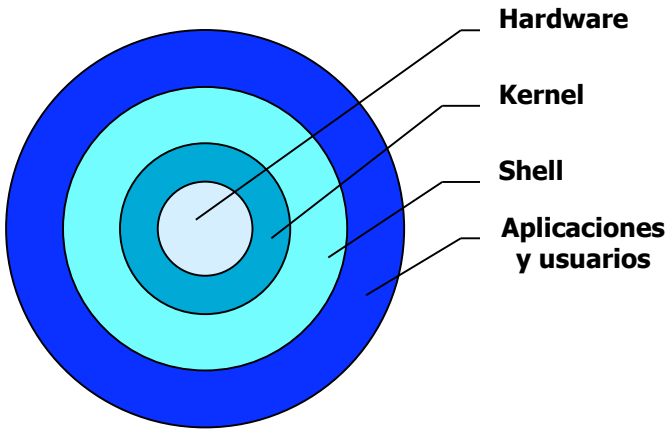


Fig. 1.3 Estructura de Linux

Interpretación de las versiones

Las versiones del Kernel se identifican con tres números, de la siguiente forma: XX.YY.ZZ

- *XX*: Señala la serie principal del Kernel. Hasta el momento sólo existen la 1 y 2. Este número cambia cuando la manera de funcionamiento del Kernel ha sufrido un cambio muy importante.
- *YY*: Indica si la versión es de desarrollo o de producción. Un número impar, significa que es de desarrollo; uno par, que es de producción.
- *ZZ*: Advierte nuevas revisiones dentro de una versión, en las que lo único que se ha modificado, son fallos de programación/bugs.

Filosofía

- Su forma modular lleva a su filosofía principal:
  "Entre más pequeño mejor".
- Permite la reducción de esfuerzo combinando las herramientas.
- Es un sistema personalizable de a cuerdo a las necesidades de cada usuario.

Distribuciones más comunes

Una distribución de Linux, es el núcleo, combinado con todas las utilerías, comandos, aplicaciones, entornos y otros programas que se incluyen en un sistema operativo.

Existen muchas distribuciones que pueden ser adquiridas por medio de Internet, haciendo la compra en el sitio oficial que le comercializa o descargándolo gratuitamente de los sitios de FTP. Todas las distribuciones utilizan la versión del Kernel que se actualiza constantemente; la diferencia radica en que los diseñadores y programadores le dan su toque especial, principalmente en la presentación de la interfaz gráfica, implementación de nuevas herramientas, programas, formatos gráficos, etc. (Tabla 1.4).

• **Debian GNU/Linux.**	• **Linux Red Hat Eurielec**
• **RedHat Linux**	• **ZipHam**
• **Slackware Linux**	• **MK – LinuxHispa**
• **Corel Linux**	• **Gentoo**
• **ESware Linux**	• **Knoppix**
• **Linux Mandrake**	• **GNU/LinEx**
• **SuSe Linux**	• **MkLinux**

Tabla 1.4 Distribuciones más comunes

¿Por qué utilizar Slackware?

Slackware es un avanzado sistema operativo Linux, diseñado con dos objetivos facilidad de uso y estabilidad como meta prioritaria. Incluye el más popular software reciente mientras guarda un sentido de tradición, proporcionando simplicidad y facilidad. Slackware trae lo mejor de todos los Linux a su disposición.

Slackware Linux proporciona tanto a los nuevos como a los experimentados usuarios un sistema con todas las ventajas, equipado para servidores, puestos de trabajos y máquinas de escritorio. Una larga lista de herramientas para programación, editores, así como las librerías actuales son incluidas para aquellos usuarios que quieren desarrollar o compilar software adicional.

1.1.1 Sistema de archivos

Linux esta compuesto de un sistema de archivos jerárquico en el cual no existen unidades de disco; en su lugar, cada unidad de almacenamiento así como cada dispositivo de hardware son reconocidos como un archivo o directorio dentro del sistema.

Existe sólo una raíz en el sistema la cual representa la parte más alta de la estructura de directorios y es conocida como root o raíz. Se representa por el símbolo de */* a partir del cual se desprenden diferentes ramas de directorios como se muestra en la tabla 1.5.

/	Raíz del sistema
/var	Variables del sistema
/var/spool/mail	Archivos de correo electrónico
/var/log	Bitácoras del sistema
/dev	Dispositivos de Hardware
/etc	Archivos de configuración del sistema
/home	Directorios de usuarios
/bin	Comandos del sistema, ficheros binarios o ejecutables
/sbin	Órdenes ejecutables sólo por el administrador
/boot	Archivos de inicio del sistema
/mnt	Dispositivos montables
/lib	Librerías
/tmp	Archivos temporales
/usr	Archivos de configuración y programas usados por el sistema

Tabla 1.5 Sistema de archivos

1.2 Estructura básica de los comandos.

La estructura de los comandos es muy rígida y generalmente sigue la misma sintaxis, es decir, siempre se escribe primero el comando; enseguida las opciones (si se requieren) y al final los argumentos necesarios.

Las opciones modifican el funcionamiento de un comando, generalmente se utilizan para agregar información a la salida de un comando o aumentar el potencial del comando a ejecutar. Las opciones no son necesarias para la ejecución del comando, siempre van antecedidas de un guión y se puede utilizar más de una posición.

Ejemplo:

comando -opciones argumento1 argumento2 ...

Reconocimiento del servidor y del sistema

hostname

-i	Esta opción nos muestra la dirección IP que tiene asignado el servidor de cada trabajo. Ejemplo: 132.248.75.120
-d	Nos muestra el dominio del servidor. Ejemplo: mascarones.unam.mx
-f	Presenta un listado completo incluyendo el nombre del servidor y su dominio. Ejemplo: atena.mascarones.unam.mx

uname .- Nos indica el tipo de Linux que se está utilizando.

- n	Indica el nombre del servidor. Ejemplo: atena
- r	Muestra la versión del kernel. Ejemplo: 2.4.18
-m	Indica la arquitectura del procesador. Ejemplo: i586
-a	Presenta todos los datos antes mencionados

passwd

Este comando permite realizar el cambio de contraseña actual.

pwd

Muestra el directorio actual de trabajo, es decir, donde nos encontramos posicionados dentro del sistema de directorios indicándonos la ruta absoluta.

touch

Crea archivos vacíos, de tamaño 0. Este comando es útil cuando nos interesa generar archivos que tienen una función especial como las bitácoras.

mkdir (Make directory)

Este comando nos permite crear varios directorios en la misma línea de comando, escribiendo uno a continuación de otro separados por un espacio. Ej.:

mkdir examen ejercicios ejemplos música imágenes

rmdir (Remove Directory)

Borra directorios con la condición de que se encuentren vacíos.

Ejemplo:rmdir nom_directorio

ls (list).- Muestra un listado del contenido del directorio actual de trabajo. Algunas opciones:

-l	Muestra un listado con formato largo indicando todas las propiedades de los archivos y directorios del directorio actual de trabajo.
-a	Despliega todo el contenido del directorio, incluyendo los archivos y directorios ocultos.
-t	Presenta un listado tomando como opción la fecha de modificación del archivo o directorio o subdirectorios a partir de la ruta que se le indique.

Comodines *, ?

Pueden sustituir uno o más caracteres durante la ejecución de comandos. Ejemplo:

ls ??a	Muestra aquellos archivos y directorios cuyo nombre contiene tres letras y la tercera corresponde a la letra a.
ls *u*t*	Muestra aquellos archivos y directorios los cuales tengan en el nombre una u y una t en cualquier posición pero en el orden que se le indica.

Change Directory (cd)

Este comando nos permite movernos entre directorios, para ello es necesario indicarle la ruta del directorio al que nos queremos cambiar.

1.2.1 Rutas Absolutas y Relativas

Las *rutas absolutas* siempre inician con el símbolo de */* que indica el punto más alto de la estructura de directorios. A partir de aquí se debe poner la ruta completa para llegar al punto que se quiera afectar.

cd /var/spool/mail

Las *rutas relativas* nunca inician con el símbolo de raíz / en lugar de ello comienzan con la ruta a partir del directorio actual de trabajo.

cd ../ ..

copy (cp)

El comando cp nos permite hacer copias de archivos uno por uno, múltiples archivos o incluso de estructuras de directorios.

cp ruta_origen ruta_destino

rm (remove)

El comando rm borra archivos. Puede ser eliminado un sólo archivo o varios haciendo uso de los meta caracteres *?

rm –r directorio

La opción –r hace un borrado en forma recursiva, esta opción es de mucho cuidado ya que se eliminará el directorio especificado con todo su contenido.

mv (move)

Mueve archivos o estructuras de directorios de un lugar a otro, y renombra archivos o directorios.

mv borrador capítulos

Consultar ayuda del sistema

El sistema cuenta con un manual en línea de los comandos, el cual puede ser desplegado en pantalla mediante el comando man, teniendo como argumento el comando que queremos consultar.

man ls .- Aquí se esta consultando el manual del comando ls.

1.2.2 Trabajo con archivos

cat .-Nos permite ver el contenido de uno o más archivos, su función original es +- concatenar archivos.

more

Este comando nos permite ver el contenido de uno o más archivos imprimiéndolos en pantalla. Para avanzar una pantalla se presiona la barra espaciadora y para una línea se presiona la tecla de enter.

more archivo1 archivo2...

less

Presenta el contenido de uno o dos archivos, permitiendo retroceder para ver información que ya paso; se usa de la misma manera que *more.*

less archivo

head

Nos muestra las primeras 10 líneas contenidas en un archivo.

-n esta opción le indica al comando que despliegue las primeras *n* número de líneas en lugar de 10. Ejemplo: head –20 nom_archivo

tail

Nos muestra las últimas 10 líneas contenidas en un archivo.

tail nom_archivo

pico

Programa que permite editar archivos de texto, es muy fácil de utilizar, ya que presenta un menú de comandos en la parte inferior de la pantalla y no requiere de muchos comandos para poder editar el contenido del archivo.

1.2.3 Redireccionamiento

En Linux existe una salida estándar que es el monitor, de igual manera hay una entrada estándar que es el teclado, de tal modo que todo lo que queramos hacer en forma convencional se lo debemos indicar al sistema operativo insertando las instrucciones desde el teclado, de la misma forma todo resultado de los comandos será impreso por default en la pantalla a no ser que se le indique otra cosa al sistema.

Redireccionamiento de salida

El caracter mayor que (>) nos permite redireccionar la salida de un programa hacia un archivo, ej.

ls > borrador

Manda la salida del comando *ls* a un archivo llamado borrador, el resultad o del comando *ls* no se imprime en pantalla, sino en el archivo indicado. Si el archivo borrador no existe será creado automáticamente, pero si ya contamos con él será sobrescrito y su contenido se perderá.

Redireccionamiento de entrada

El caracter menor que (<) nos permite redireccionar un archivo hacia un programa.
cat < borrador

Redireccionamiento de un programa a otro

El caracter pipe (**|**) funciona como un filtro enlazando la salida de un programa con la entrada de otro.

ls – l | more

Redireccionamiento no destructivo o de adición

Doble mayor que (**>>**) nos permite redireccionar la salida de un programa hacia un archivo, si el archivo no existe lo crea, pero si el archivo existe el resultado del comando lo añade al final del archivo existente sin eliminar el contenido del archivo,

ls >> borrador

Redireccionamiento condicionado

Existe una salida estándar de verdadero (1) y una de falso (0), de tal forma que si en algún comando no se cumple una condición no se imprime en pantalla sino se redirecciona hacia un archivo, ej.

ls –R /etc 1>miprueba 2>error

grep

Este comando busca patrones en archivos. Por defecto devuelve todas las líneas que contienen un patrón determinado en uno o varios archivos. Utilizando las opciones se puede variar mucho este comportamiento. Si no se le pasa ningún archivo como argumento hace la búsqueda en la entrada estándar. *Sintaxis:*

grep [opciones] <patrón> [ficheros]

1.3 Tipos de archivos

Dentro del sistema operativo vamos a encontrar archivos de diferentes tipos, los cuales los vamos a distinguir por el primer bit de sus propiedades. El atributo del primer bit nos indica si el elemento listado es directorio o archivo, así mismo nos indica de qué tipo de archivo se trata.

1.3.1 Archivos más comunes.

d	Nos indica que el elemento es un directorio.
-	Archivo ordinario.
l	El elemento listado es una liga suave.
c	Archivo de tipo caracter (Archivos de acceso de hardware mediante un bit como el mouse).
b	Archivo de bloque (acceso por bloques en paralelo como la impresora).

Tabla 1.6 Archivos más comunes

1.3.2 Permisos y chmod

Cuando un archivo es creado sobre Linux el sistema operativo no sólo acumula el nombre del archivo y la fecha de cuando se creó, sino también el ID (el dueño), así como el grupo al cual pertenece el archivo.

Para poder proteger un sistema de archivos contra accesos no deseados, los permisos son guardados separadamente para así restringir el acceso de acuerdo a las características dadas por el dueño a ciertos grupos de usuarios.

Esto es posible definiendo permisos generales, los cuales se ven reflejados en la salida mostrada por *ls*:

```
d r w x r _ x r _ _x 2   NATASHA   USERS   4096   2005-04-23   09:49   ejercicio1
```

Existen tres tipos de permisos que se aplican tanto a directorios como a archivos. Debemos asignar los permisos a los archivos de acuerdo a la función que van a realizar, dependiendo para qué fueron creados.

r lectura .- Permite que el archivo pueda ser leído y copiado

w escritura .-Permite escribir el archivo, hacer modificaciones o borrarlo

x ejecución .- Son aquellos que pueden realizar un proceso

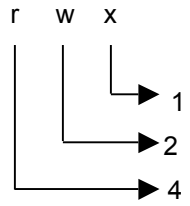
chmod

Permite cambiar permisos a los directorios o archivos, mediante dos métodos: El simbólico y el octal.

u	Indica que se modificarán los permisos del usuario dueño del elemento.
g	Indica que se cambiarán los permisos asignados al grupo.
o	Indica que se modificaran los permisos de los otros usuarios que no pertenecen al grupo.
a	Indica que afectará los permisos para todos los usuarios.

Método simbólico	***Ejemplos***
+ Agregar un permiso	chmod u+rx,g+r,o+w prueba
- Quitar un permiso	chmod u=rw,g=r,o=r prueba
= Asignar permisos	chmod a=rwx prueba

Método Octal **(Se asignan directamente los permisos)**



Ejemplo:

chmod 777 prueba

chown

Sirve para cambiar de dueño y grupo.

chown usuario:users /home/usuario/host

1.4 Ligas

Otra característica del sistema de archivos Linux es crear ligas. Si accedes a varios puntos del sistema de archivos, este archivo podría simplemente ser copiado; pero esto sería un desperdicio de memoria y en este caso, en Linux la creación de ligas provee una alternativa más practica. Las ligas pueden ser duras o simbólicas.

1.4.1 Ligas duras

Son una referencia adicional para un directorio a un archivo o sus inodos. El contador de las ligas tabula el número de tal referencia. Si un archivo al cual se refieren varias ligas puede ser borrado, el contador de la liga se decrementa. Sólo después de que el contador de la liga busca el valor el archivo puede ser borrado físicamente.

Así mantiene una relación con un archivo el cual está ligado; ambos tienen el mismo tamaño, al editar el contenido de uno se actualiza automáticamente el otro, pero a diferencia de un archivo común es que la liga está asociada al mismo inodo del archivo con el que está ligado.

Sintaxis: ln nom archivo nom_liga

1.4.2 Ligas suaves

Son un vínculo hacia un archivo, pero las ligas son de tamaño muy pequeño ya que sólo se hace referencia al archivo que se está ligando, funcionan como un acceso directo, su función es asociar un programa de nombre complejo con un nombre más fácil de recordar.

Sintaxis: ln –s nom_archivo nom_liga

1.5 Comandos básicos de red

Dentro del sistema operativo Linux existen diferentes comandos que permiten mantenernos en constante comunicación con los usuarios, sin importar el lugar donde se encuentre alojada su cuenta o el servidor. Tabla 1.7.

who	Muestra un listado sencillo de quien se encuentra conectado en el servidor, a qué hora se conectó y que está haciendo.
w	Muestra un listado con detalles de los usuarios conectados en el servidor.
whoami	Indica con qué login estoy registrado dentro del servidor. Este comando es útil ya que permite que un usuario se convierta en otro diferente, utilizando el comando su.
finger	Presenta información de los usuarios conectados en el servidor, por ejemplo, el nombre real del usuario.
finger login o finger nombre	Despliega información en específico del usuario indicado, como el nombre real del usuario, directorio $HOME, última fecha de acceso al servidor, etc. Ejemplo: finger usuario.

Tabla 1.7 Comandos básicos de red

1.5.1 Correo electrónico (mail)

Tecleando el comando sólo revisa la lista de correos. Todos los correos revisados con este programa son almacenados en un archivo llamado *mbox* que se encuentra en nuestro directorio $HOME

mail cuenta@correo.com

Mandará el correo electrónico a la cuenta que se le especifica. Una de las limitantes de este correo es que no permite enviar archivos adjuntos como tal.

1.5.2 Revisión de correo

h	Muestra una lista numerada de los títulos de los correos recibidos.
#	Presenta el contenido del correo deseado, tecleando el número que aparece en el listado de correos.
d#	Borra el correo marcado en la lista con el número que le estamos indicando.

talk

Este programa permite realizar una conversación interactiva en tiempo real, además es posible mantener una conversación con más de dos usuarios al mismo tiempo, para ello es necesario que los usuarios se encuentren conectados en el servidor.

talk login1 login2 login3 …

ytalk

Este comando tiene funciones muy parecidas a talk, sólo que este comando permite realizar una solicitud de conversación a más de un usuario, como único requisito es que se tengan un mínimo de tres líneas para cada usuario.

ytalk usuario@yahoo.com

diff

Este comando activa o desactiva la recepción de mensajes proporcionados por el sistema, como los de correo electrónico.

Programas útiles dentro de Linux

date

Muestra la hora del sistema.

df

Muestra el estado de las particiones de los discos duros, indicándonos el espacio utilizado y el estado libre en disco.

last -n

Presenta las últimas conexiones de los usuarios al servidor.

sort

Ordena alfabéticamente el contenido de un archivo (por líneas).

wc –l

Cuenta únicamente las líneas.

cal

Entrega el calendario del mes en curso.

Ejemplos:

cal 6 1975 nos mostrará el calendario del sexto mes del año 1975

cal –y mostrará el calendario del año en curso.

1.6 Procesos

Un proceso es un programa que se encuentra corriendo dentro del servidor. Éste tiene un tiempo de vida que va desde el momento en que presionamos la tecla *enter* hasta que termina la ejecución del proceso.

Todos los procesos tienen un dueño el cual es el usuario que lo invocó desde el intérprete de comandos, de tal forma que ningún otro usuario puede afectar las tareas que el primero está realizando, debido a que no es dueño del proceso.

Cuando iniciamos una sesión dentro del servidor, en realidad estamos ejecutando una copia del intérprete de comandos que tenemos por default. A esta copia del intérprete se le asigna un número, asociado al proceso en ejecución. Por cada sesión que iniciemos se va a ejecutar una nueva copia del *shell,* asignándole un nuevo número de identificación para este proceso. Es importante recalcar que todo proceso tiene una dependencia del shell principal del sistema.

