UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN

“INFORME DE UN SISTEMA DE FACTURACIÓN EN LÍNEA IMPLEMENTADO CON SOFTWARE LIBRE”

TRABAJO POR ESCRITO

EN LA MODALIDAD DE SEMINARIOS Y CURSOS DE ACTUALIZACIÓN

P R O F E S I O N A L

QUE PARA OBTENER EL TÍTULO DE :

INGENIERO EN COMPUTACIÓN

P R E S E N T A :

LILIA HERI REYES HERNÁNDEZ

DIRIGE: ING. SILVIA VEGA MUYTOY

SAN JUAN DE ARAGÓN, ESTADO DE MÉXICO **2007**

A Dios

Porque en ti encontré la luz y fuerza para poder culminar una de mis metas en mi vida.

Gracias por tus bendiciones

A mis Padres, José y Madai

Por darme la vida, su amor y comprensión, así mismo guiarme y apoyarme en los momentos más difíciles.

Papá tu eres mi fuerza, mi ejemplo a seguir; me dejaste sin decirme adiós, pero se que desde el cielo me seguirás guiando y no permitirás que desmaye, siempre te llevo en mi corazón.

Mamá te admiro, eres mi amiga, me comprendes y me das consejos, no dejaste que desmayara y así logre mi meta con tu apoyo incondicional. Tenerte como mi mamá es una gran bendición.

Son únicos, los Respeto, Admiro y Amo

A mis hermanos, Nubia, Gamaliel y Dacia

Por su comprensión y apoyo en esta parte de mi vida.

Arian y **Alondra** han sido un gran motivo para poder llegar hasta aquí.

Los Quiero Mucho

A Pedro

Quien es una persona muy especial para mí y es parte importante de mi vida. Ha estado a mi lado en todo momento fuesen alegrías o tristezas. Me cuida, apoya, comprende, aconseja y no permite que me quede en el camino; gracias por tu amor.

Así mismo agradecer a tu familia (Pedro, Alejandra, Cinthia y Dalia), por soportarme, escucharme y compartir su hogar con migo.

Gracias por existir Amor

A Brenda

Fue, es y será quien me aliente, comprenda y apoye en momentos buenos y malos.

Eres mi amiga incondicional

A mis Tíos y Primos

Ustedes me brindaron su apoyo moral y su comprensión, así mismo han estado a mi lado en momentos difíciles.

Son parte de mí

A mis Amigos y Compañeros

Moni, Erika, Ruth, Pedro, Mario, Eduardo, Felipe, Samanta, Janeth, Mardol y todos aquellos con los cuales conviví durante la carrera; los cuales me brindaron su apoyo.

A todos Mil Gracias…

INDICE

OBJETIVO GENERAL

Conocer herramientas administrativas que permitan desarrollar e implementar sistemas para el control de procesos de información, que funcionen de forma natural en red o por Internet, empleando herramientas de software libre que han demostrado tener una alta confiabilidad, alto desempeño y funcionalidad.

OBJETIVOS ESPECÍFICOS

- Conocer el sistema operativo Linux, sus requerimientos de instalación, comandos y utilerías básicas.
- Instalar el sistema operativo Linux en PCs y configurar la tarjeta de red, video y particiones del disco duro, así mismo conocer el uso eficiente de los recursos y tener un control mas preciso de los mismos.
- Elaborar paginas WWW, apoyado con HTML.
- Identificar el procedimiento para instalar, configurar y administrar un servidor WWW en un servidor de plataforma Linux.
- Crear aplicaciones dinámicas e interactivas para el Web utilizando el lenguaje PHP.
- Desarrollar una aplicación de base de datos que funcione a través del WWW, empleando herramientas de software libre.
- Reconocer la importancia de la seguridad e identificar los elementos que permitan proteger el sistema y la información.
- Crear aplicaciones dinámicas e interactivas de bases de datos para Internet con técnicas avanzadas del lenguaje PHP y la base de datos PostgreSQL.

INTRODUCCIÓN

Linux es un sistema basado en Unix, creado por Linus Torvalds en 1991. Linux se encuentra bajo la licencia GNU General Public License (Licencia Pública General), para que cualquiera pueda hackearlo y hacerle mejoras.

“Linux” sólo se refiere al Kernel, el centro del sistema operativo, parte responsable de controlar procesador, memoria, unidades de disco duro, y periféricos. Controla las funciones de la computadora y verifica que todos los programas funcionen correctamente. El Kernel y sus programas conforman un sistema operativo, al cual se le conoce como distribución de Linux.

Slackware fue la primera distribución de Linux en lograr popularidad entre la gente, fue creada por Patrick Volkerding. Slackware es diseñado bajo dos objetivos: facilidad para ser usado y estabilidad. Cuenta con un programa de instalación de uso fácil basado en un sistema de menús y el concepto de manejo de paquetes. Esto permite al usuario fácilmente agregar, actualizar o quitar paquetes de software del sistema.

Actualmente Linux ha crecido en el mercado de los sistemas operativos, corre en una variedad de arquitecturas, y se ha estado desarrollando por miles de programadores en el mundo. Ejecuta programas como: Apache, PHP, MySQL, PostgreSQL que son algunos de los software de servidor más populares en Internet y los cuales también se encuentran bajo la licencia GNU.

Apache es un servidor HTTP de código abierto para plataformas Unix, Windows y otras, su configurabiliadad, robustez y estabilidad hacen que cada vez millones de servidores reiteren su confianza en este programa. Apache es un servidor altamente configurable de diseño modular, trabaja con una gran cantidad de lenguajes de script, mensajes de error configurables, bases de datos de autentificación, entre otras.

PHP (acrónimo de "PHP: Hypertext Preprocessor") es un lenguaje de "código abierto" interpretado, usado generalmente para la creación de aplicaciones para servidores, programación de páginas Web dinámicas, creación de aplicaciones gráficas independientes del navegador, etc.

Permite la conexión a diferentes tipos de servidores de bases de datos tales como MySQL, Postgres, Oracle, ODBC, etc; lo cual permite la creación de Aplicaciones Web muy robustas. PHP se puede utilizar en una variedad de sistemas Operativos, soporta varios servidores Web y además permite las técnicas de Programación Orientada a Objetos.

MySQL es uno de los Sistemas Gestores de bases de Datos (SQL) más populares, tiene como características: velocidad, robustez, facilidad de uso, amplio subconjunto del lenguaje SQL, disponibilidad en gran cantidad de plataformas y sistemas, transacciones y claves foráneas, conectividad segura, entre otras.

PostgreSQL es un Sistema de Gestión de Bases de Datos Objeto-Relacionales (ORDBMS). PostgreSQL está ampliamente considerado como el sistema de bases de datos de código abierto más avanzado del mundo, gracias a su estabilidad, disponibilidad y su eficiencia.

Al ser Linux un sistema bajo la licencia GNU, así como la mayoría de las aplicaciones que se ejecutan sobre él, conduce a altos niveles de ***seguridad***. Aunque cualquiera puede acceder al código fuente para encontrar las debilidades en corto tiempo, también es corto el tiempo en que tarda en aparecer la solución para cualquier debilidad.

Gracias a esto, Linux es conocido por su alto nivel de estabilidad que parte del propio núcleo del sistema operativo. Linux tiene disponible todos los servicios habituales en una red: Bases de datos, servicios de Internet, utilidades necesarias para mantener el nivel de seguridad requerido, entre otros.

CAPITULO I

1. Sistema Operativo Linux.

1.1 Sistema Operativo Linux

Linux es el primer sistema operativo de la familia de los Unix realmente libre.

El Proyecto GNU General Public License (Licencia Publica General) fue lanzado en 1983 por Richard Stallman, para el desarrollo de un sistema operativo compatible con las operaciones de los Unix de ese momento, llamado GNU, teniendo la intención de ser completamente software libre. Muchos programas y utilidades fueron contribuciones de programadores de todo el mundo y hacia el año de 1991 la mayor parte de los componentes estaban listos. Pero aún faltaba algo muy importante, "el kernel".

En 1991, un estudiante de la universidad de Helsinki en Finlandia, teniendo por nombre Linus Torvalds, quién estaba usando un sistema Minix (sistema no libre) como base, comenzó a escribir su propio kernel. Este comenzó por escribir los controladores de los dispositivos y el acceso al disco duro y hacia septiembre de ese año ya tenía una base, a la cual la llamó versión 0.01. Este kernel, que posteriormente se llamaría Linux fue combinado con el sistema GNU, para producir así un sistema operativo completamente libre.

El 5 de octubre de 1991, Linus Torvalds, envió un correo a la lista de discusión "comp.os.minix", donde anunciaba la liberación de una versión 0.02 de su trabajo, una versión básica que aún necesitaba Minix para funcionar, pero que atrajo el interés de varios integrantes de la lista. El kernel fue rápidamente mejorado por Torvalds y un gran número de desarrolladores del mundo también estaban aportando sus conocimientos (todo esto por medio de Internet). Y hacia el 19 de diciembre del mismo año, un funcional e independiente sistema operativo fue liberado como la versión 0.11, bajo una licencia libre, esta ideada por el propio Linus Torvalds.

El 5 de enero de 1992, la versión 0.12 de Linux un kernel mejorado y estable, fue liberada bajo la licencia GPL (General Public License) de GNU, una licencia bien establecida. La siguiente versión fue la 0.95 reflejando el hecho de que este era un sistema completo y destacado. Después de esto el software desarrollado para Linux cada vez se fue haciendo más libre, permitiendo que Linux se hiciera un fenómeno, con un grupo muy amplio de desarrolladores se fue depurando y corrigiendo algunos errores en el kernel y así hasta estos días.

Linux siguió siendo mejorado a lo largo de los años 90's y comenzó a ser usado a gran escala como un repositorio de sitios Web (Web Hosting), servidor de bases de datos, usándose también en la infraestructura de las redes de ese momento. La versión 2.2, una modernización del kernel de Linux, oficialmente fue liberada en el año 1999. Hacia el año 2000 muchas compañías de computadoras apoyaron el proyecto de Linux de una u otra manera, reconociendo un estándar común que finalmente unifica el mundo fracturado de las guerras Unix. La versión principal fue la 2.4 que, entre sus principales mejoras, era la del soporte para las próximas generaciones de procesadores (procesadores de 64 bits).

1.1.1 Características, plataformas y distribuciones más comunes

Multitarea: Capacidad para ejecutar varias tareas simultáneamente sin necesidad de tener que parar la ejecución de cada tarea, es decir, el sistema operativo se encarga de administrar

el procesador, así las tareas se ejecutan simultáneamente en tiempos pequeños y el resultado es igual que si se estuviesen ejecutando al mismo tiempo.

Multiusuario: Varios usuarios pueden acceder a las aplicaciones y recursos del sistema Linux al mismo tiempo, ya sea localmente o en forma remota.

Multiprocesador y multiplataforma: Linux corre sobre varias de ellas.

Código fuente está disponible, incluyendo el núcleo completo, los drivers, las herramientas de desarrollo y los programas de usuario.

Múltiples consolas virtuales, permite diferentes sesiones, se crean dinámicamente hasta un máximo de 64.

Sistemas de archivos, soporta varios de ellos para guardar los datos. El ext2fs, ha sido desarrollado específicamente para Linux.

Un *S.O* es un conjunto de programas o software que permite la comunicación del usuario con la computadora y gestionar sus recursos de manera cómoda y eficiente. Comienza a trabajar cuando se enciende la computadora y gestiona el hardware de la computadora desde los niveles más básicos. *Plataforma* es la denominación que se le da a diferentes sistemas operativos, por ejemplo, Linux, Unix, Macintosh, Windows, etc. Algunas plataformas para Linux son: 386, 486, Pentium, Alpha, ARM, MIPS, SPARC, PowerPC/Macintosh, PC/Windows, Apple/Mac-Os, WS/Windows NT, Unix, WS/Linux o PC/Linux, etc.

Las *distribuciones,* son utilerías, programas, herramientas que con un kernel de Linux se distribuye en CD-ROM para poder ser instalado. El *kernel o núcleo* es la parte más oculta del sistema operativo, posee un conjunto limitado de comandos. Este contiene un código llamado controladores, con el permite al sistema controlar el hardware de la máquina desde el teclado hasta los servicios de red TCP/IP. Es imprescindible y viene en todas las distribuciones. Algunas distribuciones son: Slackware, Knoppix, Morphix, Red Hat, Mandrake, Conectiva, Debian, GNULinux, SuSe, TurboLinux, Caldera Open Linux, HispaFuentes, Esware, Fedora, Gentoo, Mklinux, etc.

1.2 Requerimientos de instalación

Para instalar Linux se necesita una computadora con procesador 80386 en adelante o equivalente y memoria RAM 16MB, monitor a color VGA, mouse y teclado, esto para una configuración mínima.

- **Hardware**

CPU: se necesita un procesador 386, 486, Pentium, Pentium Pro, etc. Linux soporta arquitecturas como: AMD, IBM, ALPHA, Cyrix, PowerPC, etc.