1.6.1 ps

Permite ver los procesos que se están ejecutando desde nuestra sesión abierta, y que se encuentran en uso en este momento, también muestra un listado de los procesos en formato corto.

ps –u

Da un listado de todos los procesos que está ejecutando un usuario, aunque se encuentre trabajando en varias sesiones el sistema listará todos los procesos que le pertenecen.

ps –fea

Muestra todos los procesos ejecutándose en el servidor en formato largo, donde podemos ver la dependencia de los procesos el UID, PID y el PPID.

El **UID** es el identificador del Usuario dueño del proceso El **PID** es el número del identificador del Proceso en ejecución. El **PPID** es el identificador del Proceso Padre del Proceso en ejecución.

Los procesos pueden encontrarse en diferente estado, dependiendo de la atención que estén teniendo en un momento en específico.

ps –h

Nos indica el estado en que se están ejecutando los procesos.

Estados de los procesos:

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

1.6.2 Procesos en primer y segundo plano.

Una vez que se esta ejecutando un programa, podemos enviarlo a segundo plano presionando la combinación de teclas ctrl+z.

jobs

Este comando permite ver los programas que se están ejecutando en segundo plano, y se asigna un número que sirve para identificar el proceso.

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

kill

Este comando permite, entre otras opciones, matar procesos. Los usuarios únicamente pueden matar sus propios procesos. Sólo cuando sea necesario.

kill –9 PID mata el proceso indicado

1.6.3 Respaldos

tar

Empaca una estructura de directorios; su función principal es almacenar el contenido de un directorio en forma recursiva dentro de un archivo.

Sintaxis:

tar [opciones] archivo.tar archivo

gzip

Realiza una compresión de archivos, manejando 9 niveles de compresión que van desde el –1 al –9 donde el nivel -1 es el de compresión más bajo mientras el nivel -9 es el de compresión más alto. Al momento de ejecutar esta utilera el programa le añade la extensión *.gz* al archivo comprimido, por default maneja el nivel de compresión 6.

Sintaxis: gzip [-n] archivo

1.6.4 FTP (Protocolo de Transferencia de Archivos)

FTP es un programa especialmente diseñado para la transferencia de archivos a través de Internet. No es confiable ya que deja pasar la información, en texto claro, en caso de que algún paquete no sea recibido en forma íntegra el sistema recibe un mensaje de error y automáticamente reenvía el paquete sin que haya pérdida de información.

Para trabajar con el servicio de *ftp* se debe realizar un proceso de registro al servidor que se quiere acceder, el cual se compone del inicio de sesión *ftp*, la apertura de la llamada al servidor, el registro del nombre del usuario y la contraseña de usuario para acceder al sistema mediante *ftp.* A continuación debemos posicionarnos dentro del directorio en el cual queremos extraer o depositar la información.

Ejemplo:

ftp: Inicia una sesión de ftp.

open: 132.248.75.120 Abre la conexión de ftp en el servidor que se le indica.

name (132.248.75.120:angeles): Solicita el nombre de usuario o login.

Password: Pide el password de acceso del usuario.

Sólo si todos los datos insertados son correctos tendremos acceso al servidor para realizar la transferencia de archivos.

get .- Descarga un archivo del servidor

put.- Sube un archivo al servidor.

close o quit.- Cierra la conexión de ftp en el servidor.

1.7. Perfil y actividades del administrador de Linux

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

1.7.1 Importancia de la administración

- Proporcionar un ambiente seguro, eficiente y confiable.
- Dividir el trabajo entre varios administradores, dependiendo el tamaño del sistema.
- Planear las actividades.
- Guardar copias de seguridad (bases de datos, correo electrónico, archivos originales del sistema, etc.)
- Conocer las utilerías del sistema. *Básicas:* cut, sort, paste, dic, comm, tail, head, Grez, compress, etc. *Control de tareas:* at, crontab. *Respaldos:* dump, dd, restore, tar, cpio. *Herramientas de programación*: c, shell, awk, perl. *Documentación:* man, aproposs, *información Impresa:* libros, manuales*. Internet*.

1.7.2 Conocimientos requeridos

- Técnicas de programación (dominio de al menos un lenguaje).
- Técnicas de administración del Sistema Operativo.
- Conocimientos básicos de hardware y mantenimiento de dispositivos.
- Comprensión profunda sobre redirección, tuberías y procesamiento en segundo plano, etc.
- Manejo de *vi* (ya que es el común en sistemas UNIX).
- Programación *shell.*
- Conocer el hardware de la máquina.
- Mantener canales de comunicación con los usuarios.

1.7.3 Establecimiento de políticas de uso y administración

- Apertura de cuentas.
- Horas de mantenimiento.
- Responsabilidad de los respaldos.
- Borrado de archivos temporales.
- Cuotas de disco.
- Seguridad del sistema.

1.7.4 Tareas administrativas comunes

- Administración de usuarios.
- Configuración y mantenimiento de dispositivos.
- Programación periódica de respaldos.
- Capacitación y asesoría de usuarios.
- Seguridad del sistema.
- Registro de cambios del sistema.
- Mantenimiento de claves de usuarios.
- Instalación y actualización de software (comercial y de dominio público).
- Configuración de las interfaces de red.
- Administración de los recursos (CPU, memoria y disco).
- Monitoreo del sistema y detección de fallas.
- Auditoría e implantación de la seguridad del sistema.

1.8 Instalación

Slackware Linux no requiere de un sistema extremadamente potente para ejecutarse ya que se puede ejecutar en equipos 386 o superiores.

1.8.1 Tipos de instalación

- CD ROM
- Discos de arranque.
- Disco de red.
- Disco PCMCIA (cuando se desea instalar Linux en una computadora portátil).

1.8.2 Proceso de Instalación de Slackware 10.0

Una vez elegido el método de instalación procedemos a instalar Linux. Aparecerá una pantalla de bienvenida al programa de instalación Linux, en la parte inferior de ésta aparecerá el indicador boot:

- El primer paso de la instalación es elegir la configuración de nuestro teclado: qwerty/es.map
- A continuación debemos probar la localización de los caracteres en el teclado para verificar que su distribución corresponda al tipo de mapa que elegimos. Con el no. 2 se cambia el mapa y con el 1 se acepta.
- Después debemos registrarnos como *root* y crear las particiones necesarias para la instalación.
- Para particionar el disco usamos el comando fdisk/dev/hda, donde las tres primeras particiones deben ser primarias y la última extendida.
- Un ejemplo podría ser:

swap	512M
/	5G
/home	10G
/var/spool/mail	5G
/usr/local	3G

En la segunda partición tenemos que definir el principio y el final de los cilindros. Por default nos ofrece el primer cilindro disponible de disco, para indicar el final de

la partición se usa un método de suma de mega bites a partir del punto de inicio; representado como:

Last cylinder or +size or +sizeM or sizeK (511-1048;default 1048):+218M

- Tenemos que cambiar la partición a tipo swap, para esto, presionamos t (de tipo) e indicamos al sistema que queremos modificar la partición número 2. A continuación nos pide el código en formato hexadecimal. Si desconocemos el código podemos consultar tecleando l (de listar) y elegiremos el código que corresponde. (82 es de Linux swap).
- Ahora, generaremos una segunda partición de Linux, en la cual se instalará el sistema.
- La siguiente partición es de tipo extendida y después lógicas hasta la 6.
- Guardamos los cambios y comenzamos con el comando setup. (w.- Guardar cambios).
- El sistema detecta automáticamente la partición swap y nos solicita darle formato para preparar la instalación.
- El proceso de instalación reconoce las particiones de Linux, la que elijamos en este momento es donde se va a instalar el sistema.
- Ya que elegimos la partición, solicita darle formato rápido, después elegimos el tipo de sistema de archivos con el cual va a ser formateado, elegimos ext3 para todas.
- Después entrega un reporte de inodos. Donde seleccionamos el que esta por default. 4096 (default).
- El siguiente paso es elegir el recurso desde donde se va a realizar la instalación. CD or DVD.
- Elegir el modo de instalación (full es el modo automático).
- Debemos elegir el kernel que está por default: /cdrom/kernels/bare.i/bzImage
- El asistente nos pide configurar el MODEM.
- En el HOTPLUS le decimos que sí lo habilite.
- Ahora debemos instalar LILO, en modo experto. EXPERT USE LILO.CONF SETUP MENU.
- Nos lleva a otro menú donde comenzamos con: BEGIN START LILO CONFIGURATION WITH A NEW LILO HEADER.
- Ahora nos pide la resolución. Hoy en día casi todas las tarjetas soportan una resolución de 1024x768x256.

- Después instalamos LILO en el MBR: MBR USE THE MASTER BOOT RECORD. (Cuando sólo hay dos sistemas).

- Instalamos LILO en /dev/hda.
- Elegimos un tiempo de espera antes de que inicie con el primer sistema. Y seleccionamos FOREVER.
- Comenzamos agregando un sistema Windows: WINDOWS ADD A WINDOWS FAT OR NTFS PARTITION TO THE LILO.CONF Y solicita la partición apropiada /dev/hda1. Después para la etiqueta WindowsXP.
- Agregamos Linux, se elige la partición apropiada /dev/hda3 y la etiqueta Slackware10.
- Seleccionamos INSTALL LILO.
- Elegimos tipo de mouse ps/2 y habilitamos el modo consola en el siguiente paso en GPM.
- Configuramos la red.
- Nos pregunta si deseamos reemplazar /etc/fstab, y le decimos que sí.
- Creamos algún password muy difícil de adivinar.
- Retiramos el CD y tecleamos Ctrl.-Alt.-Supr. Para reiniciarlo.

Si se tiene problemas con la interfaz gráfica debemos buscar las características de la tarjeta de video que tenemos y de acuerdo a ésta hacer unas modificaciones en un archivo que se encuentra en: /etc/X11 llamado xorg.conf en la sección de monitor. (Se puede auxiliar de Internet). Además en este mismo archivo podemos cambiar características del mouse, teclado, etc.

Para iniciar con interfaz grafica debemos editar un archivo: /etc/rc.d/rc.local, agregar una línea /opt/kde/bin/kdm.

1.9 Dar de alta o de baja el sistema

De esto depende en gran medida el rendimiento y confiabilidad del sistema, incluso después de un reinicio, también puede ser útil para rescatar un sistema cuando no inicia apropiadamente. Pero para esto es necesario comprender qué sucede cuando encendemos un equipo y cuál es el proceso de arranque de un sistema.

Lo primero que realiza el proceso de arranque de Linux es cargar una copia del kernel que se encuentra en el directorio */boot*, a continuación detecta todos los dispositivos de hardware y los pone a disposición del sistema, busca en los scripts de configuración para leer el nivel de inicio para el cual esta configurado por default, luego busca los scripts de run level, y finalmente, busca el script rc.local.

El nivel de inicio por default se encuentra definido en el archivo /etc/initab, para ello debemos conocer los diferentes niveles que están predefinidos en el archivo de inicio ilustrados en la tabla 1.8.

# 0	Halt
# 1	Modo monousuario
# 2	Sin uso, pero configurado como el nivel 3.
# 3	Modo multiusuario, runlevel de default en slackware
# 4	X11 con KDM/GDM/XDM
# 5	Sin uso, pero configurado como el nivel 3.
# 6	Reboot

Tabla 1.8. Niveles que están predefinidos en el archivo de inicio

Dependiendo del nivel de inicio, la rutina de encendido-apagado puede variar.

En la siguiente línea se muestra la configuración del nivel de inicio por default en Linux slackware. Como puede verse se encuentra definido en el nivel 3 de inicio.

```
# Default runlevel (Do not set to 0 or 6)
id:3:initdefault:
```

1.9.1 Salida del sistema

Es necesario indicarle al sistema que queremos terminar la sesión de trabajo en el servidor, para ello existen varios comandos, entre ellos se encuentran *exit* y *logout* o la combinación de teclas ctrl.+d.

poweroff

Apagado inmediato del sistema, muy peligroso para ciertas tareas ya que no verifica que los procesos se hayan terminado en forma apropiada.

halt

Muy parecido a poweroff, sólo que este hace un llamado al comando halt quién es quien apaga el sistema.

Reboot

Comando para reiniciar el sistema pero no verifica que los programas sean terminados eficientemente.

Shutdown

Comando considerado el más eficiente para detener o reiniciar el sistema, debido a que "da de baja el sistema de una forma segura". Se tiene control del apagado del sistema ya que se puede definir un tiempo y todos los usuarios se les avisa que se va a dar de baja mediante la instrucción SIGTERM; deshabilita el login, se comunica con el programa initt y le solicita el nivel 0 de ejecución.

Este comando envía una sincronización de discos, manda la señal de término a los programas y los cierra eficientemente, después verifica que no hay tareas en ejecución, da de baja los demonios y finalmente apaga de forma segura.

Shutdown –h now	Apaga el sistema de forma inmediata.
Shutdown –r now	Reinicia el sistema de forma inmediata.
Shutdown –k	No apaga el sistema, sólo manda un aviso a los usuarios conectados.
Shutdown –c	Cancela la instrucción shutdown.
Shutdown +n acción	Espera n minutos antes de realizar la acción.

runlevel .- Sirve para saber qué nivel se está ejecutando.

```
angeles:~# /sbin/runlevel
N 3                    //Aquí nos dice que el sistema se inició en modo 3.
```

Nota: La mayoría de las tareas administrativas tienen que realizarse en modo monousuario o de mantenimiento para evitar que se inicien los dispositivos de red y los servicios del sistema.

1.9.2 Claves de usuarios

Una de las principales tareas es la administración de las cuentas de usuario, para ello se debe tener un plan administrativo de claves, contemplando altas, bajas y cambios; también es importante definir los grupos a los que va a pertenecer uno o

más usuarios, así como tener cuidado en la asignación de permisos y en la administración adecuada de recursos para evitar abuso de ellos.

adduser.- Agrega usuarios.
groupadd.- Agrega grupos.
userdel.- Elimina usuarios.
groupdel -r.- Elimina grupos en forma recursiva.
passw Tabla. 1.9 archivos más comunes

1.9.3 Instalación y mantenimiento de dispositivos

El sistema dispone de un directorio /*mnt,* el cual forma parte de la estructura estándar de Linux, dentro de éste, existen algunos otros que se sugieren para montar las unidades de disco. Cuando instalamos Linux, este reconoce todas las unidades de disco y crea un directorio para cada una de ellas en /mnt. Si agregamos más unidades de disco al CPU podemos crear tantos directorios como necesitemos.

El sistema cuenta con apuntadores que ayudan con ciertas tareas, tal es el caso de la unidad de CD-ROM, en */dev* existe un apuntador que se dirige a directorio mount /mnt/cdrom, el cual está declarado dentro de un archivo de configuración, en este caso se puede montar el CD-ROM en forma rápida:

mount /mnt/cdrom	ó bien,	/dev/cdrom /mnt/cdrom

Cabe aclarar que son pocos los dispositivos que cuentan con estos apuntadores y es posible generarlos para agilizar tareas.

Como es el caso de Windows, para poder montarlo debemos posicionarnos en el directorio /*mnt* y ahí crear un directorio que se llame Windows.

Después editar el archivo */etc/fstab* y agregar la siguiente línea:

```
/dev/hda1       /mnt/windows       vfat       defaults       0   0
```

Y así podremos tener acceso a Windows de la misma manera como lo hicimos con el CD-ROM; una manera más sencilla es darle clic izquierdo al dispositivo al qué se desee acceder en la interfaz gráfica.

Para desmontar un dispositivo debemos seguir ciertos lineamientos, primero el dispositivo no tiene que estar en uso y/o nadie se debe encontrar colocado en la estructura de dicha unidad y así, escribir la instrucción: umount /mnt/cdrom ó umount /dev/cdrom.

CAPÍTULO 2

ADMINISTRACIÓN WEB MEDIANTE EL SERVIDOR APACHE

OBJETIVOS ESPECÍFICOS

Explicar:

- Aspectos generales de las redes.
- Cómo se generan las páginas Web con Html.
- En qué consiste el hipertexto :
  - Referencias locales.
  - Documentos Html externos.
- La inclusión de imágenes.
- El manejo de formularios.
- Cómo se crean las tablas.
- Qué son los frames.
- Aspectos generales de un servidor y el funcionamiento de http.
- La configuración básica del servidor Web Apache.

JUSTIFICACIÓN

En la actualidad es muy importante el uso de las páginas Web en Internet, por lo tanto, se tiene una gran necesidad de contar con servidores de WWW que atiendan la creciente demanda de nuestros sitios.

Un servidor Web es un programa encargado de ofrecer comunicación mediante el protocolo HTTP y servicios dentro del World Wide Web en Internet.

Apache es un servidor HTTP de código abierto para plataformas Unix como GNU/Linux, es usado por un porcentaje importante de sitios Web y continúa ganando terreno constantemente. Está diseñado para ser un servidor Web potente y flexible que pueda funcionar en la más amplia variedad de plataformas y entornos.

Las diversas plataformas y los entornos, hacen que a menudo sean necesarias ciertas características o funcionalidades, o que alguna de ellas sea implementada de diferente manera para obtener una mayor eficiencia; es por ello, que Apache se ha adaptado siempre a una gran variedad de entornos a través de su diseño modular. Este diseño permite a los administradores de sitios Web elegir qué características van a ser incluidas en el servidor, seleccionando los módulos que se van a cargar, ya sea al compilar o al ejecutar el servidor.

2. ADMINISTRACIÓN WEB MEDIANTE EL SERVIDOR APACHE

2.1 Aspectos generales de las redes

Se conoce como *Internet* a un conjunto de redes independientes (de área local y extensa) que se encuentran conectadas entre sí, permitiendo el intercambio de datos y constituyendo una red mundial que resulta medio idóneo para el intercambio de información, distribución de datos de todo tipo e interacción personal.

Una *red* es un sistema de comunicaciones de datos que permite conectar computadoras y periféricos para así compartir información y recursos, Figura 2.1. Las redes suelen clasificarse según su extensión en:

LAN (Local Área Network). Son las de área local. Su extensión suele estar restringida a una sala o edificio, aunque también podrían utilizarse para conectar 2 ó más edificios próximos. Algo habitual entre instituciones.