Memoria RAM: Linux necesita poca memoria para funcionar. Se debe tener al menos 16MB de RAM. Cuanta más memoria tenga más rápido será el sistema.

Disco Duro: La cantidad de espacio requerida depende de las necesidades de cada usuario y del software que se instale, así como de un espacio para guardar información del usuario.
Bus de E/S: El tipo de bus especifica como el CPU se comunica con el hardware y es una característica de la tarjeta madre. Se debe utilizar el bus ISA, EISA, PCI, o VL.

CD-ROM: Como casi todas las distribuciones de Linux vienen en CD-ROM se necesita un lector que puede ser IDE, SCSI o con norma propia, a una velocidad de 2x, 8x o mayor.

Monitor y adaptador de video: Linux soporta desde una hércules, pero es recomendable usar una placa de video compatible con VGA para la terminal de la consola. Prácticamente cualquier tarjeta gráfica moderna es compatible con VGA, CGA, MDA, monocromo, etc.

Otro Hardware: Linux soporta una gran variedad de dispositivos como mouse, impresora, escáner, módem, tarjeta de red, etc. Sin embargo, no se requiere ninguno de estos dispositivos durante la instalación del sistema.

- **Software**

Es el conjunto de programas de cómputo (en sus distintas formas: código fuente, binario o código ejecutable), procedimientos, reglas, documentación y datos asociados que puede ejecutar el hardware para la realización de las tareas de computación. El software es la parte intangible de la computadora.

La *GNU GPL* (General Public License o licencia pública general) es una licencia creada por la Free Software Foundation a mediados de los 80, y esta orientada principalmente a los términos de distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software libre.

GNU/Linux es la denominación defendida por Richard Stallman y otros para el sistema operativo que utiliza el kernel Linux en conjunto con las aplicaciones de sistema creadas por el proyecto GNU. Comúnmente este sistema operativo es denominado simplemente Linux.

1.3 Comandos y utilerías básicas

La sintaxis básica de todos los comandos es:

comando [opción] argumento

donde:

- *comando*: es el nombre del comando.
- *[opción]*: son caracteres opcionales propios de cada comando.
- *argumento*: son los objetos con los que va a trabajar el comando (normalmente son archivos).

Se pueden poner varios comandos en una misma línea separándolos con un punto y coma (;). Algunos comandos básicos son:

- *date*: Se utiliza para obtener la fecha y la hora del sistema.
- *cal*: Se utiliza para obtener el calendario del mes actual del sistema

cal mes año: Muestra el mes y año requerido. El mes se representa numéricamente (1-12), el año es requerido, pues si se pone un único número en la línea de comando, *cal* considera que se refiere al año y no al mes.

- *uname*: Muestra el tipo de sistema en el que se esta trabajando.
- *who*: Muestra una lista de los usuarios que están conectados al sistema. En la salida se observa el nombre del usuario, etiqueta de la terminal que se esta utilizando, la fecha y hora en que se conecto al sistema.
- *whoami*: Muestra el nombre del usuario.

1.3.1 Inicio y término de sesión

Al encender la computadora, elegir la entrada Linux del menú de sistemas disponibles que aparece durante el inicio. Para iniciar una sesión de trabajo en Linux hay que identificarse en el cuadro de diálogo inicial que aparecerá, proporcionando los datos de usuario y contraseña.

Una vez identificado dentro del sistema ya se puede empezar a trabajar. Linux puede trabajar en modo línea de comando y en modo entorno gráfico (con ventanas).

El administrador de un equipo Linux es el usuario root cuya contraseña se establece en el momento de la instalación. No conviene utilizar este usuario para sesiones de trabajo normal, es preferible hacerlo como un usuario normal, sin derechos de administración que siempre son un riesgo.

Para terminar una sesión se teclea *exit* o [Ctrl]+[D].

1.3.2 Archivos y directorios

Un *archivo* es una colección de información que se almacena en un disco. Los archivos del sistema son estructuras que la computadora utiliza para organizar y almacenar información. Existen 2 tipos de archivos del sistema:

- *Archivos ordinarios (-)*: estos archivos contienen datos, textos y programas ejecutables.
- *Archivos directorios (d)*: estos archivos contienen nombres de archivos. Los directorios no se utilizan para almacenar datos, sino que se utilizan para organizar otros archivos en grupos.

Todos los archivos tienen asociado un "nombre de archivo"; este nombre identifica el archivo y su contenido. El nombre de un archivo puede tener de 1 a 255 caracteres; pero se pueden utilizar únicamente los siguientes caracteres: Letras mayúsculas y minúsculas (A - Z , a - z), números (0 - 9), subrayado (_), punto (.) y coma (,).

Los archivos cuyos nombres comienzan con un punto (.) se llaman "archivos invisibles" porque no aparecen cuando se pide el listado de un directorio, estos generalmente almacenan información que el sistema utiliza automáticamente.

Directorio raíz se le llama a aquel directorio que contiene todos los demás directorios y archivos, es decir, toda la estructura en árbol de los directorios y archivos crecen a partir de

él. El directorio raíz se indica siempre con una diagonal (/). El *directorio home*, es un subdirectorio del directorio raíz, es donde residen los directorios y archivos del usuario, generalmente tiene el mismo nombre que el nombre de usuario.
El *directorio de trabajo*, es el directorio en el que el usuario se encuentra en ese momento. Cuando se accede por primera vez al sistema, el directorio de trabajo se configura como el directorio del usuario, por ejemplo: --/home/pedro. *pwd* (Path to working directory) es el comando que muestra el nombre y ruta de acceso del directorio de trabajo actual.

Algunos comandos para trabajar con archivos y directorios

Comando	***Descripción***
Cd	Cambia de directorio. Se pueden utilizar rutas absolutas o relativas.
Ls	Lista el contenido del directorio actual, mostrando el nombre de los archivos; si se quiere obtener más información sobre esos archivos se utilizan las opciones del comando, cuya sintaxis general es: *ls [opción] archivo o directorio*
Medir	Crea directorios, su sintaxis general es: *mkdir nombre*
Cp	Copia archivos y directorios, su sintaxis general es: *cp Origen Destino*
Mv	Renombra (el contenido del archivo no cambia, sólo cambia el nombre) o mueve archivos y directorios, su sintaxis general es: *mv Origen Destino*
rm, rmdir	Borra archivos, su sintaxis general es: *rm nombre_ archivo*
cat, more	El comando ***cat,*** se utiliza para visualizar por pantalla el contenido de uno o más archivos. Su sintaxis general es: *cat nombre_archivo* El comando *more*, se utiliza para mostrar información pantalla a pantalla, de un archivo de texto muy largo. Su sintaxis general es: *more nombre_archivo*

1.3.3 Rutas relativas y absolutas

Al especificar los archivos en el directorio de trabajo actual, se puede hacer referencia a ellos sólo con el nombre de archivo correspondiente. Pero, al hacer referencia a directorios y archivos que estén fuera del directorio de trabajo actual, se debe utilizar *nombres de ruta*. Existen dos tipos de rutas:

Rutas relativas: Indican el acceso directo a la ubicación de los archivos y directorios desde el directorio de trabajo actual hasta el archivo deseado.

Rutas absolutas: Indican la ruta que conduce a un directorio o archivo desde el directorio raíz del sistema, a través de cada subdirectorio, hasta el archivo deseado. El último nombre de la ruta es el directorio o el archivo al que se desea llegar.

1.3.4 Redireccionamiento

El *redireccionamiento* es hacer que la *shell* cambie lo que está considerado como entrada estándar o el lugar donde va a almacenar la salida estándar. Todos los programas tienen por defecto una entrada estándar (teclado) y dos salidas: la salida estándar (pantalla) y la salida de error (pantalla). En ellos se puede sustituir la entrada y salidas estándar por otro dispositivo, es decir, hacer que se lea un archivo que contenga las opciones a ejecutar y un archivo de salida, respectivamente.

- *Entrada estándar o stdin*, es de sólo lectura y representa generalmente al teclado, los programas leen a través de esta entrada todo lo que se escribe.
- *Salida estándar o stdout*, es de sólo escritura y representa la salida de un programa, cuando un programa quiere escribir algo en la pantalla, lo que hace es escribir en esta salida.
- *Error estándar o stderr*, es de sólo escritura, y se utiliza para imprimir mensajes de error.

La entrada estándar como la salida no necesariamente debe ser el teclado y la pantalla respectivamente, es decir, se puede cambiar dicha entra y salida utilizando los siguientes operadores:

- >: Símbolo para redireccionar la salida estándar a un archivo
  comando > nombre_archivo
- <: Símbolo para redireccionar la entrada estándar a un archivo
  comando < nombre_archivo
- >&: Símbolo para redireccionar la salida y el error estándar a un archivo
  comando >& nombre_archivo

El *redireccionamiento no destructivo*, permite añadir la salida o el error estándar a un archivo, es decir, conserva la información actual que hay en un archivo, y agrega la nueva información al final del mismo. Si el archivo no existe, este comando lo creará. Para ello se utiliza el doble signo mayor que (>>).

- >>: símbolo para añadir la salida a un archivo
  comando >> nombre_archivo
- >>&: Símbolo para añadir la salida y error a un archivo
  comando >>& nombre_archivo

1.3.4.1 Uso de tuberías, |

En la línea de comandos la integración entre diferentes programas se realiza por medio de la redirección de las entradas y salidas a través de *pipes* o tuberías. Estas relacionan la salida estándar de un comando con la entrada estándar de otro comando. Es decir transfiere datos entre diferentes procesos.

- |: Símbolo de un pipe
  comando_1 | *comando_2* | ...

Las tuberías también se pueden usar para imprimir solamente determinadas líneas de un archivo. Utilizando tuberías no es necesario utilizar archivos temporales ni hacer pasos intermedios para obtener la información que se desea.

1.3.4.2 Filtros

Un filtro es un comando que lee datos de la entrada estándar, los procesa de alguna forma, y manda los datos procesados a la salida estándar. Usando la redirección, la entrada y salida estándar pueden ser referenciadas desde archivos. Una línea de comando puede contener varios filtros.

Algunos comandos que pueden ser usados con los filtros

Comando	***Descripción***
wc	Se utiliza para saber el número de líneas y caracteres dentro de un archivo. Su sintaxis es: *wc [opción] [archivo]* Se pueden combinar varios argumentos a la vez. Si se omite el argumento *archivo*, toma los datos de la entrada estándar.
last	Comando que muestra una lista de entrada y salida de usuarios que se han autenticado en el sistema, nombre de usuario, que terminales usaron, tiempo que estuvo conectado al sistema. Su sintaxis es: *last [opción]*
grep	Permite buscar las líneas que contiene una cadena de caracteres especificada mediante una expresión regular. Lee la entrada estándar o una lista de archivos y muestra en la salida sólo aquellas líneas que contienen la expresión indicada. Su sintaxis general es: *grep [opción] patrón archivo (s)* *patrón*: Expresión regular a buscar, esto para realizar búsquedas más flexibles. *archivo*: donde se buscara el patrón.
find	Se utiliza para buscar archivos dentro de una rama de directorios, que cumplan con determinados criterios. Su sintaxis general es: *find [ruta] [expresión]*
tee	Permite desviar una copia de los datos que se transmiten entre los comandos a un archivo sin modificar el modo en que funciona la canalización.

1.3.5 Atributos y permisos de archivos

Los *atributos* de un archivo son: el nombre, el tipo, la localización (donde se ubica), derechos de acceso, tiempo de creación, tiempo de acceso, tiempo de modificación, UID del creador, etc.

Los *permisos* de un archivo son definidos para tres tipos diferentes de usuarios y tres modos diferentes de acceso al archivo.

Clases de usuarios:

- *Propietario*: Usuario que crea el archivo, tiene la capacidad de controlar quien puede acceder al archivo.

- *Grupo*: Grupo de usuarios, normalmente relacionados por un departamento o función. Un usuario de este tipo puede acceder al archivo, pero no puede cambiar quien puede acceder al archivo.
- *Otros*: Cualquier otro usuario del sistema. Estos usuarios pueden únicamente acceder al archivo si tienen permiso para ello.

Modos de acceso al archivo:

- *Lectura (r)*: Permite a un usuario leer el contenido del archivo o en el caso de un directorio, listar el contenido del mismo (usando ls).
- *Escritura (w)*: Permite a un usuario escribir y modificar el archivo. Para directorios, permite crear nuevos archivos o borrar archivos ya existentes en el directorio.
- *Ejecución (x)*: Permite a un usuario ejecutar el archivo si es un programa o guión del intérprete de comandos. Para directorios, permite al usuario cambiar al directorio en cuestión con cd y buscar en el.

Para asignar el tipo de usuario y el modo de acceso se utilizan los comandos: *chown* y *chgrp*.