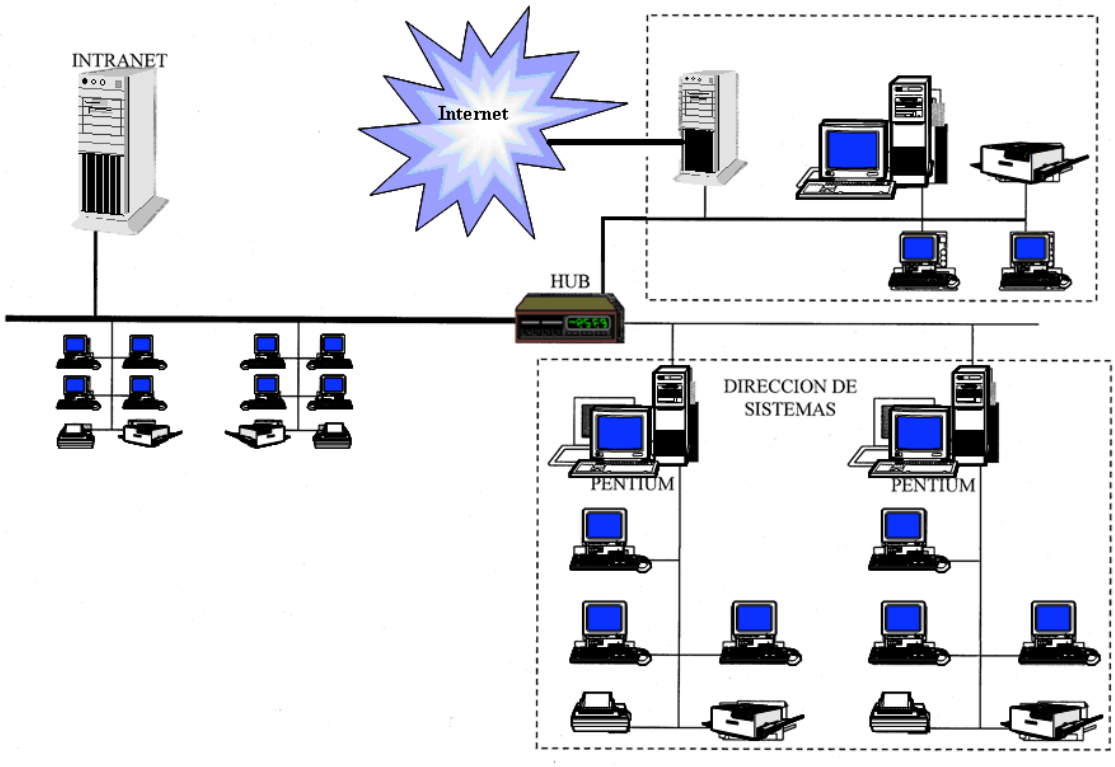


Fig. 2.1 Conexión de una red

WAN (Wide Area Network). Redes formadas por ordenadores que físicamente están distantes entre sí, como ocurre en las redes de empresas multinacionales.

MAN como WAN pueden considerarse como redes únicas al pertenecer a una misma entidad y utilizar sus infraestructuras, compartiendo arquitectura y sistema[1].

Varias redes pueden conectarse entre sí formando una red lógica de área mayor. Para que la transmisión entre todas ellas sea posible se emplean los ***ruteadores,*** que son los sistemas que se encargan de dirigir la información por el camino adecuado entre varias redes.

Cuando las redes que se conectan son de diferente tipo y con protocolos distintos, se hace necesario el uso de los ***gateways*****,** los cuales además de encaminar la información son capaces de convertir los datos de un protocolo a otro. Generalmente los términos ruteador y gateway se emplean indistintamente para referirse de forma general a los sistemas encargados de encaminar datos.

Para ello, es necesario el uso de un protocolo de comunicaciones en común. El protocolo que proporciona la compatibilidad necesaria para la comunicación en Internet es el TCP/IP (Transport Control Protocol/Internet Protocol).

Los protocolos de comunicación definen las normas que posibilitan que se establezca una comunicación entre varios equipos y dispositivos. Ya que estos equipos pueden ser diferentes entre sí, la interconexión se lleva a cabo a través de una interfaz, la cual es la encargada de la conexión física entre los equipos, definiendo las normas para las características eléctricas y mecánicas de la conexión.

2.1.1 Modelo básico de un sistema de comunicación

Elementos:

- *Fuente*.- Es el lugar donde se origina la información.
- *Transmisor.*- Prepara la señal de transferencia para acoplarla al medio de transmisión para que pueda viajar y llegar al receptor.
- *Medio*.- Enlace físico por el cual fluye la información. Líneas de comunicación, circuito por donde viaja la información o espacio libre.
- *Receptor*.- Se encarga de aceptar la información que le interesa al usuario final.
- *Destino*.- El usuario final.

2.1.2 Modos de operación

- *Simplex*. En este modo de operación los datos viajan a través de la línea de comunicación en una sola dirección, Ejemplos: Televisión y audio.

[1] Cisco System, Inc., *Academia de Networking de Cisco Systems: Guía de primer año*, 2ª Ed. Pearson Education, p.52

- *Half-duplex.* La transmisión se lleva cabo en dos direcciones, pero sólo en una a la vez (también conocido como bidireccional alternado). Ejemplos: sistemas de preguntas/respuestas.
- *Full - Duplex.* Se transmite simultáneamente en ambas direcciones. Ambos lados están habilitados para enviar y recibir al mismo tiempo. Ejemplo: comunicación de datos entre equipos mainframes.

2.1.3 Modos de transmisión

- *Serial.* Solamente se transfiere un bit al mismo tiempo, por lo que sólo se utiliza un hilo para enviar información.
- *Paralelo.* Se envían todos los bits utilizados para representar un caracter al mismo tiempo. Emplea tantos hilos como número de bits utilizados, uno por cada bit que se está transmitiendo. La transmisión es más cara pero más rápida.

2.1.4 Modelo OSI (Open System Interconnection)

"El modelo de referencia OSI es el modelo principal para las comunicaciones de red"[2]. Este modelo fue creado por el ISO (Organización Internacional de Normalización), y se compone de *siete niveles* o capas donde cada una de ellas define las funciones que deben proporcionar los protocolos con el fin de intercambiar información entre varios sistemas. Cada nivel depende de los que están debajo de él, y a su vez proporciona funcionalidad a aquellos superiores. Tabla 2.2.

Capa 7 Aplicación	El nivel de aplicación es el destino final de los datos donde se proporcionan los servicios al usuario.
Capa 6 Presentación	Se convierten e interpretan los datos que se utilizarán en el nivel de Aplicación.
Capa 5 Sesión	Encargado de ciertos aspectos de la comunicación, como el control de los tiempos de transmisión.
Capa 4 Transporte	Transporta la información de una manera fiable para que llegue correctamente a su destino.
Capa 3 Red	Nivel encargado de enrutar los datos hacia su destino eligiendo la ruta más efectiva.
Capa 2 Enlace de datos	Controla el flujo de los mismos, la sincronización y los errores que puedan producirse.
Capa1 Física	Se encarga de los aspectos físicos de la conexión, tales como el medio de transmisión o el hardware.

Tabla 2.2 Niveles o capas

[2] Ibidem, p.52.

2.1.5 Topología de redes

Es la forma física o la estructura de interconexión entre los distintos equipos (dispositivos de comunicación y computadoras) de la red. Hay dos categorías de diseño de topología que dependen de sí, la red es de área local (LAN) o una conexión de Inter-redes con encaminadores y conexiones de red de área extensa (WAN).

2.1.6 Criterios para establecer una topología de red

- **Fiabilidad:** Brindar la máxima fiabilidad y seguridad posible, para garantizar la recepción correcta de toda la información que soporta la red.
- **Costos**: Proporcionar el tráfico de datos más económico entre el transmisor y receptor en una red.
- **Respuesta**: Ofrecer el tiempo de respuesta óptimo y caudal eficaz o ancho de banda, que sea máximo.

2.1.7 Topologías de red más comunes

Jerárquica (Tipo árbol)
Es una de las más extendidas en la actualidad. El software es sencillo. Las tareas de control están conectadas en la jerarquía o nivel más elevado de la red y hoy en día incorpora en su operación el trabajo descentralizado en los niveles inferiores, para reducir la carga de trabajo de la jerarquía superior.
A pesar de ser fácil de controlar, tiene como desventajas: la posibilidad de cuellos de botella, la centralización y saturación de datos, la opción a que falle la parte principal, con lo cual toda la red dejaría de funcionar.

Horizontal (Tipo bus)
Muy frecuente en las redes de área local LAN. Permite que todas las computadoras conectadas en red, llamadas estaciones de trabajo o terminales, reciban todas las transmisiones. La desventaja está en que suele existir un sólo canal de comunicación para todos los dispositivos de la red. En consecuencia si falla un tramo de la red, toda la red deja de funcionar. Esta topología se recomienda cuando la red de datos a implementar es menor o igual a cuatro estaciones de trabajo. Tiene poca seguridad.

Topología en estrella
Cuando varias estaciones de trabajo se interconectan a través de un nodo central. Este nodo puede actuar como un distribuidor de la información generada por una terminal hacia todas las demás estaciones de trabajo o puede hacer funciones de conmutación. Los nodos son implementados mediante equipos llamados *hubs* o concentradores. Este tipo de topología se recomienda para redes que tienen cinco o más estaciones de trabajo. Es más segura que la topología en bus y su costo de implementación es intermedio entre la topología bus y anillo. Aquí puede suceder

que, si una estación de trabajo no tiene comunicación en la red, las otras pueden estar trabajando normalmente.

Topología de anillo
Se llama así por su forma de anillo y su uso está bastante extendido. En esta topología son raros los embotellamientos y su software es sencillo. Si falla un módulo del sistema, o incluso si se corta el cable, la señal se transmitirá y seguirá funcionando. La desventaja más grande es que el cable es más caro y complejo que el de otros sistemas y es más difícil localizar averías.

Topología en malla
Es muy empleada en las redes de área amplia (WAN), por su ventaja frente a problemas de tráfico y averías, debido a su diversidad de caminos o rutas y la posibilidad de orientar el tráfico por trayectorias opcionales. Su desventaja esta en su implementación, ya que es cara y compleja, pero aún así, muchos usuarios la prefieren por su confiabilidad. Un ejemplo de esta red es Internet, llamada la Telaraña Mundial o Red de redes.

2.2 Generación de páginas Web con HTML

HTML es un lenguaje de programación que se utiliza para la creación de páginas en la WWW. No es un lenguaje de alto nivel como C o Visual Basic. En lugar de compilarse y ejecutarse, HTML se compone de una serie de comandos que son leídos, o interpretados, por un agente usuario conocido como navegador Web.

El HTML se utiliza, principalmente, para crear páginas Web. *Tim Berners-Lee,* programador en el *Centro Europeo de Física de Partículas (CERN*). Berners-Lee desarrolló el HTML para:

- Proporcionar un medio en el que los científicos pudieran publicar sus trabajos, utilizar sus recursos y recuperar los datos, 24 horas al día.
- Crear un lenguaje internacional de codificación de ordenadores que facilitará el acceso universal, independientemente de la plataforma, red o terminal.

El lenguaje html evoluciona rápidamente y en ocasiones programas antiguos no entienden nuevas etiquetas, ya que las consideran erróneas y no realizan la acción que deseábamos. Por lo tanto, cuando creamos código html hay que hacerlo lo más estándar posible para permitir que el documento pueda ser visto de forma efectiva por distintos navegadores en máquinas distintas.

Cuando se almacena un fichero es conveniente que se le ponga la extensión .html, que es el tipo de fichero que por defecto buscará el visualizador, (aunque puede visualizar ficheros con otra extensión).

2.2.1 Partes fundamentales de un documento Html

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

La *estructura de un documento HTML* será:

```
<HTML>
<HEAD>
Definiciones de la cabecera
</HEAD>
<BODY>
Instrucciones HTML
</BODY>
</HTML>
```

La etiqueta <TITLE> (es opcional), debe ser usada dentro de las etiquetas que definen la cabecera de la siguiente forma:

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

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. 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.

Para representar los distintos colores se usa el siguiente formato:

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

- BGCOLOR=#rrvvaa ó *nombre del color pero en inglés.*

Indicará el color del fondo del documento HTML, sólo se utilizará si no se ha definido una imagen de fondo, o si esta imagen no puede obtenerse.

- TEXT=#rrvvaa ó *nombre del color*:

Especificará el color del texto normal dentro del documento HTML. Por defecto será normalmente negro.

- LINK=#rrvvaa ó *nombre del color*:

Indicará el color que tendrán los hiperenlaces que no han sido accedidos. Por defecto será azul.

<P> Definirá un párrafo, se usará al comienzo o al final de un párrafo de texto e introducirá una separación (normalmente dos líneas) con el próximo texto que se exprese. Esta etiqueta se puede utilizar para introducir un espacio entre cualquier elemento HTML y no sólo servirá para separar texto.

El efecto se conseguirá introduciendo la etiqueta <P> en el punto en el que deseemos que se produzca el espacio. La etiqueta de fin de párrafo </P> es opcional no siendo necesario incluirla.

Existen elementos HTML que ya introducen separaciones de líneas, tanto antes como después. No es imprescindible utilizar esta etiqueta ni antes ni después de cabeceras <H*n*>, después de <HR> (reglas horizontales), <ADDRESS>, <BLOCKQUOTE> ó <PRE> . Tampoco se requiere dentro de listas, tras los elementos <LI>, <DT> ni <DD>, que se utilizan para separar los distintos elementos de una lista.

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

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

Se usa para dividir un documento en distintas secciones; mostrará una línea horizontal de tamaño determinable. Se asemejará al salto de página dentro de un documento.

Si no se especifican atributos dibujará una línea que ocupe el ancho de la pantalla del navegador, introduciendo una separación con el texto anterior y siguiente, equivalente a cambio de párrafo. No es necesaria la etiqueta de fin *</HR>.* Con los atributos podemos especificar su forma y tamaño.

<PRE> Texto preformateado

Muestra todo lo que se encuentre entre las etiquetas de inicio y fin de texto preformateado; se mostrará tal y como se expresa en la fuente del documento html.

<CENTER> Centrado de texto e imágenes

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

```
<CENTER> El texto y la imagen se centran<BR> <IMG SRC="/imagenes/mundo.gif"> </CENTER>
```

Representación de caracteres especiales

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

Estas secuencias de escape comienzan todas con el símbolo ampersand (&), seguido de un texto (siempre en minúsculas) que define el caracter deseado y termina con el símbolo punto y coma (;). No es necesario dejar espacios en blanco ni antes ni después de los caracteres especiales. Ejemplos:

á	**á**
ñ	**ñ**
¿	**¿**

Cabeceras de títulos <H1> - <H6>

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

<FONT SIZE=*n*> : Tamaño de la fuente

El atributo SIZE permite indicar el tamaño de la fuente, su valor puede estar entre 1 y 7. Incrementándose de tamaño progresivamente desde 1, que es la fuente de menor tamaño, hasta 7 que la fuente de mayor tamaño.

<FONT COLOR=*texto color* ó *rrvvaa*>: Color de la fuente

El atributo COLOR nos permite definir el color que tendrá el texto incluido entre las etiquetas de inicio y fin. Este atributo sólo funciona en el *Internet Explorer de Microsoft* y en el *Netscape* versión 2.0 o superior. El modo de definir los colores es similar al explicado para los atributos de BODY.

Tipos de letra

Con html se pueden especificar distintos tipos de letra. Los básicos son negrita, cursiva y courier, que utilizan los códigos <B>, <I>, <TT> respectivamente y <U> subrayado.

Listas

La información se puede representar como listas ordenadas (números) y como no ordenadas (marcas, como círculo o cuadrado). En ambos casos el párrafo debe iniciar con <L1>. Otro tipo de listas se utiliza para presentar la información sin marcas, ni números sino jugando con las sangrías de los párrafos. <OL> se utiliza para las sangrías.

<DL> sirve para listas de definición. Donde se utiliza <DT> para indicar el término que lleva sangría y <DL> para indicar su definición que no la llevará. También pueden anidarse.

2.2.2 Hipertexto

El hipertexto se basa en el concepto de enlace. Un enlace es una referencia a un documento HTML, o a cualquier otro recurso de WWW, expresado por medio de un formato universalmente aceptado que lo identifica unívocamente. Esto permite que el navegador, cuando detecta que se quiere "saltar" a un documento determinado, sea capaz de localizarlo y traerlo.

Referencias locales

Este tipo de referencias se orienta a documentos locales, de nuestro propio servidor. Son ficheros almacenados en la misma máquina y en directorios cercanos al del documento que contiene el enlace, lo que simplifica las cosas.

La directiva responsable de los enlaces es <A> (del inglés anchor, ancla). Delimita un texto o una imagen que al ser seleccionado por el usuario, hace que el browser inicie la apertura del elemento referenciado. La dirección del documento se indica por medio del atributo href:

```
<a  href="./info.html">Información</a>
```

Referencias a documentos html externos

Las tres características que hacen atractiva a la WWW, frente a otros sistemas basados en Internet, son hipertexto, multimedia y la navegación por la red. Esta última funcionalidad, también conocida como net surfing, consiste simplemente en la utilización de referencias a documentos externos, de forma que un simple clic nos puede llevar a un servidor situado en el otro extremo del mundo.

Las referencias a documentos HTML externos se construyen de la siguiente manera:

http:///(dirección del servidor)/(directorio)/(pagina)

<A href="http://www.google.com">Google</A>

2.2.3 Inclusión de imágenes <IMG>

Hace uso es9encialmente de un atributo src, que sirve para indicar la dirección de la imagen que se está empleando:

<IMG src=".../imagenes/mundo.jpg" title="Mundo">

Se puede ubicar en cualquier lugar como un elemento más del texto, e incluso se ve afectada por la acción de directivas como <CENTER>, <A>.

Comentarios <!--, utilizaremos la etiqueta especial, definiéndose un comentario de la forma: <!-- Esto es un comentario -->

2.2.4 Formularios

Elementos creados con la etiqueta INPUT (Esta etiqueta tiene un gran número de atributos).

<INPUT

type = " text | password | checkbox | radio | submit | reset | file | hidden | image | button"

name= "nombre" <!--Se usa para asignar un identificador al campo--

value= "valor" <!--Se usa para asignar al campo un valor inicial--

size="tamaño"> <!--Indica la anchura para su visualización para los campos de texto--

Text (Texto)

Se utiliza para dar entrada a una sola línea de texto.

Sintaxis: <INPUT

type=”text”

name=”nombre”

value=”valor”

size”tamaño”

maxlength=”LongitudMaxima”>

Sigmund

Password (contraseña)

Es una variante del control text, ya que los caracteres que teclea el usuario no son visibles en la pantalla. En su lugar aparece un caracter máscara, por ejemplo un asterisco. Sintaxis:

La sintaxis es parecida a la de text; lo que cambia es: type=”password” Sigmund

Checkbox (casilla de verificación)

Se utiliza para presentar una opción que es verdadera o falsa.

Sintaxis: type=”checkbox”

name=”nombre”

value=”valor”

checked> <!--Indica que tiene un estado inicial seleccionado --

Casilla de verificación 1
Casilla de verificación 2

Radio (opción)

Los controles de opción como las casillas de verificación, se usan para una opción que es verdadera o falsa. Cuando hacemos clic sobre uno, todos los demás quedan desactivados.

Sintaxis:

La sintaxis es muy parecida a la de checkbox, lo que cambia es el tipo: type=”checkbox”


si
no

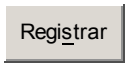
Submit (envío)

Para enviar un formulario al servidor, debe haber un modo de indicar que el usuario ha terminado de introducir datos.

type=”submit”

name=”nombre”

value=”valor”



Reset (reestablecer)

El control Enviar está relacionado con el control reestablecer (reset) que se usa para reestablecer todos los valores del formulario a sus originales. Como en el botón enviar, el botón reestablecer no aporta ningún dato al flujo de datos que se manda al servidor.

Sintaxis: type=”reset”

value=”valor”

File (fichero)

Algunas aplicaciones cargan ficheros para transferirlos al servidor. Las páginas Web que sirven de interfaces para este tipo de aplicaciones pueden utilizar el control de entrada File. Y así tenemos el Hidden, otros tipos de entrada y de etiquetas.

2.2.5 Tablas

<TABLE> Directiva que delimita el conjunto de una tabla. No es preciso determinar inicialmente el número de filas o columnas; el browser se encargará de averiguarlo

a medida que profundice en su contenido. Cada celda se delimita con <TD> (table data), cada fila con <TR> (table row) y cada elemento de cabecera con <TH> (table head).

2.2.6 Frames

Los Marcos o Frames permiten mostrar documentos en diferentes vistas que pueden ser ventanas independientes o subventanas.

Con múltiples vistas se puede mantener información desplegada mientras que otra es reemplazada. Se puede mantener visible en el primer plano el título, en el segundo plano un menú de navegación (como links) y en el tercero el documento principal. Ver figura 2.3

Fig. 2.3 Frames

Para construir el frame anterior se necesita el siguiente código:

```
<HTML>
        <HEAD>
                        <TITLE>SEGURIDAD INFORMÁTICA</ TITLE>
                        <FRAMESET ROWS =20%,*>
                                        <FRAME SRC ="cabecera.html"  NAME="cabecera"
MARGINWIDTH="10"  MARGINHEIGHT="10"  NORESIZE>
                                        <FRAMESET COLS =17%,*>
                                                <FRAME SRC ="links.html"
NAME="links"  NORESIZE>
                                        <FRAME SRC ="central.html"  NAME="central">
                                        </FRAMESET>
                        </FRAMESET>
        </HEAD>
</HTML>
```

Donde:

ROWS.- Marcos Horizontales.
COLS.- Marcos Verticales.
SRC.- Localización del contenido inicial que se contendrá en el marco.
NORIZE:- Dice al usuario si la ventana del Marco no puede variar su tamaño.
NOTA: Aquí estamos mandando a llamar a otros documentos Html que ya hemos realizado con anterioridad como cabecera.html.

2.3 Introducción a un servidor configurable para diferentes entornos de trabajo

2.3.1 ¿Qué es un servidor WWW?

Es un programa encargado de ofrecer comunicación mediante el protocolo HTTP y servicios dentro del World Wide Web en Internet. Fig. 2.4

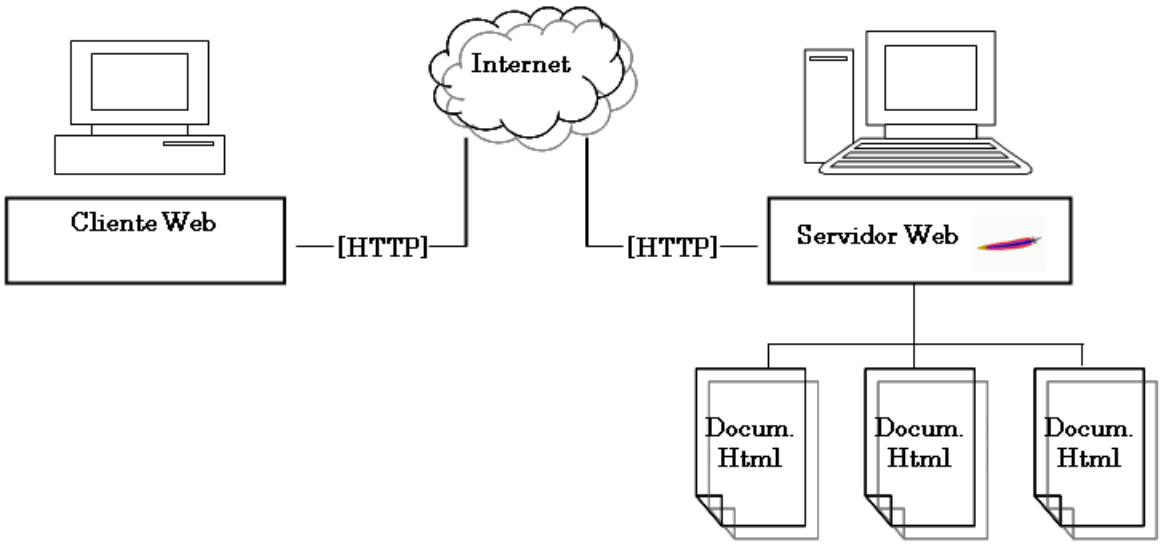


Fig. 2.4 Servidor WWW

¿Cómo funciona HTTP?

HTTP[3] es el protocolo (simple y poderoso) de red para WWW; el cual basa su operación en la arquitectura "Cliente – Servidor".

- El servidor http es el encargado de publicar "recursos" electrónicos.
- El cliente http consulta los recursos que el servidor ofrece.
- El cliente abre una conexión.
- El server manda un " acknowledge" notificando que se ha abierto una sesión.
- El cliente envía un " request message" solicitando un recurso.
- El servidor responde con un "response message" que contiene el recurso solicitado y cierra la conexión.

Servidores de WWW populares

• 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

2.3.2 Razones para utilizar Apache

• Configurable para diferentes entornos de trabajo, con un alto nivel de seguridad.
• Disponible para una gran variedad de plataformas.
• Soporte para servicio de proxy.
• Soporte para scripting languages integrados como módulos (por ejemplo PHP, mod_perl).
• Incluye el código fuente del servidor.
• Soporte para accesos restringidos.
• Soporte para SSL.
• Robusto, soporte de un gran número de transacciones, es libre y gratuito.

2.3.3 Instalación de Apache

- Por paquetes:
  - ❖ apt-get install (Debian)
  - ❖ emerge (Gentoo)
  - ❖ rpm (Red Hat, Mandrake, SuSe)
  - ❖ installpkg (Slackware)

- En cualquier Unix (Compilación)

```
$ tar zxvf httpd-2.0.54.tar.gz
$ cd httpd-2.0.54
$ ./configure --prefix=/usr/local/apache
$ make
# make install
```

[3] HTTP: Protocolo de Transferencia de Hipertexto

Para levantar Apache nos tenemos que posicionar en:
cd /usr/local/apache/bin

Y escribir:

./apachectl start (para iniciar)

./apachectl stop (para detenerlo)

./apachectl restart (para reiniciarlo)

2.3.4 Configuración básica del servidor Web Apache.

Directivas

• Las directivas son parámetros de configuración que determinan el comportamiento del servidor Web.
• El administrador controla qué directivas estarán disponibles de acuerdo a los módulos que integre en Apache.
• Apache es administrado por más de 200 directivas.

Grupos de Directivas

La configuración de Apache se rige mediante directivas y éstas se clasifican en tres grupos:
• Global Environment
• Main Server
• Virtual Servers
• La sección Global Environment administra las directivas generales de operación del Servidor de Web.
• 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's o nombres de dominio.

Listen

Esta directiva define el puerto en cual escuchará el servidor Web.
Es posible indicar a Apache que escuche en múltiples puertos.
Valor por default: 80

ServerAdmin

Esta directiva define el correo electrónico del administrador del servidor Web e indica la dirección a incluirse en los mensajes de error que serán enviados al cliente.
Es necesario tener habilitada la directiva:
ServerSignature Email

ServerName
Define el nombre para el servidor Apache, el cual será utilizado al construir los URLs cuando el cliente solicite otros recursos Web (html, phps, cgis, etc.) del servidor.

ServerType *(Eliminado en 2.0)*
Define el esquema mediante el cual operará Apache, las posibles opciones son:
- standalone: Operando de forma independiente.
- inetd: Operando dependiente del superdeamon inetd.

Nota: *No se recomienda inetd debido a los recursos de cómputo que se consumen al obligar al superdemonio (inetd o xinetd) a generar un proceso httpd cada vez que se genere una nueva conexión.*

DocumentRoot
Esta directiva define la ruta absoluta donde se almacenarán los archivos html que se desean publicar.
Ejemplos comunes de Document Root:
• / usr/local/apache/htdocs
• / var/www/html
Esta directiva se utiliza tanto en el Main Server como en los VirtualHosts.

ServerRoot
Esta directiva define la ruta absoluta donde se instala Apache, o donde los directorios conf, logs, bin, modules podrán ser encontrados.
Ejemplos comunes de Server Root:
• / usr/local/apache
• / var/www

MaxSpareServers
Esta directiva define el número máximo de procesos servidores que son mantenidos en reserva “spare”.
Apache monitorea el número de procesos en “spare” si se supera el valor de “MaxSpareServers” se eliminan los daemons necesarios, siempre y cuando no estén realizando alguna tarea.

StartServers
Esta directiva define el número de servidores que se iniciarán cuando se arranca Apache.
Valor por Default: 5

MaxClients
Esta directiva define el número máximo de procesos servidores que podrán ser iniciados, con el objeto de atender las solicitudes de los clientes.
Apache no debe superar este valor, es el tope máximo de procesos que puede mantener en operación.
Valor por Default: 20

ErrorDocument

Si al recibir la solicitud por un recurso ocurre un problema con Apache, es posible configurarlo para que haga una de las siguientes acciones:
1. Enviar un mensaje de error (default).
2. Enviar un mensaje personalizado
3. Redirigir a un URL local quien manejará el problema o error.
4. Redirigir a un URL externo quien manejará el problema o error.

PidFile
Define la ruta del archivo PidFile donde se almacenará el PID, el cual es un valor numérico que identifica al proceso httpd "padre".
Ejemplo:
Pidfile logs/httpd.pid
Nota: Si este archivo se pierde es común tener problemas cuando arranca o se detiene Apache, vía el script apachectl.

Bitácoras - Logging
Apache cuenta con dos archivos donde se registran los mensajes sobre el estado del servidor. Las bitácoras son:
• access_log
Registra todos los accesos al sitio.
• error_log
Registra los errores que generen los accesos al sitio o aquellos que los procesos de Apache reporten.

Logging
ErrorLog <filename>
Esta directiva define el nombre del archivo donde el servidor Web, Apache, registrará los errores que se presenten durante su operación.
Si <filename> comienza con una "/" se considera que la ruta tiene origen en la raíz del Filesystem, sino se toma como origen el ServerRoot.

Logging
CustomLog <filename>
Esta directiva define el nombre del archivo donde Apache registrará los accesos de los visitantes que se conecten al servidor Web.
Si <filename> comienza con una "/" se considera que la ruta tiene origen en la raíz del Filesystem, sino se toma como origen el ServerRoot.

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

HostNameLookups [on|off]
Nota: Por razones de rendimiento se sugiere no habilitarla. Para la resolución de nombres en las bitácoras existe el programa logresolve, también de Apache, que puede realizar la resolución de nombre fuera de línea.

CAPÍTULO 3

INTERACCIÓN CON BASES DE DATOS VÍA WEB POR MEDIO DE PROGRAMACIÓN PHP

OBJETIVOS ESPECÍFICOS

Explicar:

- Las diferencias entre los lenguajes de programación interpretados y los compilados, así como sus ventajas y desventajas.
- El proceso de instalación y programación en PHP.
- El concepto de bases de datos y los elementos que las conforman.
- Las características de los lenguajes del DBMS.
- Cómo interactuar con MySQL mediante programación PHP.

JUSTIFICACIÓN

Existen diversas herramientas para la creación de páginas Web dinámicas. La creación de éstas exige la utilización de bases de datos y programación.

PHP como lenguaje de programación y MySQL como lenguaje de gestión de bases de datos, constituyen un entorno de desarrollo de software orientado a la creación, gestión y utilización de sitios dinámicos e interactivos.

Una de las mejores opciones de PHP es que el código se puede introducir directamente en código Html, sirve como nexo de unión entre la base de datos y los clientes potenciales de una aplicación Web, además, es el lenguaje que fue utilizado en el diplomado “Desarrollo e Implementación de Sistemas con Software Libre en Linux”.

Por otro lado, MySQL es una base de datos de código libre, muy rápida y comparativamente fácil de administrar; inclusive es considerada una base de datos estándar para sitios Web dinámicos en sistemas Linux debido a su facilidad de uso, velocidad y ligereza.

3. INTERACCIÓN CON BASES DE DATOS VÍA WEB POR MEDIO DE PROGRAMACIÓN PHP.

3. 1 Lenguajes de programación interpretados

Son aquellos que van siendo codificados por la computadora al mismo tiempo que se están ejecutando. Es decir, un intérprete es un traductor de lenguajes de programación de alto nivel, el cual realiza el proceso de compilación paso a paso, los intérpretes ejecutan un programa línea por línea.

El programa siempre permanece en su forma original (programa fuente) y el intérprete proporciona la traducción al momento de ejecutar cada una de sus instrucciones, es decir, el programa será ejecutado sin necesidad de ser codificado antes, de encontrarse un error la ejecución se detendrá en el comando o acción errónea.
No se genera un archivo binario. Ejemplos ASP, PHP, PERL.

3.1.1 Ventajas y desventajas del lenguaje interpretado

Ventajas
Se puede editar y comprobar de forma rápida, ya que no requiere volver a compilar. Es más cómodo realizar una aplicación ya que las fases de ejecución y edición están más integradas, y dependiendo del lenguaje es portable con la edición casi nula del código.

Desventajas
Es más lento que los lenguajes compilados ya que no produce un código objeto y recorre el código fuente cada vez que son ejecutados, además que no protegen la implementación ya que se requiere el script para su ejecución.

3.2 Lenguajes de programación compilados

Son aquellos que necesitan ser codificados antes de ser ejecutados y obtener resultados, de encontrarse un error a la hora de codificar los comandos del programa, éste nunca podrá ser ejecutado. Un compilador es un programa que traduce el lenguaje de alto nivel a lenguaje máquina de una computadora.

Según va ejecutando la traducción verifica, comprueba y coteja los errores hechos por el programador. Un compilador traduce un programa una sola vez generalmente. Un programa compilado indica que ha sido traducido y está listo para ser ejecutado.

La ejecución de los programas compilados es cinco veces más rápida que la de los programas interpretados, ya que el intérprete debe traducir mientras está en la etapa de ejecución. Ejemplos Java, C, C++.

3.2.1 Ventajas y desventajas del lenguaje compilado

Ventajas
Es más rápido que el lenguaje interpretado, ya que no ejecuta el código cada vez que se manda a llamar, sino que produce un código objeto que es el que se ejecuta y protege debido a que sólo existe un binario y no se requiere del código fuente.

Desventajas
El ciclo de edición requiere de más tiempo ya que cada vez que se realiza un cambio requiere que se vuelva a ejecutar el código, no es portable (a excepción de Java) y requiere del código fuente para sus modificaciones.

3.3 Instalación de PHP

PHP (Hypertext Preprocesor). Es un procesador de hipertexto, es decir, con PHP se puede crear programas cuyo resultado sean páginas Html. [1]

Un archivo PHP es un archivo de texto, no es necesario compilarlo y puede leerse con cualquier editor de texto.[2]

Se puede instalar apache nuevamente, pero con otra ruta para no tener conflictos con el apache anterior, en dado caso que se hallan hecho modificaciones; o de lo contrario se puede trabajar con el que se instaló en el capítulo anterior. El siguiente es el proceso de instalación.
tar –zxvf httpd 2.0.54 tar.gz
cd 2.0 54
./configure --prefix=/usr/local/apache4
--enable-module=so ****opcional****
make clean
make
make install
Nos posicionamos en la siguiente ruta para buscar un módulo
/usr/local/apache4/bin
./httpd mod_so.c

[1] Juan Diego Gutiérrez Gallardo, *Desarrollo Web con PHP5 y MySQL*. Guía practica para Usuarios. Anaya Multimedia, p. 43.
[2] Ibidem p. 49.

Se puede instalar de 2 formas:

- *Como CGI*

Disminuye el rendimiento ya que es el sistema operativo quien se encarga de gestionar todos los procesos derivados de la ejecución del script de PHP.

- *Como Módulo*

Es más rápido y eficiente ya que las ejecuciones del programa las realiza el servidor Web.

NOTA: Además si se instala como CGI tanto Apache como PHP se tienen que recompilar en caso que se agregue soporte adicional por parte de PHP, y como Módulo sólo se recompila PHP.

Instalación de PHP

tar-zxvf php-4.3.x.tar.gz
cd php-4.3.x/
./configure --prefix=/usr/local/php4 --with-apxs2=/usr/local/apache4/bin/apxs --with-mysql
make
make install
cp php.ini-dist/usr/local/php4/lib/php.ini
Agregar las siguientes líneas al archivo httpd.conf:
AddTypeapplication/x-httpd-php.php.phtml
AddTypeapplication/x-httpd-php-source.phps

Verificar que exista la línea:
LoadModulephp4_module modules/libphp4.so
Crear un archivo llamado info.php en el Document Root de Apache
/usr/local/apache4/htdocs con el siguiente contenido:

```
<? phpinfo(); ?>
```

Reiniciar Apache.
Teclear en un navegador:http://localhost/info.php

CGI

CGI es una norma para establecer comunicación entre un servidor Web y un programa, de tal modo que este último pueda interactuar con Internet. También se usa la palabra CGI para referirse al programa mismo, aunque lo correcto debería ser script.

Es un programa que se ejecuta en tiempo real en un Servidor Web en respuesta a una solicitud de un Navegador. Cuando esto sucede el Servidor Web ejecuta un proceso hijo que recibirá los datos que envía el usuario (en caso de que los haya), pone a disposición del mismo algunos datos en forma de variables de ambiente y captura la salida del programa para enviarlo como respuesta al Navegador.

Propósito de los CGI´s

- Generar páginas de forma dinámica.
- Procesamiento de formularios.
- Interacción con Bases de datos.
- Comercio electrónico.
- Lectura y escritura de archivos.
- Motores de búsqueda.
- Foros de discusión.

3.4 Código en PHP

```
<html>
 <head>
        <title>Ejemplo de PHP</title>
 </head>
 <body>
        <?php echo "<p>Hola Mundo</p>"; ?>
 </body>
</html>
```

Variable
Es el nombre que se le da a una posición de la memoria en la cual se guarda la información.

Sintaxis
$nombre_de_variable
El nombre de las variables puede tener letras, números y guiones.
NOTA: PHP es case sensitive porque diferencia entre mayúsculas y minúsculas.

Tipos de datos simples
Integer.
Enteros, números que varían entre –2 billones y 2 billones, se puede representar en formato decimal, octal o hexadecimal.

Float
Números con decimales: -1.48, 8.32e+11, 5.6e-3.

String
Cadena formada por cero o más caracteres encerrados entre “” ó ‘ ’.
Hay caracteres especiales para las cadenas \n cambio de línea ‘$ caracter’.

3.4.1 Datos compuestos

Array (Arreglos)
Estructuras que permiten el almacenamiento de un conjunto de datos bajo un mismo nombre. El primer elemento del arreglo tiene el índice cero.