El comando *chown*, permite cambiar el propietario de los archivos del sistema de archivos. Pero cuando se es un usuario normal no se podrán cambiar de propietario los archivos que pertenecen a root o a otros usuarios. Pero como root si se podrá cambiar el propietario de cualquier archivo. Sintaxis general:

- *chown usuario archivo(s)*: Se cambia el nombre del usuario y el grupo se conserva.
- *chown usuario:grupo archivo(s)*: Se cambia el nombre del usuario y grupo.
- *chown -R usuario:grupo directorio/*: Se cambia el usuario y grupo dueños tanto del directorio y de todo el contenido, es decir, lo hace recursivamente.

El comando ***chgrp,*** permite cambiar el grupo de un archivo o lista de ellos.

- *chgrp grupo archivo(s)*: Se cambia el grupo.

El comando ***chmod,*** se utiliza para cambiar los permisos de un archivo y de un directorio. Existen dos formas de cambiar los permisos. Se pueden cambiar teniendo en cuenta los permisos existentes (modo simbólico), o se pueden asignar permisos independientemente de los ya existentes (modo absoluto).

Modo simbólico: Dentro de este se pueden añadir o quitar permisos a los archivos y directorios. El formato del comando es:

chmod [who] código-operador permisos archivo

- *who*: A que tipo de usuario afecta.
  u - Usuario, *g* - Grupo, *o* - Otros, *a* - Todos los usuarios del sistema.

- *Código-operador*: Indica la operación que se va a realizar.
  + : Añade permisos, - : Quita permisos.

- *Permisos*: Tipo de permiso.

r - Lectura, *w* - Escritura, *x* - Ejecución

Modo absoluto u octal: Este modo se especifica con 3 dígitos numéricos; cada número representa los permisos de cada tipo de usuario. Estos dígitos se obtienen, para cada clase de usuario, a partir de los valores siguientes:

- *4* : permiso de lectura
- *2* : permiso de escritura
- *1* : permiso de ejecución.

Así se tiene:

- *0* : ningún permiso
- *1* : permiso de ejecución
- *2* : permiso de escritura
- *3* : permiso de ejecución y escritura (1+2)
- *4* : permiso de lectura
- *5* : permiso de lectura y ejecución (4+1)
- *6* : permiso de lectura y escritura (4+2)
- *7* : permiso de lectura, escritura y ejecución (4+2+1)

La sintaxis para el comando es: *chmod modo archivo*

- *modo*: Son 3 dígitos numéricos. Cada uno de ellos corresponde a los permisos de cada tipo de usuario.

1.3.6 Ligas

En Linux existe la posibilidad de generar ligas a directorios y archivos. Las ligas permiten dar a un único archivo múltiples nombres. Los archivos son identificados por el sistema por su número de *inodo* (nodo-i, nodo índice o i-node en inglés), el cual es una estructura de datos propia de los sistemas de archivos. Un inodo contiene las características (permisos, fechas, ubicación, pero no el nombre) de un archivo, directorio, o cualquier otro objeto que pueda contener el sistema de archivos. Cada inodo queda identificado por un número entero, único dentro del sistema de archivos, y los directorios recogen una lista de parejas formadas por un número de inodo y nombre identificativo que permite acceder al archivo en cuestión: cada archivo tiene un único inodo, pero puede tener más de un nombre en distintos o incluso en el mismo directorio para facilitar su localización.

Un directorio es una lista de números de inodo con sus correspondientes nombres de archivo. Cada nombre de archivo en un directorio es un enlace a un inodo particular. Así las ligas no son una copia del archivo, sino que son referencias a un archivo, y a la hora de trabajar son equivalentes al propio archivo, se pueden editar, modificar y los resultados se actualizan en el archivo original.

Para crear una liga a un archivo se utiliza el comando *ln*:

ln archivo_origen archivo_destino

Las ligas pueden ser duras (físicas - hard) o suaves (simbólicas).

- Las ligas duras son una copia del archivo, pero cada modificación que se haga en un archivo se actualizará en el otro, el contenido del archivo no se perderá hasta que no se borren todos los enlaces. Sólo se pueden crear ligas duras entre archivos del mismo sistema de archivos. Las ligas que se generan con el comando *ln*, por defecto son duras.
- Las ligas suaves, en realidad son accesos directos al archivo, y si se borra el archivo queda el enlace referido a un archivo inexistente. Para generar ligas suaves se utiliza el comando *ln -s*.

1.3.7 Control de trabajos y procesos

Un proceso puede ejecutarse en *foreground (Primer Plano)* o en *background (Segundo Plano)*. Pero sólo puede haber un proceso en primer plano a la vez, el proceso que está en primer plano es con el que se interactúa, recibe entradas de teclado y envía las salidas al monitor mientras no se haya especificado otra cosa, así el prompt del sistema no reaparece hasta que el comando ha terminado de ejecutarse. El proceso en segundo plano, no recibe ninguna señal desde el teclado por lo general, el prompt del sistema reaparece aunque no se haya terminado de ejecutar el comando.

Un programa se transforma en proceso en el momento en que éste se ejecuta y esta en memoria. Además del nombre que el proceso recibe, que es el nombre del programa que esta corriendo, recibe también un número identificativo llamado PID (process ID o ID de proceso). Si se ejecuta el comando *ps* se observan los procesos que se están ejecutando en este momento con el UID (identificador de usuario), es decir que el usuario esta corriendo. Si se ejecuta el comando *ps –ax* se observan los procesos que se están ejecutando en el sistema.

Con los comandos anteriores se observara, que se encuentran varios procesos ejecutándose al mismo tiempo, pero sólo uno de ellos esta activo. Para verificar cual es el proceso que se esta ejecutando, se debe observar la columna *STAT* aparecerá en la línea del proceso inactivo la letra *S* de *SLEEP*, ya que en ese momento el intérprete de comandos esta esperando a que el proceso termine.

1.3.7.1 Suspensión y reinicio de trabajos, fg, bg

Para interrumpir un trabajo se teclea Ctrl-C (generalmente), cuando se suspende el proceso deja de estar en memoria y se pierden todos sus datos no guardados, aunque algunos programas cuando capturan que se ha pulsado Crtl-C guardan todos sus datos ordenadamente y salen. Una vez interrumpido, el proceso no puede continuar ejecutándose, y deberá ser lanzado otra vez para volver a realizar sus tareas. Una forma de mandar procesos a segundo plano es añadiendo un carácter "&" al final de cada línea de comando justo antes de presionar enter.
Para eliminar un trabajo, se utiliza el comando *kill*. Este comando toma como argumento un número de trabajo o un número de PID. Cuando se identifica el número de trabajo, se debe preceder el número con el carácter de porcentaje ("%").No es necesario usar el "%" cuando se refiere a un trabajo a través de su PID.
El comando kill no sólo sirve para eliminar una tarea sino también sirve para enviar todo tipo de señales. Con la opción *-15* manda la señal TERM que hace que el proceso guarde

sus datos antes de finalizar, con *-9* se elimina la tarea igual que kill, con -19 se para la tarea (STOP), con -18 se reinicia la tarea (CONT).

Los procesos pueden ser suspendidos, así un proceso suspendido es aquel que no se está ejecutando actualmente, pero sus datos siguen en memoria. Después de suspender una tarea, puede indicar a la misma que continúe, en primer plano o en segundo, según necesite. Retomar una tarea suspendida no cambia en nada el estado de la misma, la tarea continuará ejecutándose justo donde se dejó.

fg: Se utiliza para pasar un trabajo que se está corriendo en background a foreground. Su sintaxis general es:

fg [% número trabajo]

número trabajo: Es el número del trabajo que se quiere pasar a foreground. Este número es el que aparece en la 1ª columna de la salida del comando jobs. Sino se especifica ningún número, se pasará a foreground el trabajo que aparezca con un signo - , en la salida del comando jobs.

bg: Se utiliza para pasar un trabajo que se está corriendo en foreground a background. Para ello se realizan los siguientes 2 pasos:

- Se suspende el trabajo tecleando y entonces aparecerá el prompt del sistema.
- Se manda ejecutar el trabajo en background con el comando *bg*.

jobs: Lista por pantalla todos los trabajos que se han lanzado en background. La salida de este comando es de 4 columnas:

1ª columna: Corresponde al número de orden en que se van a ejecutar los comandos.
2ª columna: Puede estar en blanco, contener un signo más (+) el cual indica el último trabajo que se ha parado o un signo menos (-) el cual indica el penúltimo trabajo que se ha parado.
3ª columna: Indica si el trabajo está en background o se ha parado.
4ª columna: Contiene el nombre del comando que se está corriendo en background o que está parado.

1.3.8 Editor vi

Un editor de texto es simplemente un programa usado para la edición de archivos que contienen texto, como un programa en C, un archivo de configuración del sistema. Mientras que hay muchos editores de texto disponibles en Linux, el único editor que está garantizado encontrar en cualquier sistema UNIX es *vi* el "visual editor". El cual es pequeño y potente, pero difícil de usar, usa una pequeña cantidad de memoria, lo cual permite una operación eficiente cuando la red está muy ocupada. Además, usa teclas alfanuméricas estándares para los comandos, para poder usarlo en alguna terminal virtual o una estación de trabajo sin tener que preocuparse por mapeos de teclas inusuales.

vi tiene dos modos: *modo comando* en éste vi espera alguna orden, *modo edición* (inserción) en éste vi espera que se escriba texto. Cuando se abre un editor, éste se encuentra en modo comando; para pasar al modo de edición se puede pulsar i (insertar), a

(añadir o añadir una línea). Para pasar al modo de comandos, se pulsa Escape o Suprimir. Su sintaxis es:

vi <nombre_archivo>

nombre_archivo: Es el nombre del archivo que se desea editar. Si el archivo no existe *vi* lo crea.
Si sólo se teclea *vi*, en este caso pedirá un nombre de archivo cuando grabe éste. La columna de caracteres "~" indica que está al final del archivo.

Inserción de texto.

a	Añade texto después del cursor
A	Añade texto al final de la línea.
i	Inserta el texto antes del cursor.
I	Añade texto delante del primer carácter de la línea.
o	Abre una nueva línea después de la actual.
O	Abre una línea antes de la actual.

Borrado de texto.

x	Borra del carácter actual hacia la izquierda.
d	Borra la línea actual.
5d	Borra 5 líneas hacia delante a partir de donde se encuentra el cursor.
Shift+d	Borra desde el cursor hasta el final de la línea.
dw	Borra una palabra.

Guardar archivos y terminar vi.

:q!	Salir sin hacer cambios.
:wq	Salvar los cambios y salir del programa.
vi –r	Iniciar vi recuperando el último archivo que no fue cerrado apropiadamente desde vi.
vi –R archivo	Abrir el archivo en modo sólo de lectura.
:w	Salvar el archivo sin salir de vi.

Buscar

:/palabra	Busca la palabra solicitada.
/	Repite la última búsqueda.
?palabra	Hace la búsqueda de la palabra desde donde se encuentra el cursor hacia el inicio del documento.

1.3.9 Comandos shutdown, halt, reboot

En Linux una de las cosas importantes es el correcto apagado del sistema. Este no se debe apagar simplemente presionando el botón de power, ya que, el filesystem podría ser dañado y por lo tanto el correcto funcionamiento del sistema también. Hay varios métodos correctos para poder reiniciar o apagar el sistema.

El comando *shutdown*, se utiliza tanto para apagar el sistema como para reiniciarlo. Este comando cuenta con varias opciones como:

- *shutdown -h now*: Apaga la computadora en ese mismo instante. La opción *–h* detiene el sistema, *now* significa ahora.
- *shutdown -h +tiempo*: Apaga la computadora, según el tiempo indicado para apagar.
- *shutdown -r now*: Reinicia la computadora en ese mismo instante. La opción *–r* reinicia el sistema, también cuenta con la opción de minutos para reiniciar el sistema.

El comando *halt* se utiliza para apagar el sistema.
El comando *reboot* se utiliza para reiniciar el sistema.

1.3.10 Montaje de sistemas de archivos "mount".

En Linux, antes de que un disco duro (o cualquier otro dispositivo de almacenamiento) pueda ser utilizado, necesita el sistema "montar o colocar" ese dispositivo dentro de un directorio, el cual generalmente es el directorio /mnt, aunque puede ser cualquier otro.

El comando para montar dichos dispositivos es *mount*, su sintaxis es:

mount [-t tipo] dispositivo directorio

Este comando monta el dispositivo que se especifica en el directorio, que deberá existir en algún lugar del árbol de directorios del sistema. Se puede configurar un archivo que contenga una lista de dispositivos y sus respectivos directorios de montaje.
Una vez "montado" el dispositivo, si ya no se utiliza hay que "desmontar" para poder introducir nuevos dispositivos de almacenamiento, como en el caso de un CD-ROM, por ejemplo. Esto se hace con el comando *umount*, su sintaxis es:

umount dispositivo

1.3.11 Manejo de usuarios y grupos.

Para manejar los usuarios y grupos, se tienen programas y scripts que proporciona el mismo sistema. El sistema Slackware permite trabajar con los usuarios gracias a programas como: adduser, userdel, chfn, chsh, passwd, useradd, usermod. Y para trabajar con grupos proporciona programas como: groupadd, groupdel y groupmod. Todos estos programas son usados por root. Los usuarios sólo pueden usar programas como: chfm, chsh y passwd.

Para ***crear una cuenta***, se debe de iniciar sesión como root, los comandos ***useradd*** y ***adduser*** permiten crear dicha cuenta.

- *useradd nombre_usuario*

Donde *nombre_usuario* es el nombre que se le va asignar a la nueva cuenta. Al crear un nuevo usuario, el sistema crea por defecto una nueva carpeta dentro del directorio /home con el nombre del usuario.

- *adduser*

Al utilizar esta forma para agregar un usuario pide una serie de datos los cuales se irán tecleando de acuerdo a los datos del usuario.

Para ***cambiar la contraseña*** existente o especificar una contraseña a un usuario, se utiliza el comando *passwd* de la siguiente manera:

passwd nombre_del_usuario

El sistema solicitará se teclee la nueva contraseña para el usuario y que repita ésta para confirmar.

Para ***borrar un usuario*** del sistema se debe estar logeado como root y utilizar el comando *userdel*. Así este comando elimina la cuenta de usuario del sistema y de los archivos (/etc/passwd, /etc/shadow y /etc/group), pero no elimina el directorio que utiliza como su home.

userdel nombre de usuario a eliminar

Para borrar la cuenta del usuario del sistema y su directorio home, se utiliza la opción *-r*. Esta opción no elimina archivos del usuario en otro sistema de archivos, estos deben ser buscados y borrados manualmente por el administrador.

Se debe tomar en cuenta que si el usuario se encuentra logeado en el sistema en ese instante o si hay algún proceso ejecutándose que pertenezca al usuario no se podrá eliminar su cuenta del sistema.

Para ***cambiar información del usuario***, se utilizan los comandos *chsh* y *chfn*.

- *chsh*

Cambia el intérprete de comandos (la shell) que se ejecuta cuando el usuario entra al sistema. Existen varios intérpretes de comandos, estas se listan en el archivo /etc/shells. Al teclear *chsh* y dar enter**,** primero pedirá la contraseña del usuario y posteriormente la ruta de la nueva shell que se desea utilizar.

- *chfn*

Cambia el nombre completo de un usuario y otra información (extensión y número de teléfono de su oficina y número de teléfono de su casa).

También root puede cambiar la información de los usuarios, root utilizara el comando seguido del nombre de usuario al cual se le cambiara la información.

Para ***agregar o quitar un grupo*** se utilizan los siguientes comandos:

- *groups*: muestra los grupos a los que pertenece un usuario.
- *groupadd*: crea un nuevo grupo usando los valores especificados en la línea de comandos y los valores por defecto del sistema.
- *groupdel*: modifica los archivos con las cuentas del sistema, borrando todas las entradas referidas a un determinado grupo.
- *groupmod*: modifica los archivos de las cuentas del sistema para reflejar los cambios especificados en la línea de comandos.

1.3.12 Archivar y comprimir

Los archivos comprimidos utilizan menos espacio en el disco y se descargan más rápido que los archivos no comprimidos.

- Comando *tar* se utiliza para archivar, tiene la extensión *.tar*

Empaqueta varios archivos en uno solo, pero no comprime los datos. Los archivos *tar* contienen diferentes archivos, el contenido de un directorio o directorios en un archivo. Éste es un modo de crear copias de seguridad para guardar la información en un disco o para enviar por correo electrónico. Algunas opciones son:

-*c*: create, crear un archivo nuevo.
-*t*: table, ver el contenido de un archivo.
-*x*: extract, descomprimir un archivo.
-*v*: verbose, ver todos los archivos en la pantalla durante el proceso de compresión.
-*f*: file, con esta opción puede dar un nombre al archivo de empaquetamiento. Al comprimir, esta opción se introducirse en último lugar.

Para empaquetar: *tar -cvf archivo.tar /dir/a/comprimir/*
Para desempaquetar: *tar -xvf archivo.tar*
Para ver contenido: *tar -tf archivo.tar o tar –tvf archivo.tar*

- Comando *gzip* comprime un archivo, tiene la extensión *.gz*

Para comprimir: *gzip archivo*

Para descomprimir archivos que hayan sido comprimidos con gzip se tienen dos opciones: el comando "gzip -d" o el comando gunzip. Este comando básicamente lo que hace es descomprimir cualquier archivo que tenga una extensión reconocible.

gunzip archivo.gz o gzip -d archivo.gz

- Comando *bzip2* comprime archivos tiene la extensión *.bz2*.

Para comprimir: *bzip2 archivo*
Descomprimir: *bzip2 archivo.bz2*

1.5 Fuentes de información

Casi todas las fuentes de información sobre Linux están disponibles principalmente de forma electrónica, aunque también hay numerosos libros, además varias distribuciones de Linux incluyen documentación dentro de ellas, es decir, cuando se instala Linux se puede disponer de la documentación en el sistema.

El comando *man*, se utiliza para obtener ayuda sobre otros comandos. Tomar como argumento el nombre del comando del que se quiere ayuda, y buscar en una base de datos lo referente a ese comando. Si no se encuentra, dice que no existe. Su sintaxis general es:

man comando

La salida de este comando tiene varias secciones; nombre, sintaxis o formato que debe utilizar, descripción, nombre de los archivos que se utilizan en el comando, nombre de otros comandos que realizan funciones similares a la del comando descrito, discusión de cualquier mensaje de diagnóstico que produce el sistema y explicación de los errores más frecuentes.

- *man -k* palabra clave: La salida será una lista con todos los comandos relativos a la palabra clave (una palabra, parte de una palabra, o una frase en este caso, la frase va entre comillas sencillas) especificada.

- *man man*: Para obtener más información acerca del comando *man*.

La ***descripción de procedimientos*** se encuentra en los llamados ***HOWTOs***, los cuales son documentos informales, que explican acerca de cómo cumplir con una cierta tarea, son generalmente cortos. Proceden de contribuciones desinteresadas de usuarios bien informados y están disponibles en distintos formatos (HTML. texto plano, etc).

Los HOWTOs son generalmente creados para ayudar a personas inexpertas en un tema y suelen dejar de lado detalles para expertos, por lo que son en general un resumen del tema tratado. Existen un gran número de documentos de este tipo, conteniendo información sobre multitud de temas relacionados con el funcionamiento, configuración e instalación de este sistema operativo.

Otros tipos de documentación son FAQs (*Frequently Asqued Questions*), manuales y guías.

2. Instalación y Administración de Linux

2.1 Perfil y actividades de un administrador de sistemas

Un administrador de sistemas (en inglés abreviado sysadmin) es el trabajo de la persona que tiene la responsabilidad de asegurarse del correcto funcionamiento de un sistema informático, o algún aspecto de éste.

El significado preciso varía. En las organizaciones con un sistema muy grande y complicado, generalmente dividen al personal informático según su especialidad. En este caso un administrador de sistemas es aquel responsable del mantenimiento de un sistema informático existente.

2.1.1 Actividades de un Administrador

Las principales responsabilidades de un administrador de sistemas son:

a) creación de copias de seguridad
b) actualizaciones del sistema operativo y nuevas configuraciones
c) instalación de nuevo hardware y software
d) responder consultas técnicas
e) responsable de la seguridad del sistema
f) responsable de documentar la configuración del sistema
g) resolución de problemas
h) configuración óptima del sistema
i) administración de las cuentas de los usuarios
j) administración de los recursos del servidor

En organizaciones grandes algunas de estas tareas se dividen entre diferentes administradores de sistemas, pero en organizaciones pequeñas la lista anterior se extiende aún más, agregando:

k) soporte técnico
l) administrador de base de datos
m) administrador de red
n) programador, etc.

Todas y cada una de estas actividades deben ser planeadas.

2.1.2 Documentación

Como administrador de un sistema informático una de las principales tareas, es la documentación y esta debe comenzar desde la instalación hasta las últimas configuraciones que se le hagan al sistema, esto porque, por ejemplo si el sistema sufre un fallo y no se encuentra el administrador, él debe de haber dejado toda la documentación del sistema y poder así recuperarlo, puesto que la pérdida de un servidor puede ocasionar pérdidas multimillonarias dentro de una empresa.

Hay diferentes formas de realizar la documentación e incluso se puede ocupar cualquier suite ofimática (en Linux Open Office), o más profesionalmente, ocupar Latex o en un simple archivo de texto, lo importante es que ésta debe ser clara y con un formato definido, para que cualquier otro administrador pueda entenderlo.

Una de las tareas del administrador es la de proporcionar manuales de uso del sistema a los usuario, ya sean en línea, impresos o en Internet.
El principal manual que debe ofrecer el administrador es el que se puede instalar desde el principio de la instalación del sistema operativo. Y se llama *man*.

man es el paginador del manual del sistema. Las páginas usadas como argumentos al ejecutar /usr/bin/man suelen ser normalmente nombres de programas, utilerías o funciones del sistema, por ejemplo, para pedir el manual de uso del comando /bin/ls, se hace de la siguiente forma:

/usr/bin/man ls

2.1.3 Políticas de uso del Sistema y de Administración

Una política es una forma de comunicarse con los usuarios, ya que las mismas establecen un canal formal de actuación del personal, en relación con los recursos y servicios informáticos de una organización.

No se puede considerar, que una política es una descripción técnica de mecanismos, ni una expresión legal que involucre sanciones a conductas de los usuarios, es más bien una descripción de lo que se desea proteger y él por qué de ello, pues cada política es una invitación a cada uno de los usuarios a reconocer la información como uno de sus principales activos.

Las Políticas principalmente deben considerar los siguientes elementos:

a) Alcance de las políticas, incluyendo facilidades, sistemas y personal sobre la cual aplica.
b) Objetivos de la política y descripción clara de los elementos involucrados en su definición.
c) Responsabilidades por cada uno de los servicios y recursos informáticos aplicado a todos los niveles de la organización.
d) Definición de violaciones y sanciones por no cumplir con las políticas.
e) Responsabilidades de los usuarios con respecto a la información a la que tiene acceso.

Un punto importante, es que las políticas deben redactarse en un lenguaje sencillo y entendible, libre de tecnicismos y términos ambiguos que impidan una comprensión clara de las mismas, claro está sin sacrificar su precisión.

2.1.4 Mantener canales de comunicación con los usuarios

El mantener canales de comunicación con los usuarios, es muy importante para los administradores de sistemas, puesto que puede ser una forma de administrar a esos usuarios sin estar sentado con ellos a la hora de solucionarles un problema o incluso para una simple consulta. Existen diferentes formas de poder crear y mantener estos canales de comunicación y es que ya existen varias aplicaciones, por ejemplo: desde archivos de inicio de sesión (/etc/motd), o programas: /usr/bin/wall y /usr/bin/write, hasta servicios corriendo en el sistema, por ejemplo un servicio de noticias (news), de correo (mail) o de páginas web.

2.2 Instalación del sistema

Para poder iniciar la instalación del sistema, se tienen que conseguir los medios con los cuales se puede realizar la instalación y esto es por medio de la compra del paquete de CD's desde la página oficial de Slackware (http.//www.slackware.com) o también se pueden bajar los ISO's de Internet y posteriormente copiarlos a unos CD's.
Para una instalación mínima, son demasiado simples los requerimientos de hardware, a continuación se muestra una tabla en la cual se listan los requisitos mínimos del sistema:

Hardware	Requerimiento mínimo
Procesador	386
RAM	16MB
Espacio en disco	500MB
Flopy Drive	1.44MB

La instalación de slackware puede hacerse de varias formas, por ejemplo, desde un floppy con los archivos de booteeo de slackware, desde la red (NFS) o desde los CD's de la distribución. En este documento se describe la instalación de slackware desde los CD's.

Dos de los requerimientos de esta instalación son, una unidad de CD-ROM y una configuración mínima del BIOS, la cual es, la selección de la unidad de CD-ROM como la primera unidad de booteo.

Una vez hechas estas configuraciones se reinicia el equipo con el primer CD de la distribución en la unidad lectora de CD's, cuando se haya booteado el sistema, se presentará la pantalla de bienvenida, en ese momento se deberá seleccionar el kernel con el que se va a iniciar la instalación, con la tecla F2 se pueden ver algunas de las opciones que se le pueden agregar al kernel seleccionado y con la tecla F3 se puede ver una lista completa de todos los kernel's que se pueden seleccionar. En la mayoría de los casos no será necesario agregar parámetros al kernel si autodetecta todo el hardware. En caso de que no se agregue parámetro alguno al kernel o que se use un kernel diferente se puede usar el kernel predeterminado (*bare.i*), que es el que se utiliza para la mayoría de los sistemas de escritorio con una plataforma 386 o en caso de que la instalación sea en un quipo portátil (laptop) se puede seleccionar el kernel *bareacpi.i*, una vez seleccionado el kernel carga el sistema utilizando un ramdisk, posteriormente sigue la selección del tipo de teclado (mapa de teclado), se puede presionar sólo enter para elegir el predeterminado, que en este caso es el de tipo 'US'. Después de seleccionar el tipo de teclado para ser ocupado en la instalación viene una parte en la cual se puede probar para ver si en realidad funciona, para salir de esta prueba y aceptar esta configuración se teclea '1' para aceptarla o '2' para seleccionar otro mapa de teclado diferente y finalmente se presiona 'Ok'.