Se puede crear a través del constructor:
array (“lunes”, “martes”, 1, 2.34);

Son arreglos especiales en los que el índice es un valor de tipo string, de modo que cada elemento del arreglo esta definido por el par (clave, valor).
Son muy usados en PHP. Se pueden definir a través del constructor como:
$arreglo = array (‘día’ => lunes, ‘costo’ => 2.34);

Object (Objetos)

Estructura que define características propias (denominadas propiedades) y sus funcionalidades denominadas métodos.

Operadores aritméticos	*Operadores de asignación*
+ $a + $b	= $a = $b
- $a - $b	+= $a += $b => $a = $a + $b
* $a * $b	-= $a -= $b
/ $a / $b	+= $a *= $b
% $a % $b	/ = $a /= $b
	%= $a %= $b
	.= $a .= $b

3.4.2 Conversión automática de tipos

Cuando operamos con variables de distinto tipo el intérprete de PHP tiende a homogeneizar sus distintos tipos en función de la operación que se pretende realizar.

Una constante es una variable que mantiene el mismo valor durante la ejecución del programa.
Constantes predefinidas de PHP:
PHP_VERSION, PHP_OS, TRUE, FALSE.

Sintaxis de definición de constantes
define(“constante”, valor)

Operadores de cadenas

.	Operadores de concatenación de dos cadenas
.=	Concatenación y asignación

Operadores de incremento y decremento

++ ++$a	Incrementa $a en uno y después devuelve $a
$a++	Devuelve $a y después lo incrementa
-- --$ay	Predecremento
$a--	Posdecremento

Operadores de comparación

Se utilizan para comparar expresiones, el resultado es true o false; usados en la toma de decisiones.

==	$a==$b
!=	$a!=$b
===	$a===$b (Idénticos iguales y del mismo tipo).
!==, < , > , <= . >=	

Operadores lógicos

Se utilizan con expresiones que devuelven valores lógicos, con estos operandos se pueden combinar varias expresiones y evaluarlas en una sola expresión:

&& ó and	$a && $b	TRUE si a y b son TRUE
|| ó or	$a || $b	TRUE si a ó b son TRUE
!	!a	Niega el valor lógico
xor	$a xor $b	TRUE si alguno de los dos es TRUE pero no los dos a la vez

3.4.3 Estructuras de control

Nos permiten modificar el flujo de ejecución de un programa, logrando que la ejecución no tenga que ser secuencial sino que nos permite bifurcar el flujo del programa (estructuras condicionales) o que determinado código se ejecute un cierto número de veces (estructuras cíclicas).

Estructuras condicionales

Son estructuras que nos permiten elegir diferentes caminos de ejecución. Cuando se cumple una determinada ejecución. En PHP existen 2:

if

Sintaxis

```
If(expresión){
Sentencias;
}
```

Nota: Debe ser una expresión lógica, es decir, que devuelva verdadero y falso.

if … else

Sintaxis:

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

switch

Se emplea para comparar un dato con un conjunto de posibles valores.

Sintaxis:

```
switch($variable){
    case valor1:
    sentencias;
    break;
    case valorN:
    sentencias;
    break;
    default:
    sentencias;
}
```

3.4.4 Código para realizar una calculadora pequeña

Se crea el formulario en html (código practica2formulario.html), después el programa en PHP (código practica2calculadora.html), el cual va a permitir que se efectúen las operaciones haciendo uso del formulario que se realizó anteriormente (ver figura 3.1).

practica2formulario.html

```
<html>
<head>
<title> FORMULARIO <title>
</head>
<body>
<h1 <lign = "center"> FORMULARIO </center></h1>
<BR><BR>
<form method="post" action="calculadora.php">
OPERANDO1:<input type="text" name =operando1" size="15" value "5">
FORM METHOD="GET">
<select name= "tipoOperacion" selected>
<option value "MULTIPLICACION"> MULTIPLICACION </option>
<option value "SUMA"> SUMA </option>
<option value "RESTA"> RESTA </option>
<option value "DIVISION"> DIVISION </option>
<option value "FACTORIAL"> FACTORIAL </option>
<option value "POTENCIA"> POTENCIA </option>
</selected>
OPERANDO2:<input type="text" name="operando2 size="15" value="5"><br>
<input type="submit" value=”REALIZAR OPERACIÓN">
</form>
</body>
</html>
```

Código practica2formulario.html

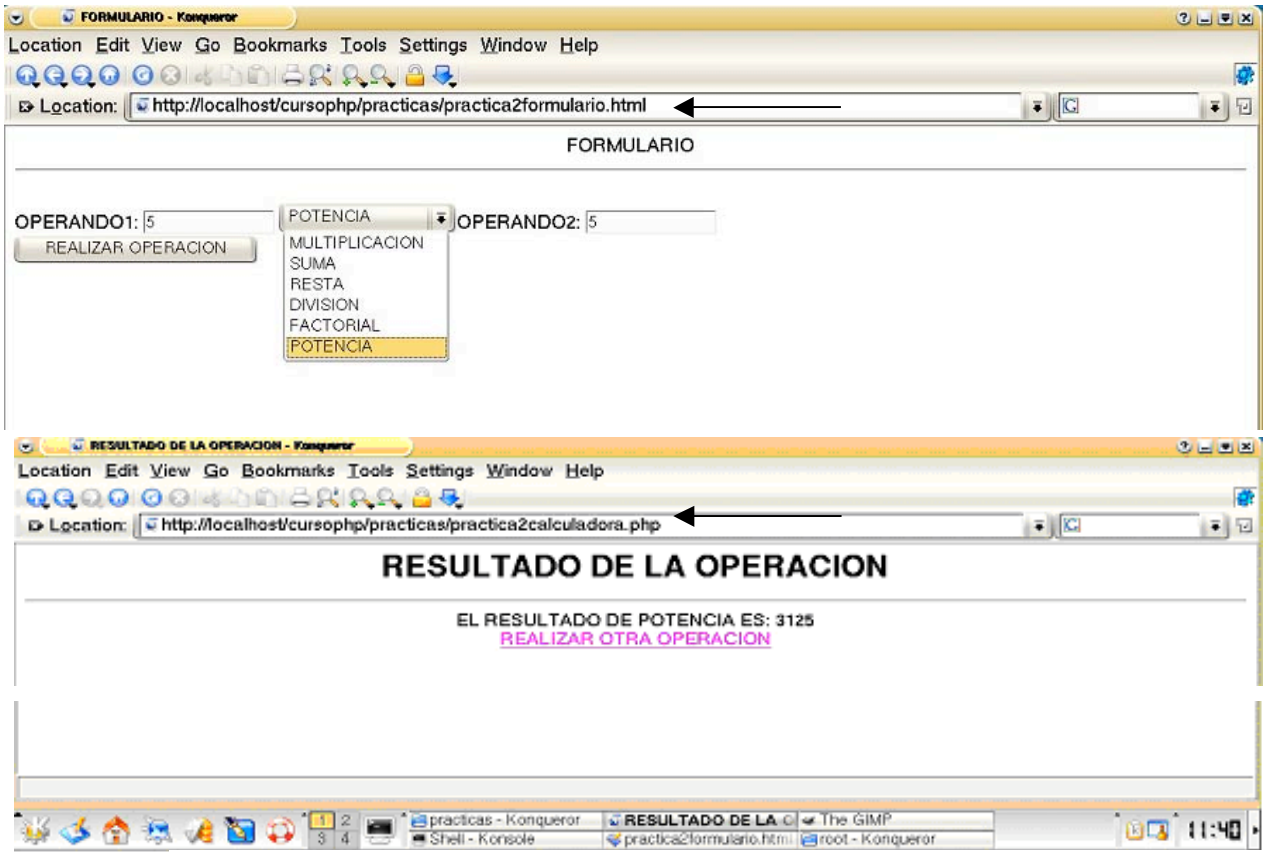


Fig. 3.1 Pantallas al ejecutar los programas practica2formulario.html y practica2calculadora.php

practica2calculadora.php

```
<html>
<head>
<title> CALCULADORA <title>
</head>
<body>
<h1 <lign = "center"> CALCULADORA </h1><hr>
<?
$temp1 = (float) $HTTP_POST:VARS[operando1"];
$temp2 = (float) $HTTP_POST:VARS[operando2"];
$tipo =$HTTP_POST_VARS["tipoOperacion"];
resul=0.0;
$error=""
if (strcasecmp ($tipo, "multiplicacion") == 0)
{
resul=$temp1 * $temp2;
}
elseif (strcasecmp ($tipo, "suma") == 0)
{
resul=$temp1 + $temp2;
}
if (strcasecmp ($tipo, "division") == 0)
{
if (temp == 0)
            $error = "no se puede realizar operación";
  else
$resul= $temp / $temp2;
elseif (strcasecmp ($tipo, "factorial") == 0)
{
if ($temp1 < 0)
$error="Imposible calcular el factorial de números negativos";
else
{
$resul=1;
            for( $i=1; $i<$temp1, $i++)
            Sresul *= $i;
}
elseif (strcasecmp ($tipo, "potencia") == 0)
if ($temp2 < 0)  //para # negativos
$resul=1; //Debido a que se había declarado con valor de cero
$temp2 *= -1;
for ($i=1; $i<=temp2; $i++)
$resul = /$resul;
}
else
{
$resul = 1;
for ($i=1; $i<=$temp2, $i++)
            $resul *=temp1;
}
}
else
{
$error="Imposible realizar operación";
}
if (strcasecmp ($error, " ") == 0)
echo "El resultado de $tipo es= $resul";
else
echo $error;
echo "<br>";
echo "<a href=\"practica2formulario.html\">";
echo "REALIZAR OTRA OPERACIÓN";
echo "</a>";
?>
</body>
</html>
```

Código practica2calculadora.html

Estructuras cíclicas

Se usan para ejecutar una ó más instrucciones un determinado número de veces, generalmente se utilizan para contar o recorrer los elementos de un arreglo. Existen 4 tipos:

foreach

Se utiliza para recorrer las estructuras de tipo arreglo obteniendo en cada interacción uno de los elementos componentes.

Tiene 2 sintaxis

arreglos:

```
foreach(nombre_arreglo as $valor){
sentencias;
}
```

arreglos asociativos:

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

for

Nos permite realizar un conjunto de instrucciones un determinado número de veces.
Sintaxis:

```
for(inicialización; condición; incremento){
sentencias;
}
```

while

Se ejecuta un número indeterminado de veces, siempre y cuando el resultado de comprobar la condición sea verdadero.
Sintaxis:

```
while(condición){
sentencias;
}
```

do while

Es lo mismo que un ciclo while, la diferencia es que por lo menos se ejecuta una vez, ya que la condición se evalua al final del ciclo.

Sintaxis:

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

break y continue

break.- Se utiliza para forzar la terminación de un ciclo o en el caso de switch para que no se sigan evaluando los *case*. A continuación se presenta un ejemplo de *seleccionaimagen.php* (fig. 3.2).

continue.- Se utiliza dentro de los ciclos, cuando queremos que no se efectúen una serie de instrucciones del ciclo y queremos pasar a la siguiente iteración.

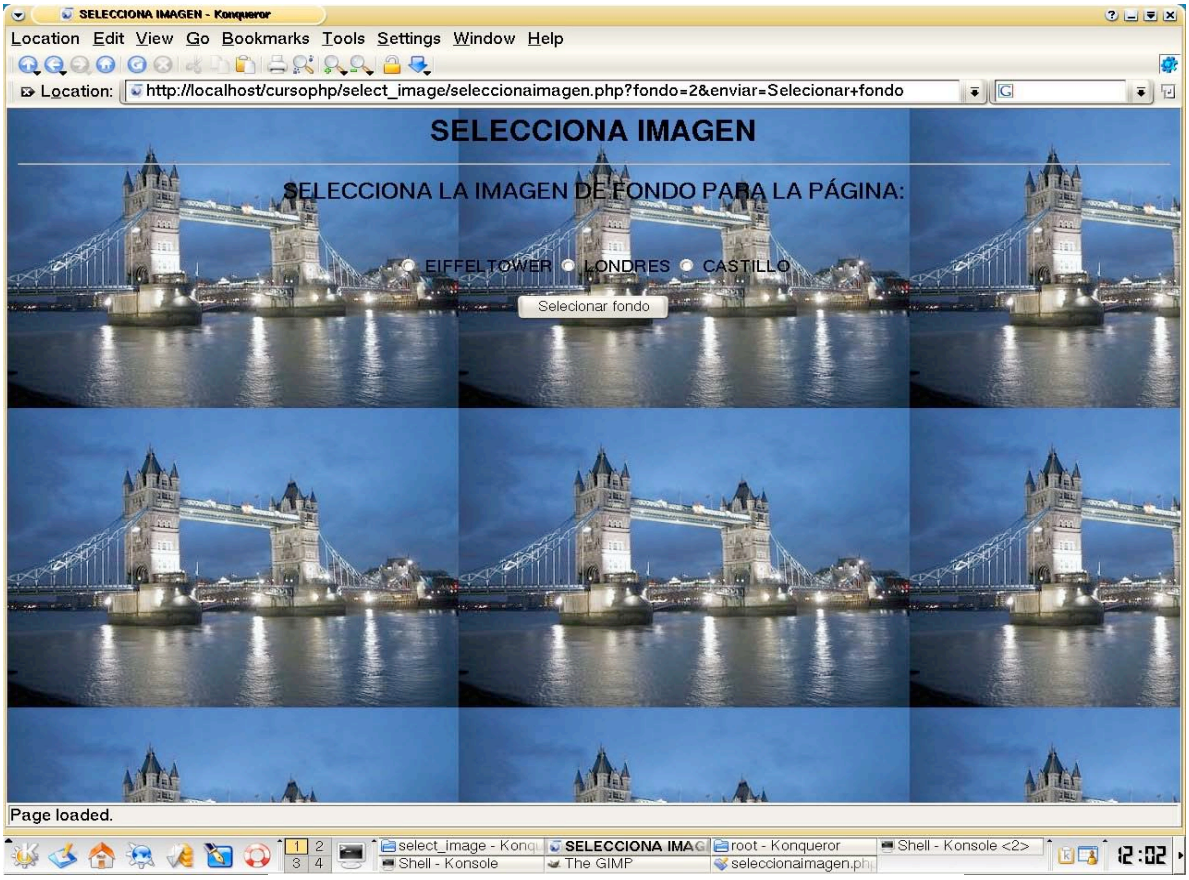


Fig. 3.2 Pantalla al ejecutar el código seleccionaimagen.php

seleccionaimagen.php

```
<html>
<haed>
<title> SELECCIONA IMAGEN </title>
<head>

<body>
<h1 align="center"> SELECCIONA IMAGEN </h1> <hr>

<?
If_($GET){
$ifondo=$_GET ['fondo'];
switch ($ifondo){
case1:
$imagen="eiffel.tower19.jpg";
break;
case2:
$imagen="londres.jpg";
break;
case3:
$imagen="loire_valley10.jpg";
break;
default;
$imagen="default.gif";
}
}
else
$imagen="default.gif";
?>

<body background="<? echo $imagen;?>">
<center>

<h2< SELECCIONA IMAGEN DE FONDO PARA LA PAGINA:<h2><br>
<form action="selecciona imagen.php" method="get">
<input type="radio "name="fondo "value="1">;
EIFFELTOWER
<input type="radio "name="fondo "value="2">;
LONDRES
<input type="radio "name="fondo "value="3">;
CASTILLO<br><br>
<input type="submit" value="SELECCIONAR FONDO" name="enviar">
</form>
</center>
</body>
</html>
```

3.4.5 Funciones

Son una sección separada de código que tiene un propósito específico, a la cual se le asigna un nombre. Se utilizan para dividir el código de un script en partes menores (modularidad).

Sintaxis:

```
function nombrefuncion (parámetros){
sentencias;
return valor;
}
```

Paso de parámetros a las funciones

Por valor.- La función recibe una copia del valor de la variable, al terminar de ejecutarse la función no se ha alterado el valor de la variable que fue pasada como argumento.

Ejemplo:

```
funcion iExponente ($a){
return $a*$a;
}
```

Por referencia.- Se altera el valor de la variable pasada por argumento a la función, esto es debido a que se traslada, en realidad una referencia de la variable y no una copia de su valor como en el caso anterior.

Ejemplo:

```
funcion iExponente ($a){
return $a*$a;
}
```

Por defecto.- Son parámetros opcionales en la llamada a las funciones, toman un valor predefinido en caso de que no se especifique el argumento en la llamada a la función.

Ejemplo.

```
funcion iExponente ($a,$exponente=2){
return $a*$exponente;
}
```

3.4.6 Inclusión de archivos

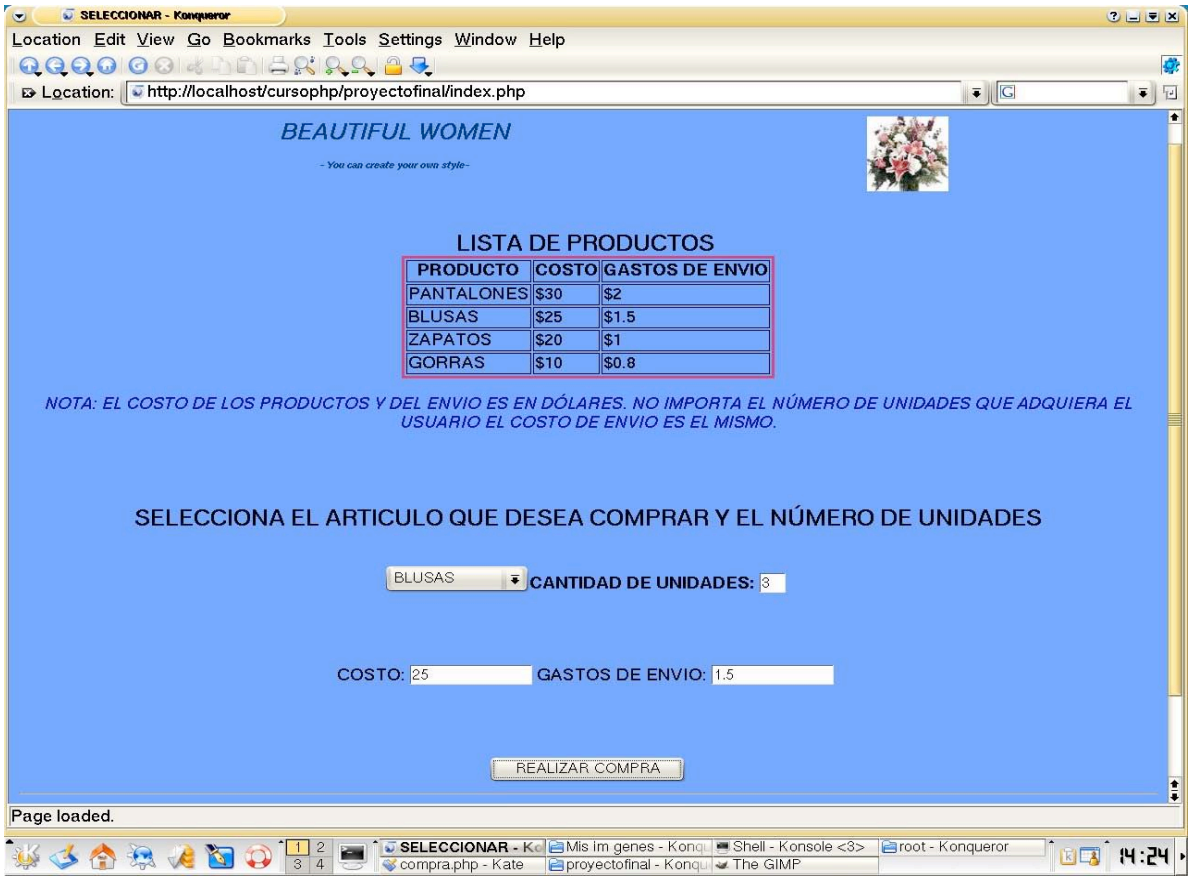
Se utilizan principalmente como la definición de librerías comunes a varios scripts, permitiendo la reutilización de código.