Cuando ya se haya seleccionado el keymap se desplegará una pantalla la cual pide un login, en este momento ya se tiene un sistema funcional con linux (por supuesto con el ramdisk). En este punto cuando se autentica como root (tecleando la cadena root y pulsando la tecla enter) sin contraseña, se tendrán que crear las particiones de tipo 'linux' y una de tipo 'swap', en las cuales se va a instalar el sistema.

Para poder realizar el particionado del disco duro, se puede hacer con dos herramientas, *cfdisk* o *fdisk*. Al ocupar *fdisk*, la forma de utilizarlo es la siguiente:

fdisk /dev/dispositivo

donde *dispositivo*, es el nombre del dispositivo a particionar.

Asumiendo que *hda* es el disco duro en el cual se va a realizar la instalación. Por default fdisk particiona el primer disco duro, en máquinas con disco IDE es /dev/hda y en máquinas con SCSI /dev/sda.

El particionado con *fdisk* se recomienda para expertos, pero en sí es muy sencillo con un poco de conocimientos en el tipo de particiones que pueden ser creadas en un disco. Cuando se ejecuta fdisk sólo se ve una línea donde dice:

Command (m for help):

Quizá esto es lo que desalienta a usuarios nuevos a instalar un sistema Linux, ya que si no es para expertos el uso de esta herramienta, si es para usuarios con un poco de conocimientos.

Dentro de fdisk y después de los dos puntos (:), se pueden escribir una serie de comandos, estos para crear, borrar o asignar nuevas particiones, lo primero que se debe hacer es teclear la letra '*m*' para ver un listado de los comandos que pueden ser usados en esta aplicación, algunos de los comandos más básicos que se utilizan son:

- *a*: Cambiar la bandera de bootable a una partición.
- *d*: Borrar una partición.
- *l*: Listar los tipos de particiones conocidas por fdisk.
- *m*: Imprime los comandos.
- *n*: Agregar una partición.
- *p*: Imprime la tabla de particiones.
- *q*: Salir sin guardar cambios.
- *t*: Cambiar el id a una partición.
- *w*: Escribir la tabla de particiones con los cambios.

Después de conocer los comandos básicos para trabajar con particiones, se puede trabajar sobre ellas para hacer el particionamiento. Primero se creara la partición para "/" y la swap.

La primera será /dev/hda2. Con "*a*" seleccionar el tipo:

- *e*: extendida
- *p*: primaria (1- 4)

Sólo pueden existir 4 particiones primarias, así que si lo que se requiere es crear más de cuatro particiones, lo recomendable es crear una partición extendida y dentro de esta crear más particiones lógicas.

Después de elegir el tipo de partición, se selecciona el sector donde se quiere crear la partición, presionar Enter (recomendable) para que empiece en el siguiente sector libre. Posteriormente se tiene que ingresar el tamaño que puede ser por medio de número, del último cilindro o usando el siguiente esquema:

+tamaño	= Para bloques
+tamañoM	= Para Megabites (recomendable)
+tamañoK	= Para Kilobytes

Cuando se haya creado se puede usar el comando "*p*" para imprimir la tabla de particiones y se verá una partición nueva con un id 83 y sistema Linux lo que indica que es una partición de linux nativa. Puesto que esta partición será usada para swap se le cambia el id system con "*t*" el cual pide el número de la partición a la cual se aplicarán los cambios. Y después el código, en esa parte se puede ver la lista de particiones conocidas, (para swap es el 82) nuevamente se puede pedir que se muestre la tabla de partición y se verá el cambio del "id" system. Ahora lo que sigue es crear la partición para "/" con "*n*" y seguir los pasos anteriores para crearla, una vez creada se le cambia el flag para que sea booteable con "*a*" y se escoge el número de la partición.

Cuando las particiones ya están creadas se puede ejecutar "*w*" y guardar los cambios y enseguida se regresará al prompt para proseguir con la instalación con "*setup*".

El *setup* es el programa de instalación para Slackware el cual tiene un menú con las opciones que guían durante el proceso de instalación cada una de estas opciones puede ser elegida en cualquier momento, aunque lo más recomendable es seguir este proceso el cual llevara (guiara) por cada uno de sus pasos.

Cuando ya se este en el prompt se prosigue a ejecutar *setup* el cual lleva a un programa gráfico con 9 opciones independientes:

- *Help*: Leer el archivo de ayuda para la instalación de Slackware.
- *Keymap*: Remapear el teclado si no se esta usando el US.
- *Addswap*: Configurar la partición(es) swap.
- *Target*: Configurar la partición destino.
- *Source*: Seleccionar el medio fuente.
- *Select*: Seleccionar las categorías de software a instalar.
- *Install*: Instalar el software previamente seleccionado.
- *Configure*: Reconfigurar el sistema Linux.
- *Exit*: Salir del programa de instalación.

El autor del *setup* recomienda seguir el procedimiento que sigue para la instalación. Seleccionar nuevamente el Keymap o mapa de teclado en caso de que no se este usando un teclado US, lo cual si se tiene que hacer por utilizar un Keymap ES.

Addswap: En éste se selecciona la partición *swap* para el sistema, se activa y agrega al archivo /etc/fstab para que se inicie con el sistema (previamente creada). Seleccionar la

partición swap y decir *Si* para que *setup* la instale como swap. Durante esto, *setup* la formatea y checa para verificar si hay sectores defectuosos, después se usa el comando swapon /dev/hdax para activar la partición y después confirma que la partición swap fue configurada y agregada a /etc/fstab.

Target: En este paso se selecciona la partición que será "/" para usarla como el root filesystem. Como en el caso anterior seleccionar /dev/hda3 donde se preguntará si se desea formatear la partición. Nótese que si se formatea se perderá toda la información, así que se deben tomar precauciones, se puede hacer un formateo rápido sin checar bloques defectuosos o un formateo lento, revisando si hay bloques defectuosos. Se recomienda hacer uno lento para hacer una revisión a la partición.

Posteriormente seleccionar el sistema de archivos de la partición, o el sistema de archivos por default es el sistema de archivos ReiserFS.

Una vez que se termina el formateo se pregunta si se quiere formatear alguna otra partición para distribuir el sistema, esto si *setup* detecta más de una partición tipo Linux native. Si hay alguna otra partición se debe seleccionar el tipo de formato, el tipo de sistema de archivos y el punto de montaje.

Cuando se haya terminado el formateo de las particiones se informa de que las particiones fueron agregadas a /etc/fstab, para que puedan ser montadas al inicio del sistema.

En el caso de que existiese alguna partición de tipo FAT o FAT32 *setup* lo mencionará y preguntará si se desea que sea agregada a /etc/fstab para que sea visible desde Linux. En caso de que se acepte esto, seleccionar *Yes*. Y posteriormente se desplegarán las particiones FAT o FAT32 que se desean agregar. Después pide un punto de montaje, para lo cual se sugiere elegir una opción de las recomendadas, por ejemplo, /fatc o /fat-d lo que haría como /mnt/fat-d ya que en /mnt suele montar otros sistemas de archivos, y es donde esta el punto de montaje de los floppy's y CD's. Y finalmente confirmar que fueron agregadas a /etc/fstab.

Source: El siguiente paso será seleccionar la fuente del medio de instalación que puede ser:

1.- CD o DVD
2.- de una partición del disco duro
3.- desde un NFS
4.- desde un directorio previamente montado.

En este caso es desde el CD-ROM y escoger si se desea buscar el dispositivo automático o manual (recomendado manual).

Select: En el cual se seleccionan las categorías generales para la instalación.

Ahora se debe de seleccionar el tipo de instalación (la forma de la elección de paquetes en el proceso de instalación). Si se tiene suficiente espacio libre en el disco duro la opción FULL (completo es más fácil, rápido y la más apropiada para usuarios nuevos). La opción *newbie* proveerá la mayor información pero también es la que consume más tiempo (ya que

presenta los paquetes uno por uno). De otra manera se pueden escoger los paquetes en menús con la opción *expert*.

En este caso se usara la opción FULL, por ser más rápida y fácil, si en algún momento se desean desinstalar paquetes ver el contenido del directorio /var/log/packages/ para ver los paquetes que se tienen instalados en el sistema, y removerlos con *removepkg* o se puede también hacer lo mismo con *pkgtool*.

Si en la instalación se optó por un entorno grafico como GNOME o KDE será solicitado el CD2 de instalación, se debe insertar en la unidad lectora de CD's de ser necesario y seleccionar continue.

El siguiente paso es instalar el kernel, que va a ser usado para arrancar el nuevo sistema (/boot/vmlinuz).

En este caso por ser una instalación inicial seleccionar el CD-ROM, después seleccionar el kernel que se desea instalar, en este caso se seleccionara /cdrom/kernels/bare.i/bzImage que es el mismo con el que se inicio el sistema para IDE después de copiar el kernel preguntará si se desea crear un floppy booteable. Se recomienda crear uno por si algo llegara a salir mal, siempre se puede arrancar con un bootdisk o con el CD 2 del conjunto de CD's de la instalación.

El siguiente paso es elegir en donde esta el modem para hacer un link simbólico a /dev/modem, esto puede ser cambiado después con el programa "modemconfig".

Ahora pregunta si se desea habilitar *hotplug* al inicio del sistema. Hotplug es un subsistema que usa el kernel de Linux para activar hardware que es conectado en un sistema que esta actualmente corriendo. Por ejemplo un dispositivo USB o un CardBus usado en laptops, el subsistema hotplug también puede ser activado al inicio para detectar nuevo hardware, por ejemplo, una nueva tarjeta de sonido, si lo que se quiere es activar al inicio generalmente pulsar opción *Yes*. Cabe mencionar que hotplug puede causar problemas con cierto hardware y con ello una inestabilidad en el sistema. En caso de que ocurra un error este se puede desactivar, pasando como parámetro al kernel al inicio del sistema la cadena nohotplug o hacer /etc/rc.d/rc.hotplug no ejecutable para evitar que se cargue al inicio del sistema.

Instalación de LILO: este es usado para cargar los sistemas operativos, la instalación simple trata de configurar lilo para bootear Linux (y DOS/WINDOWS si es encontrado) para la mayoría de los usuarios avanzados la opción *expert* ofrece más control sobre el proceso de instalación puesto que lilo no trabaja en todos los casos (y puede dañar las particiones si no es correctamente instalado). Hay una tercera opción (segura), la cual es saltar la instalación de lilo por ahora, para instalarlo y configurarlo después con el comando *liloconfig*. Elegir la opción *expert*.

El siguiente paso para instalar lilo es la creación de /etc/lilo.conf y crear los nuevos encabezados y agregar las particiones booteables (aquellas a las que se les cambio el flag

con fdisk) al archivo de configuración lilo.conf, una vez creado y hecho esto se selecciona la opción lilo.

Posteriormente sigue usar la opción framebuffer para la consola, si se desea. NOTA, no todas las tarjetas de video y monitores soportan todos estos modos.

Después seleccionar el destino de lilo. Se recomienda en el MBR el cual rescribirá el MBR (Master Boot Record), en caso de que se tenga otro sistema operativo corriendo en la máquina, por ejemplo un windows, lilo se puede configurar para que también se pueda seleccionar en este otro sistema operativo, en caso de que no se esté seguro de lo que se está haciendo (en lilo) se puede instalar en un floppy y cada vez que se quiera iniciar el sistema Linux se debe bootear con este.

Después de haber seleccionado el lugar de donde se va a instalar el lilo, sólo queda confirmar este paso.

Seleccionar el time out antes de que lilo ejecute una acción y cargue un sistema operativo, si se deja un tiempo de time out lilo booteara el primer sistema listado en el archivo /etc/lilo.conf, por ejemplo se puede seleccionar forever.

Una vez que se crearon los headers, seleccionar Linux: para agregar una partición Linux por ejemplo /dev/hda3 y seleccionar un nombre corto y único para esta partición, este nombre será el identificador para ese sistema operativo, (Linux, por ejemplo). Esto deberá de ser una sola palabra.

Si se tiene una partición de Windows se puede agregar con la opción DOS y posteriormente seleccionar la partición y asignarle un identificador único.

Configuración del Mouse:
Para ello se creará un enlace simbólico a /dev/mouse apuntando al dispositivo mouse predeterminado. Se puede cambiar el enlace /dev/mouse después si el mouse no funciona, también la información relacionada al mouse será usada para configurar el correcto protocolo para GPM y el servidor de mouse de Linux.