El siguiente programa hace uso del programa **compra.php** y **pie.php**

index.php

```
<html>
<head>
<title> SELECCIONAR </title>
</head>
<body bgcolor = "#FFFF00">
<?
include "cabecera.php";
?>
<form action="compra.php "method="post">
<CENTER>
<form>
<p><table border="3" BORDER COLOR="#0000FF">
<caption align="TOP"><h3>LISTA DE PRODUCTOS</h3></caption>
<tr><th> PRODUCTO </th>
                                        <th> COSTO </th>
                                        <th> GASTOS DE ENVIO </th>
</tr>
<tr><td>PANTALONES</td>
                                        <td>$30</td>
                                        <td>$2</td>
</tr><td>BLUSAS</td>
                                        <td>$25</td>
                                        <td>$1.5</td>
<tr><td>ZAPATOS</td>
                                        <td>$20</td>
                                        <td>$1</td>
</tr>
<tr><td>GORRAS</td>
                                        <td>$10</td>
                                        <td>$0.8</td>
</tr></table><center>
<p><i><font color="BLUE">
NOTA: EL COSTO DE LOS PRODUCTOS Y DEL ENVÍO ES EN DOLARES.
NO IMPORTA EL NÚMERO UNIDADES QUE ADQUIERA EL COSTO DE
ENVÍO ES EL MISMO <br><br><br<br>
</font></i></p>
<center>
<h3> SELECCIONA EL ARTÍCULO QUE DESEA COMPRAR Y EL NÚMERO
DE UNIDADES </h3><br>
<select name = "padquirido">
<form method="GET">
<option value= "gorras" selected= GORRAS </option>
<option value= "pantalones" selected= PANTALONES </option>
<option value= "blusas" selected= BLUSAS </option>
<option value= "zapatos" selected= ZAPATOS </option>
</option></selected></td>
<td align="right"><b> CANTIDAD DE UNIDADES: </b><input name="cantidad"
 size="2" maxlenght="2" value ="1"</td><br><br><br>
</center>
<td colspan="2" align="center"> <input value="REALIZAR COMPRA" type="submit"></td>
</center>
<hr>
<table cellspacing="10">
<tbody> <tr align="center" align="middle">
<tbody> </table>
<?
include_once "pie.php";
?>
</body>
</html>
```

Al ejecutar el programa anterior **index.php** nos mostrará la siguiente pantalla:



compra.php

```
<html>
<bodybgcolor="#FFFF00">
<?
$date = date (" j / n / y ");
$cantidad = (float) $_POST ["cantidad"];
$producto=$POST["padquirido"];
$resul=0.0;
if($producto == "gorras")
{
$resul= ((10 * $cantidad) + (0.8)) * 11;
}
elseif($producto == "zapatos")
{
$resul = ((20 * $cantidad) + (1)) * 11;
}
elseif($producto == "blusas")
{
$resul = ((25 * $cantidad) + (1.5)) * 11;
}
elseif($producto == "pantalones")
{
```

continua de compra.php

```
$resul = ((30 * $cantidad) + (2)) * 11;
}
$articulo['padquirido']="$producto";
$articulo['cantidad']="$cantidad";
$articulo['resul']="$resul";
$articulo['fecha']="$date";

Include "cabecera.php";

$archivo = "compras.txt";
$apuntador = fopen($archivo)"a+");
$array1="$producto\n";
fwrite($apuntador, array1);
$array2 = "$cantidad\n";

fwrite($apuntador, array2);
$array3 = "$resul\n";
fwrite($apuntador, array3);
$array4 = "$date\n";
fwrite($apuntador, array4);
fclose ($apuntador);

echo "<center>;
echo "<h2> SU COMPRA <h2></center>"
echo "<center><table with=\"80%\"border='1'><tr>";
echo "<td> PRODUCTO</td>";
echo "<td> NÚMERO DE UNIDADES</td>";
echo "<td> COSTO TOTAL EN PESOS MEXICANOS</td>";
echo "<td> FECHA DE COMPRA</td>";
echo"</tr>";
echo"<tr>";
echo "<td>" .  $articulo['padquirido'] . "</td>";
echo "<td>" .  $articulo['cantidad'] . "</td>";
echo "<td>" .  $articulo['resul'] . "</td>";
echo "<td>" .  $articulo['fecha'] . "</td>";
echo "</tr>";
echo "</center></table>";
echo "<b><br><br><br>;
if (producto == "gorras")
{

echo "<center><img.src='gorras.jpg'><br>";
}
else if (producto == "zapatos")
{
echo "<center><img.src='zapatos.jpg'><br>";
}
else if (producto == "blusas")
{
echo "<center><img.src='blusas.jpg'><br>";
}
if (producto == "pantalones")
{
echo "<img.src='pantalones.jpg'><center><<br>";
}
echo "<br><br><br>";
echo "a href=\"index.php\">";
echo "REGRESAR";
echo "</a>";
include_once "pie.php";
?>
</body>
</html>
```

Por ejemplo si decido comprar tres blusas me mostrará la siguiente pantalla.

Fig. 3.4 Pantalla al ejecutar el programa compra.php

pie.php

```
<?
echo "<table width='80%' align='center'><tr align='center'><td>";
echo "FECHA<br>" . date("j/n/y") . "</td><td>";
echo"<B>todos los derechos reservados a: <B></td></tr></table>";
include-once"mail.php";
?>
```

Funciones para la inclusión de archivos

Include “nombre_del_archivo”

Está función incluye y evalúa un archivo externo cada vez que es interpretada.

Include_once “nombre_del_archivo”

Es igual a include pero sólo se puede incluir el archivo una vez en el script.

3.4.7 Manejo de archivos

Operaciones que se pueden realizar con los archivos:
Abrir, Leer, Escribir, Cerrar.
Además existen algunas funciones de los archivos como: buscar si existe el archivo, su tamaño o verificar si se pudo copiarlo.
Nota: Es importante que se cuente con los permisos necesarios para poder realizar la operación sobre el archivo.

3.5 Base de Datos

Una base de datos es un conjunto de información organizada y relacionada dentro de un archivo lógico, conformado por uno o más archivos físicos.
Es una buena forma de organizar y compartir información.[3]

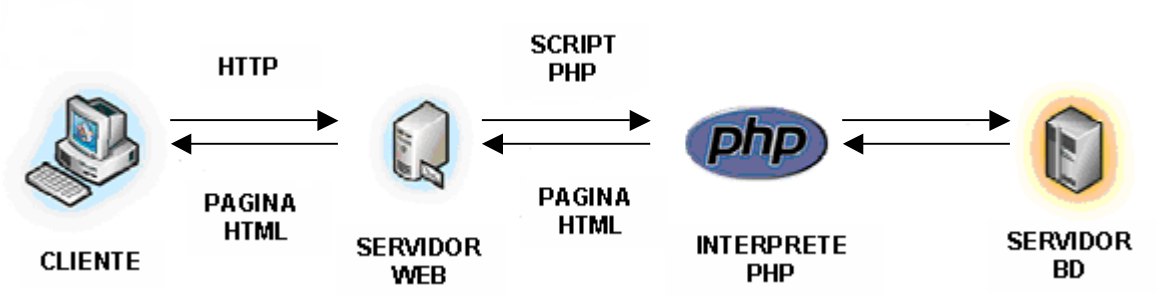


Fig. 3.5 Arquitectura Cliente/Servidor

3.6 Elementos que conforman una Base de Datos

- *Dato*. Unidad básica de almacenamiento.
- *Banco de datos. C*onjunto de datos.
- *Campo*. Componente que agrupa los datos a partir de la misma característica (columna).
- *Registro*. Componente que clasifica a los datos haciéndolos independientes unos de otros.
- *Archivo*. Elemento lógico que almacena los datos en formato tabular.
- *Información.* Conjunto de datos procesados y analizados para realizar descripciones.

3.6.1 Tipos de bases de datos

- Jerárquico
- Reticulares (de red)
- Relaciónales (MySQL)
- Orientadas a objetos

[3] Paul Du Bois, *Programación MySQL*. Anaya Multimedia, p. 169.

3.6.2 Objetivo de las bases de datos

- Disminución de la redundancia.
- Consistencia en los datos (manejar accesos concurrentes).
- Integridad de los datos (establecida a través de los datos).
- Seguridad y acceso a los datos.

3.7 Lenguajes del DBMS

DDL (Lenguaje de Definición de Datos).
Es el encargado de la definición, eliminación y alteración de objetos en las BD. Ejemplo: CREATE, ALTER y DROP.

DML (Lenguaje de manipulación de datos).
Manipulación de los objetos del tipo registro. Ejemplo: SELECT, INSERT, UPDATE, DELETE.

DCL (Lenguaje de control de datos).
Controla el acceso a las bases de datos. Ejemplos GRANT REVOKE.

3.7.1 Modelo relacional

Se basa en una representación del mundo real a través de objetos llamados entidades y sus relaciones entre ellas.
De forma gráfica se representa a través de un DER (Diagrama Entidad Relación). Ver figura 3.6.

RELACIÓN
Es la asociación que existe entre un par de entidades. Tipos de relaciones:

1:1	Uno a uno	M:1	Muchos a uno
1:M	Uno a muchos	M:M	Muchos a muchos

LLAVES
PK (Primary Key)
Es el atributo o conjunto de atributos elegidos para identificar un elemento de la entidad de forma única, puede haber llaves compuestas, si es que la llave primaria es más de un atributo.

FK (Foreign Key)
La llave foránea es la que se encarga de relacionar las entidades y está representada por la llave primaria de otra entidad, nos brinda la integridad referencial.

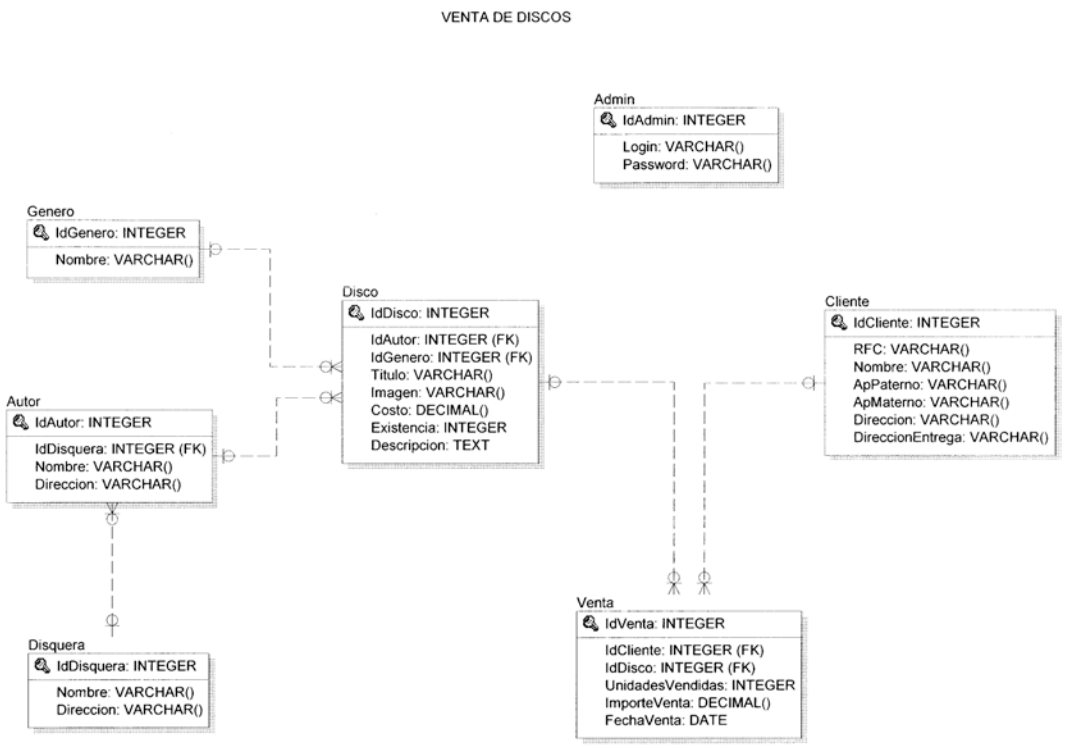


Fig. 3.6 Modelo Relacional

3.8 SQL (Lenguaje Estructurado de Consulta)

Permite comunicarnos con el sistema gestor de base de datos para realizar operaciones como crear, actualizar y manipular todos los objetos.
Está compuesto por comandos, cláusulas y funciones; además permite que estos elementos se combinen.

Para comunicarnos con MySQL utilizamos el lenguaje SQL, el lenguaje de bases de datos estándar hoy en día y las principales bases de datos lo entienden[4].

3.8.1 Instalación de MySQL

tar –zxvf mysql-4.1.4.tar.gz
cd mysql-4.1.4
./configure --prefix=/usr/local/mysql4
make
make install

[4] W. Jason Gilmore, *Beginning PHP and MySQL 5.* Second edition, Apress, 621 pp.

cp support-files/my-medium.cnf/etc/my.cnf
groupadd mysql
useradd –g mysql mysql
cd /usr/local/mysql4
/usr/local/mysql4/bin/mysqld_install_db --user=mysql
chown –R root **.**
chown –R mysql var
chgrp mysql **.**
bin/mysqld_safe –user=mysql& **--Levanta el servidor de BD**
bin/mysql
bin/mysqladmin shutdown **--Da de baja el servidor**

3.8.2 Creación de bases de datos

Crear la B.D:
CREATE DATABASE *Nombre_B.D;*

Usar la BD:
USE *Nombre_B.D;*

Mostrar como fue creada la tabla
SHOW CREATE TABLE Nombre_Tabla;

Eliminar la tabla
DROP TABLE Nombre_Tabla;

Select
Se utiliza para consultar registros de la base de datos que satisface un criterio determinado.
SELECT * FROM Nombre_Tabla;
SELECT Campo FROM Nombre_Tabla;

Comando utilizado para introducir datos en la base de Datos.
INSERT INTO Nombre_Tabla (campo1, campo2, …) VALUES (‘valor1’,’valor2’,…);

Update
Comando utilizado para actualizar uno ó más registros de la base de datos.
UPDATE Nombre_Tabla SET campo1=valor1, campo2=valor2,…[WHERE condicion];

Delete
Comando Utilizado para eliminar uno ó más registros.
DELETE FROM Nombre_Tabla WHERE condicion;

En la siguiente pantalla se mostrará como se puede consultar información en MySQL, Fig. 3.7.

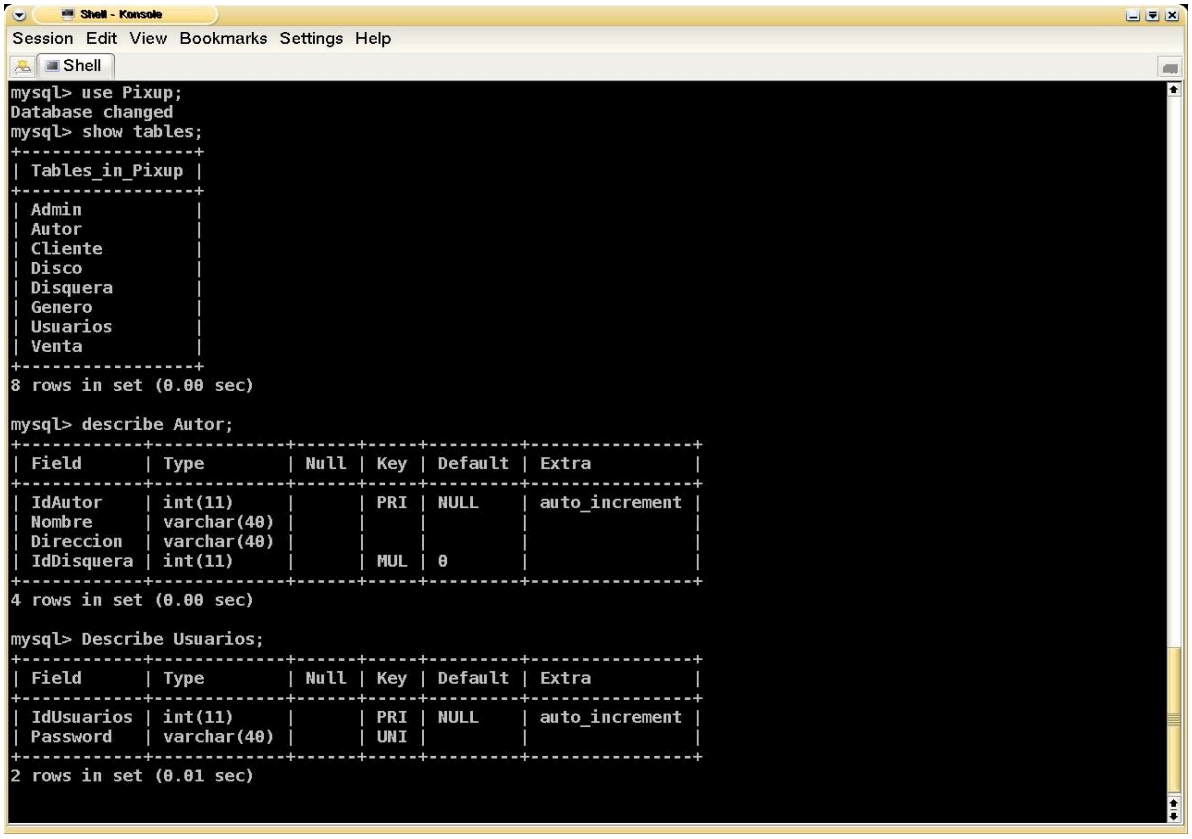


Fig. 3.7 Consultar información en MySQL

3.8.3 Respaldos y restauración de la base de datos

RESPALDAR
./mysqldump –databases Usuarios > respaldo sql
RESTAURAR
./mysql < respaldo.sql

3.8.4 Presentación de una aplicación realizada en el diplomado

Para ejecutar la aplicación es necesario levantar el servidor Apache y conectarnos con la base de datos, en este caso MySQL. (Ver página 56 y 83).