- *Ps2*: PS/2 por (la mayoría de laptops)
- *Imps2*: Microsoft ps/2 intellimouse, 2 botones
- *Bare*: Mouse serial compatible con MS
- *Ms*: Mouse serial compatible con MS 3 botones
- *Mman*: Logitech serial mouse y similares
- *ppp*: plug and play
- *usb*: Mouse con conector USB

Después se puede cambiar esta configuración con el comando *mouseconfig*, hacer nota de hotplug y usb devices.

Si no se activó hotplug al inicio y se usan dispositivos USB como teclado y mouse, no se podrá hacer uso de estos dispositivos ya que no se cargan los módulos para dispositivos usb por defecto.

GPM configuración. Si se desea copiar y pegar texto en las consolas virtuales usando el mouse, se debe correr, al inicio /etc/rc.d/rc.gpm con el parámetro que se presente por ejem:

/usr/bin/gpm -m /dev/mouse -t imps2

En caso de aceptarlo se selecciona *Yes* (es recomendable).

El siguiente paso es la configuración de la red, lo cual se vera más adelante en un capitulo dentro de este Módulo, de cualquier forma esta configuración puede ser por medio del comando *netconfig* o editando los archivos respectivos en /etc/rc.d/rc.inet1.conf.

El siguiente paso es configurar los servicios, que serán iniciados al inicio del sistema. Si no se necesitan se pueden deseleccionar para desactivarlos (lo cual podría mejorar la seguridad global del sistema y el rendimiento).

Posteriormente se pueden seleccionar los servicios que se pueden iniciar en el sistema aunque, mientras más servicios "menos seguridad". Para seleccionar los servicios que se desea que se carguen al inicio se puede hacer por medio de la barra espaciadora y presionar enter cuando se ha finalizado la selección de éstos.

Selección del administrador de ventanas predeterminado para las X Windows. Esto definirá el estilo de interfase de usuario grafico que usara el sistema.

GNOME, provee la mayoría de características y la gente con experiencia con Windows o MacOS lo encontrara fácil de usar. Otros administradores de ventanas son más fáciles para los recursos de sistema y ligeros, o proveen otras características únicas:

- xinitrc.gnome: GNU Network Object Enviroment
- XFACE: The cholesterol free desktop enviroment.
- Blackbox: The blackbox windows manager.
- Flubox: The flubox Windows Manager.
- Wmake: Windows Maker
- Awm2: F(>) virtual windows manager
- Fvwm95: Windows manager con un look and feel como windows
- tam: Tab Windows manager.

Posteriormente por razones de seguridad pide una contraseña para el administrador del sistema (root) desde este momento se convierte en un administrador de un sistema.

Después de este punto la instalación se ha completado *SETUP COMPLETE* y lo único que queda por hacer es reiniciar la máquina Ctrl+Alt+Supr y cuando reinicie, se deben de retirar tanto el floppy como el CD de la instalación de la unidad lectora, para que arranque desde el disco duro.

2.3 Inicio y baja del sistema

En Linux existen varias formas de realizar el arranque del sistema, dos de ellas son: desde el disco duro y desde un disco de booteo.

Cuando se realiza el arranque con el disco duro, generalmente se esta llamando a un administrador en especial llamado LILO (LInux LOader). Lilo es un administrador de arranque de propósito general. Al hacerlo de esta forma, el kernel de Linux es el encargado de verificar todo el sistema.

Existe un archivo y un programa que el kernel ejecuta al momento de inicializar los dispositivos. Uno es el programa /sbin/init y el otro es el archivo /etc/inittab. /sbin/init realiza nuevos procesos y restablece ciertos programas al momento de salir. Todo lo que /sbin/init realiza está controlado por el archivo /etc/inittab.

Al igual que el inicio en el momento de querer detener el sistema se puede realizar de varias formas, la primera es apagar el sistema (aunque no es recomendable ya que puede causar daños al sistema de archivos), la segunda y más recomendada es utilizar el comando *shutdown*. Su sintaxis es la siguiente:

/sbin/shutdown [opciones] tiempo [mensaje]

donde:

[mensaje]: es un mensaje, el cual es recibido por todos los usuarios conectados en ese momento en el sistema.
tiempo: es el tiempo en el cuál va a suceder el paro del sistema, por ejemplo "*now*".
[opciones]: son varias y algunas de estas son listadas a continuación

A parte de

Bandera	Descripción
-t seg	Espera seg (número de segundos) para detener los procesos.
-k	En realidad no detiene el sistema sólo envía el mensaje.
-r	Una vez detenidos los procesos reinicia el sistema.
-h	Una vez detenidos todos los procesos apaga el sistema.
-c	Cancela el proceso de detener todos los procesos.

/sbin/shutdown existen otros dos programas que realizan esta tarea, y son, /sbin/reboot y /sbin/halt. En el archivo /var/log/wtmp se guardan los cambios que se realizan con dichos comandos, y ambos, le dicen al kernel si detener o reiniciar el sistema.

Las instrucciones *halt* o *reboot* son llamadas cuando el sistema no se encuentra en el runlevel 0 o 6, el comando *shutdown* se ejecuta en su lugar (con la opción -h o -r). La sintaxis de los comandos es:

a) */sbin/halt [-n][-w][-d][-f]*
b) */sbin/reboot [-n][-w][-d][-f]*

donde:

-*n*: No existe sincronía antes del *halt* o del *reboot*
-*w*: No hay actualización, pero se escribe un registro en el archivo /var/log/wtmp
-*d*: No se escribe al archivo *wtmp*. Esta opción implica -*n*
-*f*: Forza el *halt* o el *reboot*, no se hace una llamada a *shutdown*

NOTA: Cuando se especifica la opción -*f* se hace la llamada con la señal 9, mientras que en el otro caso se realiza la llamada con la señal 15 y en este caso también se llama a las funciones *startup* y *shutdown* respectivamente.

2.3.1 Niveles de Ejecución en Linux (runlevels)

Existen 6 diferentes tipos de niveles de ejecución:

Runlevel	Descripción
0	Paro total del sistema
1	Modo monousuario o administrativo. En este estado se trabaja para el chequeo de los dispositivos.
2	Multiusuario sin NFS, donde no trabajan todos los programas en red.
3	Multiusuario completo.
4	No usado.
5	X11. El sistema X window se ejecuta en este nivel.
6	Reboot. Donde primero se realiza el logout a los usuarios y después se da de baja el sistema.

Por defecto el nivel que se ejecuta al inicio del sistema es el que está definido en el archivo /etc/inittab, un ejemplo de esta línea puede ser:

id:3:initdefault:

donde:

id: Es el identificador de la función a ejecutar.
3: Es el número del runlevel en donde se ejecuta.
initdefault: Esta cadena lo que representa es la acción que se debe tomar a leer este archivo en el momento del inicio del sistema, la cual especifica el runlevel por defecto que debe ser cargado al momento del inicio del sistema.

Para poder cambiar de runlevel se hace uso del comando /sbin/init y su sintaxis es:

/sbin/init 5

Cambiandose así al runlevel 5, iniciando el sistema X window.

2.4 Mantenimiento de cuentas de usuario

En los sistemas Linux existe un administrador, el cual es llamado *root* y este es el encargado de la administración de los usuarios del sistema y dentro de sus tareas está:

Registro de las cuentas, asignación de privilegios, creación y asignación de directorios de trabajo personales, la reasignación de usuarios a determinados grupos, la asignación de cuotas y la baja de cuentas de usuario en caso de ser necesario.

2.4.1 Creación de Cuentas de Usuario

Para la creación de cuentas de usuario, se deben de llevar a cabo varias tareas, y estas son:

a) Editar el archivo */etc/passwd*
b) Crear el directorio de trabajo (home directory)
c) Editar el */etc/group*
d) Copiar los archivos de configuración
e) Asignarle una contraseña (password) a la cuenta
f) Asignar recursos

En Linux estas tareas se llevan a cabo por medio del comando */usr/sbin/adduser*, el cual realiza la edición del archivo */etc/passwd*, le asigna un GUID (Group User ID), un UID (User ID), crea el directorio de trabajo (home directory), pide la contraseña de la cuenta, solicita algunos datos como pueden ser el nombre completo del usuario, teléfono, entre otros y finalmente pide la confirmación para la creación de la cuenta.

A continuación se muestran algunos registros del archivo */etc/passwd*.

```
prueba:x:353:100:Cuenta de Pruebas,,,:/home/prueba:/bin/ksh
gus:x:355:100:Administrador de Base de Datos,,,:/home/gus:/bin/ksh
firebird:x:358:100:Martínez Hernández Ángel,,,:/home/firebird:/bin/ksh
gork:x:396:100:Diaz Jiménez Jose,,,:/home/gork:/bin/ksh
fenix:x:463:100:fenix,55,53259000,56581111:/home/fenix:/bin/ksh
```

Este archivo tiene una sintaxis en particular, tiene 7 campos, cada uno separado por dos puntos (:), donde:

a) el primer campo es el nombre del usuario
b) el segundo, es el password. En los sistemas Linux de antes aquí se guardaba el password cifrado (hash), pero por cuestiones de seguridad ahora se guarda en un archivo el cual sólo el administrador del sistema tiene acceso, el nombre de este archivo es /etc/shadow.
c) el tercer campo es el UID (User ID), o identificador único de usuario.
d) el cuarto campo es el GUID (Group User ID), o identificador del grupo de usuario.
e) el quinto campo es el GECOS, el cual puede o no contener información personal del usuario.
f) el sexto campo es el directorio de trabajo que se le fue asignado.
g) y por último el séptimo campo es el shell por defecto que va ser utilizado por el usuario al momento de loguearse en el sistema o iniciar sesión.

2.4.2 Creación de Grupos

En cualquier plataforma UNIX, se pueden crear grupos de trabajo, los cuales compartirán los mismos recursos. El registro se lleva a través del archivo */etc/group*. Este archivo también cuenta con una sintaxis similar a la del archivo /etc/passwd, campos separados por dos puntos (:), aunque en este archivo son sólo cuatro campos y son:

a) en el primer campo se define el nombre del grupo
b) el segundo campo, se usaba en los Linux anteriores para guardar una contraseña, en la actualidad este campo ya no se utiliza y por ello sólo se pone una '*x*'.

c) el tercer campo contiene el GID (Group ID), identificador único del grupo.
d) y en el último campo se define el o los usuarios que pertenecen a ese grupo, separados por comas (,).

A continuación se muestran algunos registros del archivo */etc/group*.

```
root:x:0:
staff:x:50:
postgres:x:1002:
usuarios:x:300:juan,pedro,carlos,figo,luis
ventas:x:400:gamo,luci,flor
```

Para poder agregar grupos al sistema se hace por medio del comando */usr/sbin/addgroup* y su sintaxis es muy sencilla y es de la siguiente forma:

addgroup nombre_de_grupo

Lo cual hace la inserción de un nuevo registro al archivo /etc/group con un GID diferente al de todos los ya existentes.

2.4.3 Asignación de Contraseñas

La asignación de contraseñas se hace por medio del comando */usr/bin/passwd*, su sintaxis es muy simple, y se utiliza de la siguiente forma (la contraseña puede ser cambiada a cualquier usuario por parte del administrador del sistema y cada usuario puede cambiar su propia contraseña):

/usr/bin/passwd nombre_de_usuario

Lo cual si no se es el administrador del sistema y el mismo usuario cambia su propia contraseña, le pide que ingrese la contraseña que tiene actualmente y posteriormente le pide la nueva y una confirmación. En caso de que sea el administrador del sistema automáticamente le pide que ingrese la nueva contraseña.

NOTA. Es recomendable tener una buena política de contraseñas, puesto que este es un punto que no se debe dejar a la ligera por si se quiere mantener un sistema seguro.

2.4.4 Uso de Recursos

Al momento de crear las cuentas hay que tener en cuenta el espacio en disco que puede ser utilizado por un determinado usuario o un grupo en general, esto es designado por el administrador del sistema y lo hace por medio de una aplicación llamada *quota*, la cual especifica el espacio que puede ser utilizado e incluso los ínodos que puede crear un usuario.

Para poder hacer uso de esta aplicación se deben de crear las cuentas de usuarios en una partición diferente a la del sistema raíz '/', haciendo con esto, un sistema más seguro, puesto que en caso de una actualización del sistema, a los archivos de los usuarios no les pasaría nada y por otra parte, se podrá hacer uso del sistema de *quotas*.

Una de las dos cosas más importantes para utilizar el sistema de quotas, es saber el número de partición en la cual se alojan los directorios de trabajo de los usuarios (en sistemas linux

/home), para lograr esto se hace uso del comando */bin/df*, el cual puede arrojar una salida como la siguiente:

S.ficheros	Bloques de 1K	Usado	Dispon	Uso%	Montado en
/dev/hda2	495746	318861	151282	68%	/
/dev/hda3	405963	33821	351176	9%	/home

Con lo anterior se puede observar que la partición en donde se alojan los directorios de trabajo de los usuarios es /dev/hda3, y para activar el sistema de quotas se edita el archivo */etc/fstab*.

Para poder asignar quotas al los usuarios se agrega la palabra *usrquota* y en caso de que se quiera asignar quotas a un grupo se agrega la palabra *grpquota*.

```
# /etc/fstab: static file system information.
#
# <file system>   <mount point>   <type>   <options>      <dump>        <pass>
        proc            /proc          proc       defaults       0             0
        /dev/hda1       none           swap       sw             0             0
        /dev/hda2       /              ext2       defaults       1             1
        /dev/hda3       /home          ext2       defaults       usrquota      grpquota
```

Posteriormente como administrador del sistema crear un archivo *quota.user*, el cual va a contener los registros de los usuarios, dándole los permisos correspondientes y esto se hace de la siguiente forma:

touch /home/quota.user
chmod 600 /home/quota.user

Lo siguiente es reiniciar el sistema, para que lea este archivo al inicio y comience el comando *quotaon*. Y posteriormente ya se puede asignar el espacio a los usuarios con *edquota*.

edquota -u nombre_de_usuario

```
Quotas for user nombre_de_usuario:
/dev/hda3: blocks in use: 3872, limits(soft=0, hard=0)
inodes in use: 375, limits(soft = 0, hard = 0)
```

Mostrando el número de bloques usados, los límites de quotas inferiores y superiores (soft y hard, respectivamente) y para asignar valores se hace en kbytes (en soft y hard).

Y para comprobar o en caso de que el usuario quiera ver su quota, se hace por medio de:

quota -v nombre_de_usuario

Con la opción *-v* se ve un usuario en específico, pero si lo que se quiere es ver las quotas de todos los usuarios, se hace con *repquota -a*.

Con lo anterior se puede decir que es muy sencillo asignar quotas a los usuarios y poder con ello mantener restringido el que un determinado usuario pueda hacer uso indebido de los recursos de disco.

2.5 Sistema de archivos

Los sistemas de archivos son representados ya sea textual o gráficamente utilizando un gestor de archivos. Los sistemas de archivos más comunes utilizan dispositivos de almacenamiento de datos que permiten el acceso a los datos como una cadena de bloques de un mismo tamaño, a veces llamados sectores, usualmente de 512 bytes de longitud. El software del sistema de archivos es responsable de la organización de estos sectores en archivos y directorios y mantiene un registro de qué sectores pertenecen a qué archivos y cuáles no han sido utilizados.

Generalmente un sistema de archivos tiene directorios que asocian nombres de archivos con archivos, usualmente conectando el nombre de archivo a un índice en una tabla de asignación archivos de algún tipo, como FAT en sistemas de archivos MS-DOS o los ínodos de los sistemas Unix. La estructura de directorios puede ser *plana* o *jerárquica* (ramificada o en árbol).

En sistemas de archivos jerárquicos, generalmente se declara la ubicación precisa de un archivo con una cadena de texto llamada "*ruta*".

Los sistemas de archivos tradicionales proveen métodos para crear, mover y eliminar tanto archivos como directorios, pero carecen de métodos para crear, por ejemplo, enlaces adicionales a un directorio o archivo (enlaces "duros" en Unix) o renombrar enlaces padres (".." en Unix).

Los sistemas de archivos pueden ser clasificados en tres ramas: sistemas de archivos de disco, sistemas de archivos de red y sistemas de archivos de propósito especial:

a) *Sistemas de Archivos de Disco*

Un sistema de archivo de disco está diseñado para el almacenamiento de archivos en una unidad de disco, que puede estar conectada directa o indirectamente a la computadora.
Algunos ejemplos pueden ser: EXT2, EXT3, FAT, NTFS, ISO9660, ReiserFS, Reiser4

b) *Sistemas de Archivos de Red*

Un sistema de archivos de red es un sistema de archivos que accede a sus archivos a través de una red.
Algunos ejemplos pueden ser: CIFS (también conocido como SMB o Samba), NFS

c) *Sistemas de Archivos de Propósito Especial*

Los sistemas de archivos de propósito especial son básicamente aquellos que no caen en ninguna de las dos clasificaciones anteriores.
Algunos ejemplos pueden ser: swap, procfs

2.5.1 EXT2

EXT2 (second extended filesystem o segundo sistema de archivos extendido) fue el sistema de archivos estándar en el sistema operativo GNU/Linux por varios años y continúa siendo ampliamente utilizado. La principal desventaja de EXT2 es que no posee una bitácora, por lo que muchos de sus usuarios están migrando a sistemas de archivos como ReiserFS y su sucesor EXT3.

El EXT2 no usa una FAT, sino una tabla de i-nodos distribuidos en un número determinable de grupos a través de la superficie, lo cual permite balancear la distribución de los bloques de archivos en la superficie a través de dichos grupos para asegurar la mínima fragmentación. El EXT2 tiene un límite máximo de 4GB por tamaño de archivo.

2.5.2 EXT3

EXT3 (third extended filesystem o tercer sistema de archivos extendido) es un sistema de archivos con registro por diario (en inglés journaling) por el solo hecho de tener un "espacio apartado para el buffer de journaling". Este sistema, el cual se encuentra creciendo en popularidad entre usuarios del sistema operativo Linux, a pesar de su menor desempeño y escalabilidad frente a alternativas como ReiserFS, posee la ventaja de permitir migrar del sistema de archivos EXT2 sin necesidad de reformatear el disco.

La única diferencia entre EXT2 y EXT3 es el registro por diario (journaling). Un sistema de archivos EXT3 puede ser montado y usado como un sistema de archivos EXT2.

2.5.3 ReiserFS

A partir de la versión 2.4.1 del núcleo de Linux, ReiserFS se convirtió en el primer sistema de archivos con *journaling* en ser incluido en el núcleo estándar. También es el sistema de archivos por defecto en varias distribuciones, como Slackware, SuSE, Xandros, Yoper, Linspire, Kurumin Linux, FTOSX y Libranet.

Las principales características del sistema de archivos ReiserFS son:

a) El *journaling*, es la mejora a la que se ha dado más publicidad, ya que previene el riesgo de una pérdida o una avería del sistema de archivos.

b) *Reparticionamiento* con el sistema de archivos montado y desmontado. Podemos aumentar el tamaño del sistema de archivos mientras lo tenemos montado y desmontado (online y offline). Para disminuirlo, únicamente se permite estando offline. Esto puede ser bajo un sistema gertos de volúmenes lógicos, como LVM.

c) Comparado con EXT2 y EXT3 en el uso de archivos menores de 4k, ReiserFS es normalmente *más rápido* en un factor de 10 - 15.

2.6 Manejo de la memoria swap

El área de *swap* también es conocida con el nombre de espacio de intercambio, esta área se utiliza cuando la memoria RAM es limitada, por lo que es necesario utilizar memoria virtual o área de swap.

Linux soporta dos tipos de área de swap: particiones de swap (device swap) y archivos de swap (file system swap). Una partición de swap es una partición física de disco, la cual tiene un ID de sistema de archivos establecida con el número 82, que es el número que identifica al área de swap de Linux. Un archivo de swap es grande en un sistema de archivos, que se emplea como área de swap.

2.6.1 Sistema de Archivos SWAP

Para crear un partición de swap se requiere crear antes una partición de disco con el comando *fdisk* (este comando, también nos sirve para verificar el estado actual de la partición del disco) y etiquetarla con el ID 82, para que Linux la reconozca.

El tamaño de la partición depende de la necesidad, de que tanta memoria se requiera como memoria virtual, se tienen que crear múltiples particiones de swap, en caso de que el sistema lo requiera, en el momento de crear la partición swap, se sugiere que esta sea del doble de tamaño de la RAM.
Para crear la partición formal de swap, se debe de tener la partición ordinaria del disco, después se deben seguir ciertos pasos para hacer la partición activa. El primer paso, es crear la partición en forma similar a la de un sistema de archivos, esto es por medio del comando *mkswap*, cuya sintaxis es:

/sbin/mkswap [-c] dispositivo tamañoenbloques

donde:

- *dispositivo*: es el nombre de la partición de intercambio, como /dev/hda2.
- *tamaño en bloques*: es el tamaño del sistema de archivos de destino en bloques (en linux, un bloque tiene el tamaño de 1024 bytes).
- *-c*: mientras se crea el área de swap revisa el sistema de archivos en busca de errores en los bloques. Un ejemplo de el uso de este comando es:
  /sbin/mkswap -c /dev/hda2 19159

Después de haberse creado y preparado la partición, es necesario activarla de modo que el kernel de linux pueda utilizarla. El comando con el cual se hace esto es */sbin/swapon* y su sintaxis es:

/sbin/swapon sistema_de_archivos

donde:

- *sistema_de_archivos*: es la partición que se quiere activar como área swap. Al inicio del sistema linux hace una llamada a /sbin/swapon -a, lo cual hace el montaje de todas las particiones de tipo swap que se encuentren dentro del archivo /etc/fstab.

2.6.2 Archivo de Tipo SWAP

Estos son muy útiles si se necesita expandir el área de swap y no es posible asignar espacio en el disco duro para crear una partición mayor. La configuración de un archivo de swap es casi idéntica a la de la partición. La principal diferencia es que se tiene que crear el archivo antes de ejecutar *mkswap* y *swapon*.

Para crear un archivo de swap se utiliza el comando *dd*. Los principales aspectos antes de crear el archivo son el nombre del archivo y su tamaño en bloques. Por ejemplo si se quisiera crear un archivo llamado swap_archivo en '/' de tipo swap, se hace de la siguiente forma:

dd if=/dev/zero of=/swap_archivo bs=1024 count=10240

y posteriormente:

mkswap /swap_archivo 10240

donde:
of=/swap_archivo especifica el nombre del archivo y *count=10240* bloques o 10 Megabytes.

Antes de ejecutar *swapon*, se debe estar completamente seguro de que el archivo debe estar completamente escrito en el disco y para lograr esto, se hace un simple *sync*. Ahora si se puede activar el archivo con el comando swapon y para ello se hace de la siguiente forma:

/bin/sync

y posteriormente

/sbin/swapon /swap_archivo

En caso de que se tenga que eliminar el archivo de tipo swap, este archivo no debe estar activado y esto se hace de la siguiente forma, usando /sbin/swapoff

swapoff /swap_archivo

En este momento ya se puede borrar el archivo.

2.7 Respaldos

Diferentes tipos de problemas pueden originar la pérdida de datos: eliminación accidental de archivos, una falla de hardware, etc. Por lo que es importante el conocer las formas de respaldar o recuperar esa información.

Respaldos completos o incrementales, son los que copian todos los archivos. Los comandos relativamente simples para la creación de respaldos son: *tar* y *cpio*.

Desde el punto de vista del administrador, el sistema de archivos debe respaldarse de acuerdo con algún proceso automatizado, de preferencia cuando el sistema no se encuentre en uso. Además debe tener un plan de respaldo que satisfaga sus necesidades y que haga posible la restauración de copias recientes de archivos, utilizando una combinación de respaldos completos e incrementales.

Un respaldo completo es como se dijo inicialmente, el que contiene todos los archivos del sistema. Y el respaldo incrementado es el que contiene archivos que han cambiado desde el último respaldo. Estos pueden realizarse a diferentes niveles:

- Nivel 0 Respaldo completo.
- Nivel 1 Incrementado con respecto al último respaldo completo.

- Nivel 2 Incrementado con respecto al último respaldo del nivel 1.

2.7.1 Respaldos con el Comando TAR

Es muy simple el uso de este comando, a continuación se muestra un ejemplo de cómo se puede hacer un respaldo del directorio /home del sistema:

tar cvf /dev/fd0 /home

Nota: en la parte de las opciones, la ***c*** indica la creación de un archivo, la ***f*** especifica el archivo destino, en este caso es la unidad de disco y ***v*** verbose.

En el siguiente ejemplo se muestra el mismo respaldo aunque ahora, es un respaldo de tipo multivolúmen (esto con la opción '***M'***) y la última parte indica un archivo llamado *indice* en el directorio */root*, el cual contiene el listado de los archivos que están siendo respaldados.

tar cvfM /dev/fd0 /home | tee /root/indice

Otra forma de hacer respaldos, no enviándolos a un dispositivo en especial, como es el caso de un cdrom-R o un floppy, puede ser, haciendo esos respaldos en el propio sistema de archivos, incluso comprimiéndolos, esto para ahorrar espacio en disco y se hace de la siguiente forma:

tar cvfz respaldo_home_011205.tar.gz /home

donde: respaldo_home_011205.tar.gz es el nombre del archivo que va a tomar el respaldo, empaquetado y comprimido, es importante que este nombre sea descriptivo en cuanto a lo que se está respaldando incluso también se debe incluir la fecha de la creación y finalmente el directorio o nombre de archivo que se va a respaldar.