Para interactuar con MySQL por medio de programación PHP tenemos que crear un programa llamado **conexion.php**, y guardar en una carpeta llamada lib; se hace uso de él para insertar, eliminar o actualizar información.

conexion.php

```
<?
function conectarse() {
define ("host" , "local host");
define ("USUARIO" , "root");
define ("PASSWORD" , " ");
$conexion=mysql_pconnect(HOST, USUARIO, PASSWORD) or die ("Error al
conectarse" . mysql_error ());
return $conexion;
}
?>
```

Código para conectarse con la base de datos

eliminadisco.php

```
<html>
<head>
<title>ELIMINAR DISCO</title>
</head>
<body bgcolor="#6699cc">
<h2 align="center">ELIMINAR DISCO</h2><hr>
<center>
<?
require once"lib/conexion.p
$conexion=conectarse();
$bd="Pixup";
mysql_select_bd($bd$conexion) or die ("Error al seleccionar la base de datos" . mysql_error($conexion));
if($GET){
$iddisco=$GET['iddisco'];
$sqlactualiza="DELETE FROM Disco WHERE idDisco=$iddisco";
$resultado=mysql_query($sqlactualiza,$conexion) or die ("Error al eliminar los registros" . mysql_error(conexion));
}
print" <h3 align='center'> SELECCIONA EL DISCO A ELIMINAR</h3>";
$sqlconsulta="SELECT Disco.IdDisco as 'Identificador', Disco Titulo as Titulo' Disco.Costo as Costo, Autor where
Autor.IdAutor=Disco. IdAutor";
$resultado=mysql_query($sqlconsulta, $conexion) or die ("Error en la consulta" . mysql_error ($conexion));
print"<table>";
print "<tr bgcolor='3333ff'> <td> DISCO </td><td>
COSTO</td><td> EXISTENCIA </td><td> DESCRIPCIÓN </td><td> AUTOR </td></tr>";
$contador=0;
while($fila=mysql_fetch_array ($resultado)) {
$color="#99CCFF";
if($contador%2==0)
$color="#CC99FF";
print "<tr bgcolor='$color'>";
print "<td> <a href='" . $_SERVER[PHP_SELF'] . "? iddisco=" . $fila[0] . "'>" . $fila[1] . "</a>
</td> . <td>$fila[2]</td> . <td>$fila[3]</td> . <td>$fila[4]</td> . <td>$fila[5]</td>";
print "</tr>";
$contador ++;
}
print "</table>";
?>
</center>
</body>
</html>
```

Código para conectarse con la base de datos

Código para eliminar un disco

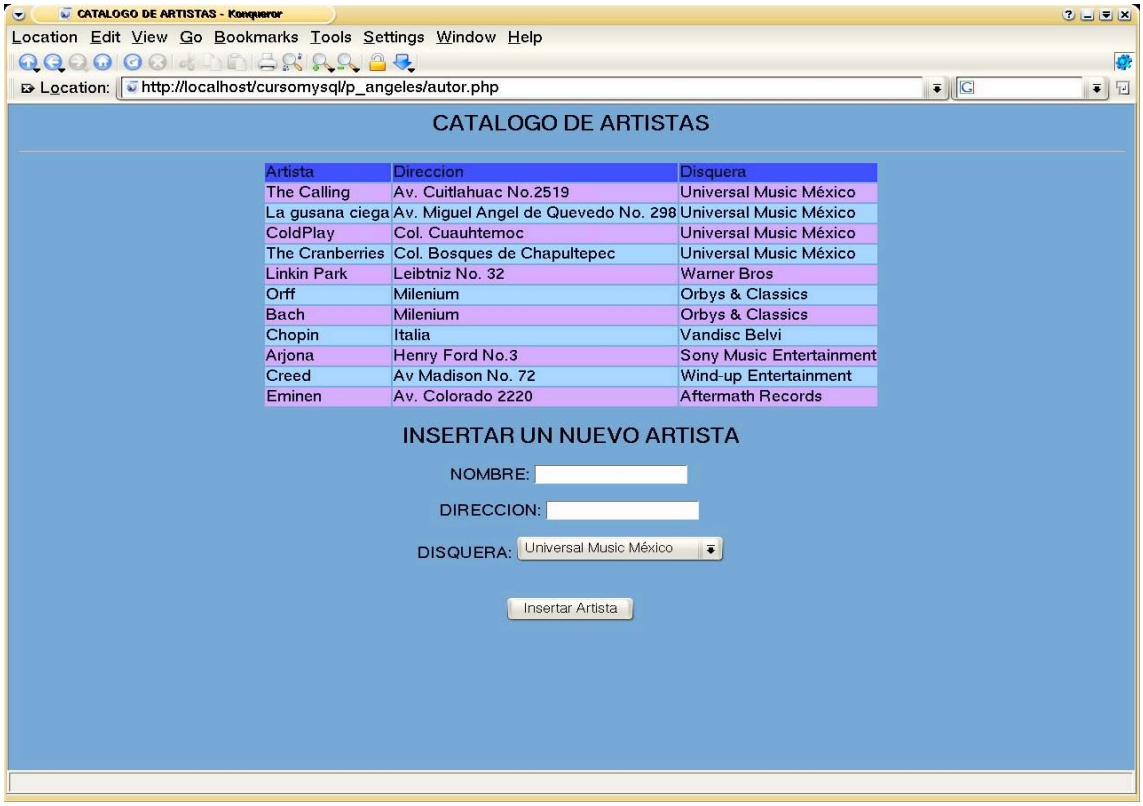


Fig. 3.8 Pantalla al ejecutar el código autor.php

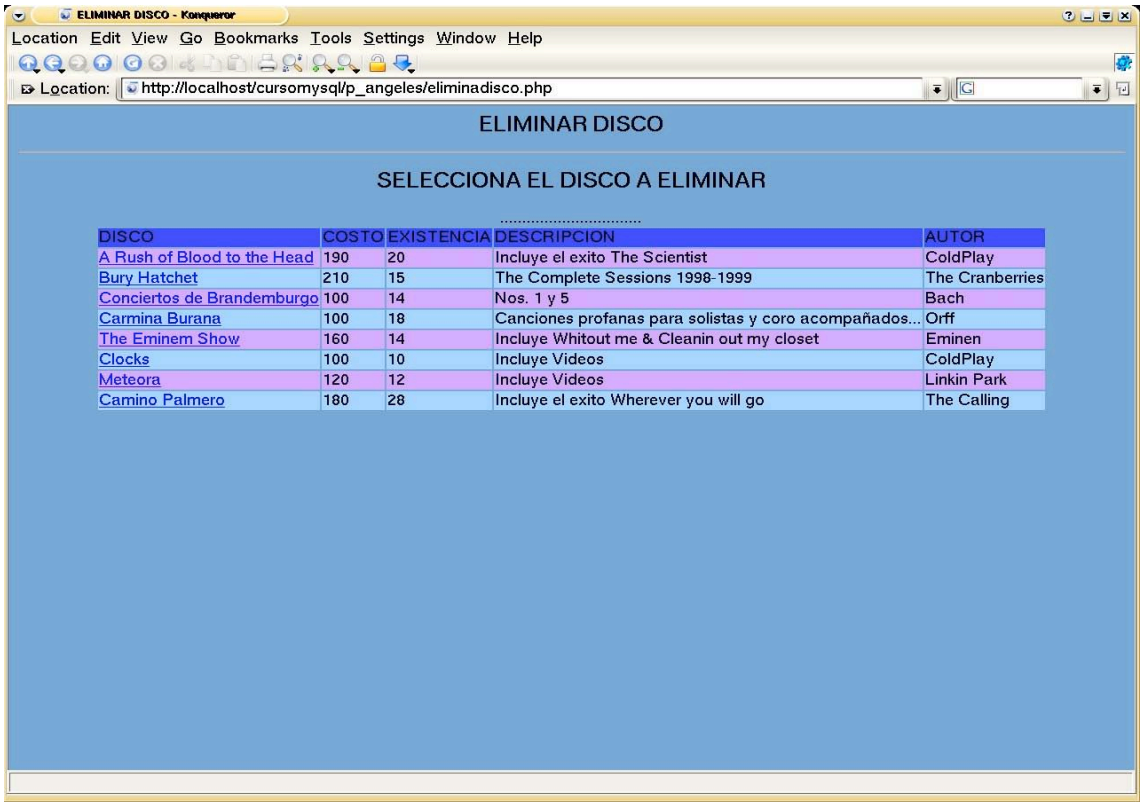


Fig. 3.9 Pantalla al ejecutar el código eliminadisco.php

CAPÍTULO 4

INTRODUCCIÓN A LA SEGURIDAD EN CÓMPUTO

OBJETIVOS ESPECÍFICOS

Explicar:

- La importancia de la seguridad y los elementos que permiten proteger el sistema y el flujo de información.
- Algunos conceptos relacionados con la seguridad informática.
- En qué consiste un análisis de riesgo.
- Qué es un plan de contingencia y qué sucedería si no se cuenta con éste.
- Los vectores y mecanismos de seguridad.
- El concepto del análisis de riesgo y qué información se obtiene a partir de éste.
- Proponer una política de seguridad aplicada al CONACYT (basada en el RFC 2196).

JUSTIFICACIÓN

La seguridad trata de preservar la confidencialidad, integridad y disponibilidad de la información o de un sistema, así mismo busca proteger de distintas amenazas que puedan surgir de entornos hostiles o frente a situaciones imprevistas. La seguridad en los sistemas de información y de cómputo se ha convertido en una de las preocupaciones más grandes.

Dada la potencialidad de esta herramienta y de sus innumerables aplicaciones, cada vez más personas y más empresas sienten la necesidad de contar con sistemas adecuados que protejan su información y sus recursos.

La seguridad en el ámbito telemático es muy importante, abarcando desde la puesta en marcha de una red privada hasta la integridad de la información que se envía por correo electrónico. Es innegable que sería imposible alcanzar un 100% de seguridad ya que los programas son escritos por humanos, y por tanto, son susceptibles a tener todo tipo de errores. Sin embargo, cada día surgen más elementos para combatir cualquier anomalía.

La complejidad de los programas (aún los más simples) impide que el programador tenga en mente todos los factores. La gran cantidad de interacciones entre los elementos de un programa y el sistema, así como cualquier cambio en éstos, puede tener consecuencias inesperadas si no se hace con cuidado y conciencia.

Existen técnicas como la criptografía, mecanismos y vectores de seguridad, análisis de riesgos, planes de contingencias, políticas de seguridad, entre otras; que sirven para preservar la confidencialidad, integridad y disponibilidad de la información o de un sistema.

4. INTRODUCCIÓN A LA SEGURIDAD EN CÓMPUTO

4.1 Seguridad en cómputo

Es un conjunto de mecanismos aplicados a las Tecnologías de Información con el único fin de preservar la confidencialidad, integridad y disponibilidad de la información o de un sistema.
Ejemplos: Seguridad en redes, en sistemas y en aplicaciones.

4.2 Algunas definiciones referentes a la seguridad en cómputo

Vulnerabilidad
Es la ausencia de una contramedida o debilidad de la misma, que permite que sus propiedades de sistema sean violadas.
La debilidad puede originarse en el diseño, la implementación o los procedimientos para operar o administrar el sistema.

Riesgo
Potencial para pérdida o falla, como respuesta a las siguientes preguntas:
- ¿Qué podría pasar? (o cuál es la amenaza)
- ¿Qué tan malo puede ser? (impacto o consecuencia)
- ¿Qué tan frecuente puede ocurrir? (frecuencia)
- ¿Qué tanta certidumbre se tiene en las tres primeras respuestas? (grado de confianza)

Exploit
Forma de explotar una vulnerabilidad. Generalmente en forma de programa/ software que realiza en forma automática un ataque.

Ataque
Acción o acciones que tienen por objetivo que cualquier parte de un sistema de información automatizada deje de funcionar, de acuerdo con su propósito definido o bien, causa destrucción, modificación o retraso. Existen ataques pasivos y activos.

Firewall
Software que sirve para bloquear el tráfico indeseable; autentica o verifica los inicios de sesión de las personas que intentan acceder a un sistema imponiendo una política de seguridad entre la organización de red privada y el Internet, de acuerdo a las necesidades puede bloquear mensajes y usuarios que no se deseen en el sistema.

Monitorea el sistema, en caso de que suceda un problema en el tránsito de los datos éste genera una alarma. Los firewalls son diseñados para evitar tráfico no deseado o no autorizado en una red. “El propósito de un firewall es proveer seguridad a sus componentes, además controlar lo que está permitido mediante ciertas restricciones”[1]. En la figura 4.1 podemos observar la representación de un firewall.

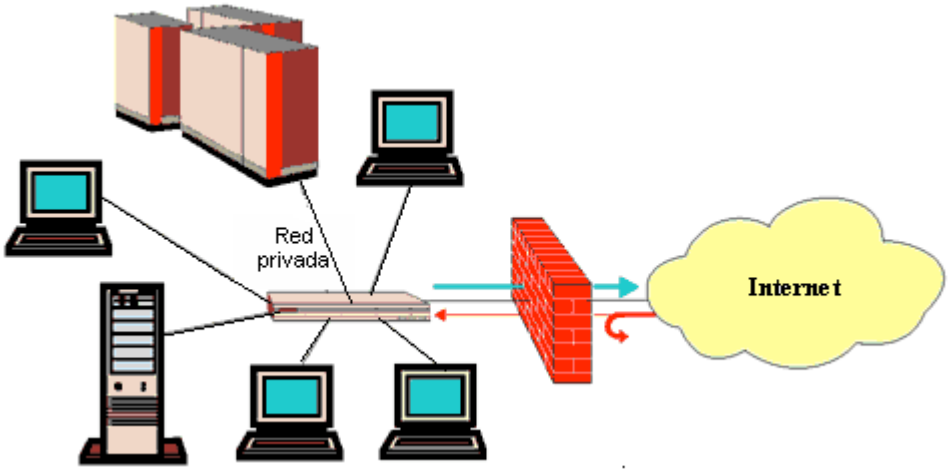


Fig. 4.1 Representación de un Firewall

4.3 Vectores de seguridad

Confidencialidad
Servicio que garantiza la privacidad de los datos. Un sistema posee está propiedad si la información no es disponible ni puesta en descubierto para usuarios, entidades o procesos no autorizados. Principal mecanismo para implementar este servicio es la criptología[2].

Integridad
Permite proteger los datos contra ataques activos. El sistema posee está propiedad si los datos que son manipulados por éste no son alterados o destruidos. Uno de los mecanismos principales son las huellas digitales.

Disponibilidad
Si la información es accesible, está disponible en el momento que lo deseen los usuarios, entidades o procesos autorizados.

---

[1] Marcus Gonçalves, *Firewalls a complete guide*. Mc Graw – Hill, p.41.

[2] Criptología.- Ciencia que estudia los aspectos y contenidos de información en condiciones de secrecía.

No repudiación
Permite comprobar mediante pruebas, las acciones realizadas por el origen o destino de los datos.
Los mecanismos principales: certificado y las firmas digitales.

4.4 Análisis de riesgo

Es el proceso de identificación y evaluación del riesgo a sufrir un ataque y perder datos, tiempo y horas de trabajo.
Su análisis no sólo nos lleva a establecer un nivel adecuado de seguridad, sino que permite conocer a fondo el sistema que vamos a proteger.

Información que se obtiene del análisis de riesgo:

- Recursos sensibles de la organización
- Amenazas del sistema
- Vulnerabilidades especificas del sistema
- Probabilidad de ocurrencia de una pérdida
- Contramedidas efectivas
- Herramientas de seguridad

Resultado:
- Implementación de un sistema de seguridad eficiente en costes y tiempo.

4.5 Mecanismos de seguridad

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

Prevención
Aumentan la seguridad de un sistema durante su funcionamiento normal.
Ejemplos: Cifrado durante la transmisión de datos, passwords difíciles, firewalls, sistemas biométricos.

- Mecanismos de autenticación

Propiedad para reconocer que la información recibida es la misma que la enviada, y que el que dice ser que la envió, realmente lo hizo.

Basados en algo que se sabe:
Passwords, frases y números de identificación personal como un nip.
Basados en algo que es:
Se realiza una medición física y se compara con un perfil almacenado con anterioridad. Biométricas y comportamiento.

Basados en algo que se tiene:
Usan un objeto que llevan consigo y que de alguna forma comprueba la identidad del portador. Tarjetas inteligentes.

- Mecanismos de control de acceso

Usados para proteger objetos del sistema (archivos, recursos, etc.)
La autenticación permite establecer quien eres.
La autorización (o control de acceso) establece que puedes hacer con el sistema.

Modelos:

Control de Acceso Discrecional (DAC)
Un usuario bien identificado (típicamente el propietario o creador del recurso) decide cómo protegerlo estableciendo cómo compartirlo, mediante controles impuestos por el sistema.

Control de Acceso Mandatario (MAC)
Es el sistema el cual protege los recursos.
Todo recurso del sistema y todo usuario tienen una etiqueta de seguridad.

- Mecanismos de separación

Mecanismos que sirven para delimitar una frontera.
Ejemplos: switch, filtros de paquetes, ruteadores y proxies.

Detección
Buscan descubrir incidentes (vulnerabilidades) al momento que ocurren o lo antes posible y también reducir el daño. Permiten identificar y detectar culpables. Ejemplos: I DS[3], Osiris[4], Proxies[5], etc.

Recuperación
Se pone en marcha después de un incidente de seguridad, se aplican cuando se detecta una violación del sistema para retornarlo a su estado normal.
Ejemplos: Respaldos, bitácoras, BCP (Capacidad para mantener la continuidad de las operaciones), dirigido a situaciones rutinarias no catastróficas. DRP (Planeación de la recuperación de desastres).

4.5.1 Criptología

Ciencia que estudia los aspectos y contenidos de información en condiciones de secrecía.
Del griego: *criptos* oculto y *logos* tratado.

---

[3] IDS.- Sistema de Detección de Intrusos, programa para detectar accesos desautorizados a un computador o una red.
[4] Osiris. - Herramienta para el monitoreo de la integridad de un sistema. El servidor se conecta con el cliente para monitorear periódicamente la integridad.
[5] Proxy.- Programa o dispositivo que efectúa una acción en representación de otro, (acceso a Internet en lugar de otro ordenador).

Se divide en:

- Criptografía
- Criptoanálisis

Criptografía
Conjunto de técnicas o procedimientos que alteran los símbolos de información sin modificar el contenido; para las partes que no disponen de las técnicas de información modificada es un conjunto de símbolos sin contenido.
Es el arte de construir códigos secretos.

Criptoanálisis
La metodología y técnicas que permiten recuperar la información que ha sido previamente tratada por un procedimiento criptográfico, sin conocer la técnica utilizada para la criptografía.

Cifrado
Algoritmo criptográfico para ocultar el mensaje en “texto plano”.
Función matemática usada para la encripción y decodificación del mensaje original.

Criptosistemas
Conjunto de procedimientos que garantizan la seguridad de la información y utilizan técnicas criptográficas. Su elemento fundamental es la “llave”, donde en algunas referencias a ésta se le conoce como clave.

4.6 Planes de contingencia

Es un conjunto de datos estratégicos de la empresa y que se plasma en un documento con el fin de protegerse ante eventualidades.

Consiste en un análisis pormenorizado de las áreas que componen una organización para establecer una política de recuperación ante un desastre.
Además de aumentar su seguridad, la empresa también gana en el conocimiento de fortalezas y debilidades.
Si no lo hace, se expone a sufrir una perdida irreparable mucho más costosa que la implantación de este plan.

4.6.1 Pérdidas por no contar con un plan de contingencia

- Clientes
- Imagen
- Ingresos por beneficios
- Ingresos por ventas y cobros

- Ingresos por producción
- Competitividad
- Credibilidad en el sector

4.7 Políticas de seguridad

Diseño de una política de seguridad.- Definir responsabilidades y reglas a seguir para evitar amenazas o minimizar sus efectos en caso de que se produzcan.

Conjunto de reglas formales que deben ser respetadas por la organización que tiene acceso a los sistemas de información; las cuales garantizan la seguridad del sistema.

4.7.1 RFC 1296

Guía para una correcta implementación de las políticas de seguridad en los Sistemas de Información.

4.7.2 Implementación de una política de seguridad aplicada al CONACYT, (basada en el RFC 2196).

INTRODUCCIÓN

Una de las razones más importantes para crear una política de seguridad en una empresa es asegurar que los esfuerzos gastados dejen beneficios desde el punto de vista resultados/inversión (sea monetario, temporal, etc.).

El análisis de riesgo incluye la determinación de lo que se requiere proteger y cómo protegerlo. Es el proceso de examinar todos los riesgos, para después tomarlos por niveles de severidad. Este proceso incluye tomar decisiones costo-efectivas sobre lo que se quiere proteger.

Se deben analizar dos puntos muy importantes:

- ¿Qué recursos estamos tratando de proteger?
  Identificar todas las cosas que necesitan estar protegidas (donde el acierto básico de la seguridad es la Disponibilidad, Confidencialidad e Integridad).

- ¿Contra qué deben de ser protegidos los sistemas?
  Aquí deberá examinarse cómo puede ser afectada cada área, es decir, cuáles son las cosas que la amenazan.

Una política de seguridad es el establecimiento formal de reglas por las cuales a la gente se le da acceso a tecnología de una organización y a los beneficios informáticos.
La principal consideración de una política de seguridad es informar a los usuarios, el staff de operación y a los manager de los requerimientos obligatorios para la protección de las ventajas informáticas y la información tecnológica. La política debe especificar los mecanismos a través de los cuales estos requerimientos se pueden cumplir.

Otro objetivo es proveer una línea de base para obtener, configurar y auditar redes de computadoras para cumplir con las políticas.

Involucrados al formar la política

Todo el personal de una empresa debe estar envuelto en la creación y revisión de los documentos de la política de seguridad:

- Administrador de la seguridad del sitio
- Staff técnico de la información técnica
- Administradores de los grandes grupos de la organización
- Equipo de respuesta ante un incidente de seguridad
- Representante de los grupos de usuarios afectados por la política de seguridad
- Manager responsable
- Asesor legal (en algunos casos)

PLANTEAMIENTO DEL PROBLEMA

Se desea implementar una política de seguridad en el Consejo Nacional de Ciencia y Tecnología (CONACYT), debido a que en dicha institución todavía existen algunas carencias en este ámbito. Esta situación es menos frecuente en empresas privadas, donde se cuenta con más recursos para obtener una mayor seguridad de las Tecnologías de la Información.

Con base en el análisis se han identificado las problemáticas derivadas de una inadecuada política de seguridad:

- Mala distribución de los módulos de trabajo
- Inadecuado ambiente de trabajo (clima, falta de ventilación, etc.)
- Cableado mal colocado
- Utilización de material de diseño en áreas no apropiadas
- Inadecuadas salidas de emergencia
- Fallas en el conmutador y en la planta de luz
- La caída frecuente del sistema y/o tráfico en la red
- Falla de equipos (debido a virus)

- Pérdida de autenticación (por descuido o demasiada confianza)
- Robo de equipo

Se han mencionado algunas problemáticas que no se pueden solucionar de inmediato, debido a que no se cuenta con gran apoyo económico. No quiero decir que sea insuficiente, pero existe un mal manejo de éste, además se debe empezar poco a poco para lograr resultados satisfactorios y así lograr que se acepte el resto de las propuestas.

Por tal motivo expondré, según mi criterio, los puntos más importantes a cubrir. Tomé en cuenta que con algunos ya cumple la institución, sin embargo existen vulnerabilidades.

También me interesa que estos lineamientos queden por escrito para cuando ingrese una persona nueva al ámbito laboral conozca las políticas que se deben de cumplir. Asimismo, pretendo se haga un análisis y se decida con cuáles aspectos iniciar y cómo manejarlos de acuerdo a su jerarquía de importancia. Hablaré un poco en términos generales, pero más que nada enfocado al centro de cómputo, debido a que es el área que me corresponde de acuerdo a mi carrera, y por consiguiente, donde tengo un mejor conocimiento a diferencia de otras. Cabe mencionar que las mismas políticas servirán para toda la institución, pero en algunos casos no se presentarán tan estrictamente.

BASES DE LA POLÍTICA DE SEGURIDAD DE ACUERDO AL RFC 2196

Las políticas de seguridad tienen por objeto establecer las medidas de índole técnica y de organización necesarias para garantizar la seguridad de las Tecnologías de la Información y personas que interactúan haciendo uso de servicios asociados a ellos.

A continuación las propuestas:

ADMINISTRACIÓN

Administración General

- ✓ Estructura organizacional del centro de cómputo
- ✓ Programas de capacitación
- ✓ Políticas de contratación
- ✓ Inventario y actualización de equipo y software
- ✓ Evaluación de las necesidades del centro de cómputo
- ✓ Existencia de manuales de uso y administración de los bienes informáticos
- ✓ Actualización de manuales
- ✓ Reglas internas de operación

Forma de Trabajo

- ✓ Políticas y procedimiento de control para captura y proceso de sistemas
- ✓ Bitácoras de procesos, reproceso y fallas
- ✓ Administración de horarios del personal que hace uso de equipos
- ✓ Calendarios de producción
- ✓ Estadísticas de uso de equipos

Plan de Contingencias

- ✓ Plan de emergencia
- ✓ Plan de respaldo
- ✓ Plan de recuperación
- ✓ Programas de registros vitales
- ✓ Actualización de planes de contingencias

SEGURIDAD DEL PERSONAL

- ✓ Control de acceso físico
- ✓ Control de acceso en horas y días no hábiles
- ✓ Registro de visitantes
- ✓ Inspección de paquetes, portafolios, que entran y/o salen
- ✓ Control de acceso de personal de servicio
- ✓ Sistema de gafetes de visitantes

Ubicación del centro de cómputo

- ✓ Ubicación de fácil acceso
- ✓ Separación del centro de cómputo de otras áreas
- ✓ Puertas de carga y descarga
- ✓ Salida de emergencia
- ✓ Señalización de las salidas de emergencia
- ✓ Documentación y actualización de problemáticas referentes a aspectos físicos

Consideraciones de la construcción del edificio

- ✓ Restricción del acceso a interruptores de energía del centro de cómputo
- ✓ Cuidado del cableado suelto o mal colocado del equipo de cómputo e instalaciones
- ✓ Inspección de las condiciones de las instalaciones eléctricas
- ✓ Existencia de no-break y planta de energía
- ✓ Adecuado equipo de aire acondicionado
- ✓ Evitar grandes ventanales

Soporte ambiental

✓ Capacitación del personal en caso de siniestros (se debe extender, ya que existe pero no hay cupo para todos)
✓ Actualización de manuales de contingencias
✓ Revisión de instalaciones
✓ Revisión e implantación de las recomendaciones de seguridad

Riesgo y protección contra fuego

✓ Existencia de extintores
✓ Comprobación del tipo de carga de extintores y su fecha de recarga
✓ Alarmas y detección contra incendios
✓ Existencia de material combustible dentro del centro de cómputo
✓ Existencia de suficientes extintores
✓ Capacitación del personal en el uso de extintores para caso de siniestros
✓ Señalización de equipo contra incendios y salidas de emergencia

Protección de riesgos

✓ Administración de utilerías no autorizadas en disco duro
✓ Control de la información respaldada
✓ Respaldo de las utilerías de los sistemas en producción
✓ Apego a las políticas de respaldo de la información
✓ Almacenamiento de los respaldos de los archivos en otro edificio
✓ Destrucción de los listados y papel carbón de información que es obsoleta

SEGURIDAD LÓGICA

Administración y uso de claves de acceso

✓ Responsable de asignar claves de acceso
✓ Confidencialidad de las claves de acceso
✓ Jerarquización de las claves de acceso
✓ Bitácora de claves de acceso
✓ Control de claves asignadas
✓ Definición formal de procedimientos y criterios de análisis para la asignación de claves
✓ Correcta asignación de claves de acceso al personal, así como el cambio periódico de éstas

Protección de información

✓ Validación de programas y de detección de intrusos en las librerías de producción
✓ Protección de acceso a archivos de sistemas y librerías de producción
✓ Existencia de copias mensuales del archivo histórico

- ✓ Realización de operaciones de procesamiento de datos en presencia del personal ajeno del centro de cómputo
- ✓ Almacenamiento adecuado de papelería oficial
- ✓ Monitoreo y reportes de operaciones inválidas

Controles generales y operaciones

- ✓ Optimizar procedimientos para la detección y análisis de debilidades y vulnerabilidad de cada área
- ✓ Existencia de asignación y control de equipo
- ✓ Existencia de políticas y procedimientos de cada área
- ✓ Existencia de un programa de mantenimiento del equipo

Complemento de las propuestas anteriormente mencionadas

- ✓ Asignación de una persona directamente responsable que dentro de su área debe tener un control absoluto (debe haber más de uno para cubrir emergencias y debe de existir en donde se maneja información que se considere importante para la institución).
  Nota: Cultivar su fidelidad mediante recompensas como penalizaciones adecuadas para así evitar el chantaje y soborno, y nunca se debe olvidar el factor humano.
- ✓ Concienciar a las personas de lo que implica ser responsable de la seguridad de las Tecnologías de la Información.
- ✓ Asignación de jerarquías para el acceso a la información, asumiendo responsabilidad y compromisos de lo que implica el puesto.
- ✓ Existencia de configuración de firewalls (tanto internos como externos y modificar la configuración por defecto, ya que podría ser fácilmente vulnerable).
- ✓ Establecer auditorias de seguridad, las cuales deben ser imprevistas y al azar, electrónicas y también físicas, de forma cautelosa y evidente.
- ✓ Monitoreo de redes de datos y telecomunicaciones.
- ✓ Distribuir software antivirus:
  - Instalación de antivirus
  - Rastreo automático mediante la solución antivirus para detectar y eliminar virus
  - En caso de contingencia con virus, cuando el antivirus no haya detectado automáticamente, se debe llamar al área de soporte.
- ✓ Prohibir uso de programas que intenten adivinar contraseñas alojadas en tablas de contraseñas.
- ✓ Dar asesoría siempre y cuando no entorpezca las acciones de mayor relevancia.

CONCLUSIONES

El software libre ha logrado evolucionar y adaptarse para competir con el modelo propietario; sin embargo, no ha podido desplazarlo. Aún así, ésta es una buena opción a considerar, porque nos ayudará a expandir nuestra visión con respecto al desarrollo e implementación de sistemas, de una manera muy sencilla, especialmente en la obtención de herramientas para dicho desarrollo.

Linux no sólo permite realizar el trabajo diario y explorar la Web, sino que además podremos aprender a escribir consultas de bases de datos, administrar un servidor Web, acceder a servicios Web y participar en las miles de actividades proporcionadas en la informática actual.

Hoy en día son muchos los sitios de Internet que ejecutan Linux para gestionar grandes servidores Web, aplicaciones de comercio electrónico, motores de búsqueda, entre otros.

La mayoría de los sitios que tienen información importante para proporcionar, utilizan programas en la parte del servidor, donde podemos hacer uso de diversos tipos de éstos y con variados paquetes de software. Una de estas combinaciones se ha convertido en la más difundida para la implantación de estas técnicas, incluso ha recibido el acrónimo de LAMP, (que se refiere a Linux, Apache, MySQL y PHP); misma que fue utilizada en el diplomado “Desarrollo e Implementación de Sistemas con Software Libre”.

En el transcurso de este diplomado obtuve conocimientos con respecto a Linux, Html, Apache, PHP y MySQL, logrando ampliar mi visión del desarrollo de sistemas que se encuentran en línea como en Internet o en cualquier otro tipo de red.

Considero que la modalidad de seminarios de actualización y capacitación profesional es una interesante opción de titulación ya que nos permite obtener conocimientos, tanto teóricos como prácticos, los cuales son necesarios para adaptarnos al mundo tan cambiante en el que vivimos.

GLOSARIO

Análisis de riesgo.- Es el proceso de identificación y evaluación del riesgo a sufrir un ataque y perder datos, tiempos y horas de trabajo.

Archivo.- Elemento lógico que almacena los datos en formato tabular.

Arreglos.- Estructuras que permiten el almacenamiento de un conjunto de datos bajo un mismo nombre.

Ataque.- Acción o acciones que tienen por objetivo que cualquier parte de un sistema de información automatizada deje de funcionar, de acuerdo con su propósito definido o bien, causa destrucción, modificación o retraso.

Banco de datos.- *C*onjunto de datos.

Base de datos.- Es un conjunto de información organizada y relacionada dentro de un archivo lógico, conformado por uno o más archivos físicos.

Campo.- Componente que agrupa los datos a partir de la misma característica (columna).

CGI.- es una norma para establecer comunicación entre un servidor Web y un programa, de tal modo que este último pueda interactuar con Internet.

Cifrado.- Algoritmo criptográfico para ocultar el mensaje en “texto plano”.

Compilador.- Es un programa que traduce el lenguaje de alto nivel a lenguaje máquina de una computadora.

Criptoanálisis.- La metodología y técnicas que permiten recuperar la información que ha sido previamente tratada por un procedimiento criptográfico, sin conocer la técnica utilizada para la criptografía.

Criptografía.- Conjunto de técnicas o procedimientos que alteran los símbolos de información sin modificar el contenido.

Criptología.- Ciencia que estudia los aspectos y contenidos de información en condiciones de secrecía.

Criptosistemas.- Conjunto de procedimientos que garantizan la seguridad de la información y utilizan técnicas criptográficas. Su elemento fundamental es la “llave”, donde en algunas referencias a ésta se le conoce como clave.

Dato.- Unidad básica de almacenamiento.

Exploit.- Forma de explotar una vulnerabilidad. Generalmente en forma de programa/ software que realiza en forma automática un ataque.

Firewall.- Software que se instala en una computadora para bloquear el tráfico indeseable, impone una política de seguridad entre la organización de red privada y el Internet de acuerdo a las necesidades.

FTP.- Es un programa especialmente diseñado para la transferencia de archivos a través de Internet.

Gateway.- Puerta de enlace, además de encaminar la información es capaz de convertir los datos de un protocolo a otro.

GNU.- Licencia Pública General creada por la Fundación de software libre, orientada principalmente a proteger a la libre distribución, modificación y uso del software.

HTML.- Es un lenguaje de programación que se utiliza para la creación de páginas en la WWW.

HTTP.- Es el protocolo (simple y poderoso) de red para WWW; el cual basa su operación en la arquitectura "Cliente – Servidor".

IDS.- Sistema de Detección de Intrusos, programa para detectar accesos no autorizados a un computador o una red.

Información.- Conjunto de datos procesados y analizados para realizar descripciones.

Iintérprete.- Es un traductor de lenguajes de programación de alto nivel, el cual realiza el proceso de compilación paso a paso.

Linux.- Es un sistema operativo, tipo Unix, es libre y viene acompañado del código fuente

Minix.- Sistema operativo tutorial, escrito por Andrew Tanebaum.

Proceso.- Un proceso es un programa que se encuentra corriendo dentro del servidor.

Protocolo.- Definen las normas que posibilitan que se establezca una comunicación entre varios equipos y dispositivos.

Red.- Es un sistema de comunicaciones de datos que permite conectar computadoras y periféricos para así compartir información y recursos.

Registro.- Componente que clasifica a los datos haciéndolos independientes unos de otros.

Riesgo.- Potencial para pérdida o falla.

Ruteadores.- Sistemas que se encargan de dirigir la información por el camino adecuado entre varias redes.

Osiris. - Herramienta para el monitoreo de la integridad de un sistema. El servidor se conecta con el cliente para monitorear periódicamente la integridad.

Planes de contingencia.- Es un conjunto de datos estratégicos de la empresa y que se plasma en un documento con el fin de protegerse ante eventualidades.

Política de seguridad.- Conjunto de reglas formales que deben ser respetadas por la organización que tiene acceso a los sistemas de información; las cuales garantizan la seguridad del sistema.

Proxy.- Programa o dispositivo que efectúa una acción en representación de otro, (acceso a Internet en lugar de otro ordenador).

Seguridad en cómputo.- Es un conjunto de mecanismos aplicados a las Tecnologías de Información con el único fin de preservar la confidencialidad, integridad y disponibilidad de la información o de un sistema.

Servidor.- Es un programa encargado de ofrecer comunicación mediante el protocolo HTTP y servicios dentro del World Wide Web en Internet.

Shell.- Programa diseñado para aceptar comandos y ejecutarlos.

SQL.- (Lenguaje Estructurado de Consulta)**,** Está compuesto por comandos, cláusulas y funciones; estos elementos se combinan para crear, actualizar y manipular todos los objetos en la BD.

Topología de red.- Es la forma física o la estructura de interconexión entre los distintos equipos (dispositivos de comunicación y computadoras) de la red.

Variable.- Es el nombre que se le da a una posición de la memoria en la cual se guarda la información.

Vulnerabilidad.- Es la ausencia de una contramedida o debilidad de la misma, que permite que sus propiedades de sistema sean violadas.

BIBLIOGRAFÍA

- Cisco Systems, Inc, *Academia de Networking de Cisco Systems*, segunda Ed. Pearson education.
- Du Bois, Paul, *Programación MySQL*. Anaya Multimedia.
- Gilmore, W. Jason, *Beginning PHP and* MySQL 5, Apress.
- Gonçalves, Marcus, *Firewalls a complete guide.* Mc Graw - Hill. 655pp.
- Greenspan, Jay and Brad Bulger, *MySQL/PHP Database Application*, M & T Books, 602 pp.
- Gutiérrez Gallardo, Juan Diego, *Desarrollo Web con PHP5 y MySQL*. Guía practica para usuarios. Anaya Multimedia.
- Kalle Dalheimer, Matthias y Matt Welsh, *Guía de referencia y aprendizaje Linux*, 2a Ed. Anaya Multimedia.
- Petersen, Richard, *The complete reference Linux.* Third edition, Osorne, Mc Graw Hill.
- Raya, José Luís y Cristina Raya, *Redes locales*, Ra – Ma.
- Schroder, *Curso de Linux, O´reilly*, Anaya multimedia, 7003 pp., 2005.
- http://en.wikipedia.org/wiki/LAMP_(software_bundle)
- http://es.wikipedia.org/wiki/Seguridad_inform%C3%A1tica
- http://es.wikipedia.org/wiki/Software_libre
- http://gias720.dis.ulpgc.es/Gias/Cursos/Tutorial_html/indice.htm
- http://linux.com/
- http://www.php.net/
- http://sunsite.unam.mx/archivos/linux/manual.doc
- http://www.unixmexico.org/modules.php?name=News&file=article&sid=666