2.7.2 Respaldos con el Comando CPIO

El comando *cpio* toma la entrada estándar y la copia a la salida estándar. En el siguiente ejemplo el comando *ls* envía a la salida estándar los nombres de los archivos, *cpio* los toma como entrada y los copia a la salida estándar, sólo que ésta se encuentra redireccionada al dispositivo */dev/fd0*, *cpio* con la opción ***-o*** hace una copia de un archivo en la salida estándar.

ls | cpio -oc > /dev/fd0

La forma más común de usar *cpio* es en conjunción del comando *find*. Ambos forman la mancuerna infalible y son los comandos más portables entre sistemas UNIX.

find /home -depth -print | cpio -ocv -O /tmp/res.cpio

En el ejemplo anterior, lo que se está haciendo es un respaldo de todo el directorio */home* al directorio */tmp* con nombre de archivo *res.cpio*

Para ver el contenido del archivo de respaldo con formato de *cpio*. Es con el siguiente comando:

cpio -icvt -I /tmp/res.cpio

Por ejemplo si lo que se quiere es extraer uno de los respaldos que se tengan guardados en un floppy con formato cpio, se puede hacer de la siguiente forma:

cpio -icvd -I /dev/fd0

o si lo único que se quiere es un listado de los archivos contenidos en ese respaldo con formato cpio, se hace de la siguiente forma:

cpio -it < /dev/fd0 > lista_de_archivos_del_respaldo.txt

NOTA. El hacer un respaldo periódicamente del sistema, puede salvar al administrador de un desastre, ya que el respaldar un sistema debe ser una de las principales políticas de un buen administrador de sistemas.

2.8 Configuración y mantenimiento de la red

Dos de las formas por las cuales se puede configurar la red en distribuciones linux, como es el caso de Slackware, es por medio de un modem o por medio de la configuración de una tarjeta de red o interface de red, por el momento sólo se vera la configuración de la tarjeta de red.

La configuración de esta interface de red se hace por medio del comando */sbin/ifconfig*, pero es más sencillo explicarlo desde un ejemplo:

ifconfig eth0 192.168.1.10 broadcast 192.168.1.255 netmask 255.255.255.0

Lo que se esta haciendo en la línea anterior es, configurar la interface eth0 (primera interface de red), con una IP 192.168.1.10 con una dirección de transmisión (broadcast) 192.168.1255 (lo que incluye toda la subred 192.168.1.0) y una mascara de red (netmask) de 255.255.255.0 (indicando que los tres primeros octetos hacen referencia a la red y el último al número de equipo).

Ahora bien, ya se puede comunicar con los quipos conectados con la red interna, pero ahora lo que se necesita es salir a la red de redes, *Internet*. Y esto se hace por medio del comando */sbin/route*, el cual manipula la tabla de ruteo de las IPs, es decir, que se debe definir la puerta de salida a Internet (gateway) y la sintaxis a seguir, por ejemplo:

/sbin/route add default gw 192.168.1.66

Lo que se esta haciendo en la línea anterior es definir la salida a Internet (gateway) la cual tiene como dirección 192.168.1.66, es decir que los paquetes que salgan del equipo antes de llegar a Internet tienen que pasar por este dispositivo o éste los rutea hacia Internet, esto queda mejor ejemplificado en la figura 2.1.

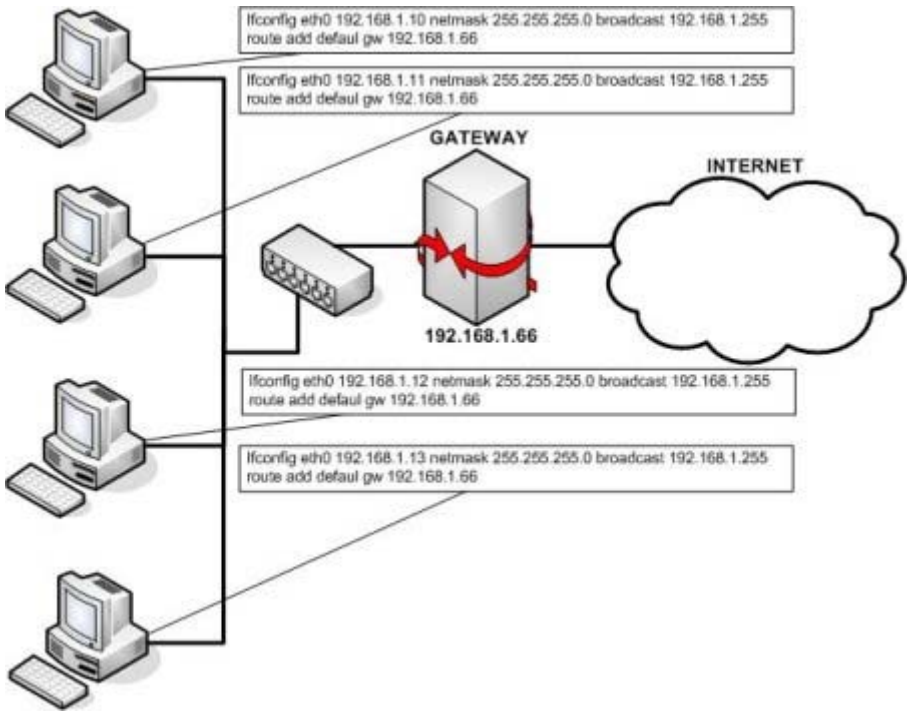


Figura 1. Esquema de ruteo de una red.

Pero esto no es todo, falta agregar algo muy importante y son los DNS's (Domain Name System). Los DNS son una base de datos jerárquica que almacena información asociada a los nombres de dominio en redes como Internet.

La asignación de nombres a direcciones IP es ciertamente la función más conocida de los protocolos DNS. Por ejemplo, si el servidor más conocido por la comunidad de Internet tiene la dirección 64.233.179.104, los usuarios cuando quieren acceder a esta página no lo hacen por medio de la IP, sino por medio de un nombre, el cual es más significativo y menos difícil de recordar, por ejemplo, www.google.com.

En el caso de la UNAM se tienen 4 DNS's, donde 132.248.204.1 y 132.248.10.2 son los dos DNS's primarios de la UNAM. Y estos se deben de agregar al archivo /etc/resolv.conf, esto para que el equipo sea agregado a la tabla de ruteo y tenga un dominio en Internet.
La línea o líneas que se agregan al archivo /etc/resolv.conf quedan de la siguiente forma:

nameserver 132.248.204.1
nameserver 132.248.10.2

Claro esto en el caso de que se este configurando la máquina en una red de la UNAM.

NOTA. Hay que tener en cuenta de que la configuración que se acaba de realizar con los comandos /sbin/ifconfig y /sbin/route no se almacena en ningún archivo, por lo que cuando se reinicie el equipo esta configuración se va a perder y lo único que quedaría almacenado serían los DNS's. Para poder solucionar esto y que cada vez que se reinicie el sistema no se tenga que ingresar estos comandos es por medio del archivo de configuración de la red de

slackware. Y es *rc.inet1*, a continuación se muestra un ejemplo de la sintaxis de este archivo.

```
HOSTNAME=`cat /etc/HOSTNAME`
#definición de la interface de loopback
/sbin/ifconfig lo 127.0.0.1
/sbin/route add -net 127.0.0.0 netmask 255.0.0.0 lo

#definición de la interface de ethernet
IPADDR="192.168.1.10"
NETMASK="255.255.255.0"
NETWORK="192.168.1.0"
BROADCAST="192.168.1.255"
GATEWAY="192.168.1.1"
/sbin/ifconfig eth0 ${IPADDR} broadcast ${BROADCAST} netmask ${NETMASK}
/sbin/route add -net ${NETWORK} netmask ${NETMASK} eth0
if [ ! "$GATEWAY" = "" ]; then
   /sbin/route add default gw ${GATEWAY} netmask 0.0.0.0 metric 1
fi
```

Como se ve es un simple script el cual levanta dos interfaces de red, es lo mismo que se hace con los comandos, la diferencia es que ya no se necesita escribirlos cada vez que se reinicie el sistema.

Existe otro archivo dentro de la configuración de la red en distribuciones slackware y es el *rc.inet2*. *rc.inet1* ofrecen la configuración de la red, pero *rc.inet2* contiene todos los demonios (programas en ejecución) que hagan uso de la red. Uno de esos demonios es *inetd*.

3. Editores para La Creación de Páginas WEB

3.1 Editores para la creación de páginas WEB

Internet se inicio en los años 60's, cuando en los E.U. se buscaba una forma de mantener la comunicación vital del país en el posible caso de una Guerra Nuclear. Este hecho marcó profundamente su evolución, ya que ahora los rasgos fundamentales del proyecto se hallan presentes en lo que hoy se conoce como Internet.

El proyecto contemplaba la eliminación de cualquier *autoridad central*, ya que sería el primer blanco en caso de un ataque. Por ello, se pensó en una red descentralizada y diseñada para operar en situaciones difíciles, el envío de datos debería ser por medio de un mecanismo que pudiera manejar la destrucción parcial de la Red. Por lo que, los mensajes deberían de dividirse en pequeñas porciones de información o *paquetes***,** los cuales contendrían la dirección de destino pero sin especificar una ruta para su arribo, por el contrario, cada paquete buscaría la manera de llegar al destinatario por las rutas disponibles y el destinatario empaquetaría los paquetes individuales para reconstruir el mensaje original.

Fue en Inglaterra donde se experimentó primero con estos conceptos, y así en 1968, el Laboratorio Nacional de Física de la Gran Bretaña estableció la primera red experimental. Al año siguiente, el Pentágono de los E.U. decidió financiar su propio proyecto, y en 1969 se establece la primera red en la Universidad de California (UCLA) y poco después aparecen tres redes adicionales. Nacía así ARPANET (Advanced Research Projects Agency NETwork), antecedente de la actual Internet. Gracias a ARPANET, científicos e investigadores pudieron compartir recursos informáticos en forma remota.

Por lo anterior se puede definir a Internet como una ***red de redes***, es decir, una red que no sólo interconecta computadoras, sino que interconecta redes de computadoras entre sí. Una red de computadoras es un conjunto de máquinas que se comunican a través de algún medio (cable coaxial, líneas telefónicas, fibra óptica, radiofrecuencia, etc.) con el objeto de compartir recursos.

Así, Internet sirve de enlace entre redes más pequeñas y permite ampliar su cobertura al hacerlas parte de una ***red global***, dicha red tiene la característica de que utiliza un protocolo (lenguaje común) que garantiza la intercomunicación de los diferentes participantes, a este ***protocolo,*** que es el lenguaje que utilizan las computadoras al compartir recursos, se le conoce como ***TCP/IP.***

Orígenes del World Wide Web

La *Web* fue desarrollado en 1990 por el Laboratorio Europeo de Partículas Físicas de Suiza, y cinco años después se había convertido en el medio más popular de acceso a Internet.

La Web proporciona una interfaz gráfica para navegar por Internet, en donde el texto y las imágenes se convierten en puentes de comunicación (*hypertext links),* ya que permiten acceder a otras páginas de Internet, no importando si estas páginas sean en la misma computadora o en una diferente. Por ello la Web es global, interactiva y multimedia.

La Web liga múltiples recursos que existen en Internet e integra diferentes tipos de información en una sólida unidad: texto, imágenes, vídeo y audio. Cuando se utiliza la *Web*

se navega sin esfuerzo entre miles de computadoras, aplicaciones e información formateada como archivos y documentos.

Los programas para acceder a la Web son conocidos como *Web browsers,* los más populares, por mencionar algunos son: el *I-Explorer, Mozilla, Opera*, etc.

Todas las páginas Web son creadas con el lenguaje *Hyper Text Markup Languaje - html*. El html se inició como una herramienta para formatear texto, pero en la actualidad se ha desarrollado a tal grado, que permite integrar imágenes, videos, sonidos y formularios que pueden llenarse en línea; en pocas palabras, el html es el lenguaje común de la Web.

3.2 HTML

HTML (Hiper Text Markup Language) es el lenguaje utilizado para representar documentos en la WWW (World Wide Web). Un documento *html* puede incluir elementos multimedia (audio, gráficos, vídeo), además del texto normal y los enlaces (links) que permiten saltar a otras partes del documento o a otro sitio cualquiera de Internet.
Para realizar un documento se necesita un editor de textos para escribir los archivos que lo compondrán y un navegador Web para visualizarlo.

Las etiquetas y sus atributos se pueden escribir en mayúsculas o minúsculas. Una vez que se haya escrito una página, se debe guardar en un archivo con extensión ***.htm*** o ***.html.***

3.2.1 Estructura básica de un documento HTML

Todos los documentos html tienen la estructura básica siguiente:

```
<html>
   <head>
        <title>
          Título del documento
         </title>
   </head>
   <body>
         Contenido del documento
   </body>
</html>
```

Las etiquetas anteriores son indispensables para que se visualice correctamente el documento html, excepto cuando se definen *frames*.

3.2.2 Etiquetas (MARKUP TAGS)

Las etiquetas y atributos se encuentran entre picoparentesis (< >), indicando por un lado la instrucción que será interpretada y por otro los atributos de la misma. Los picoparentesis permiten al navegador reconocer las instrucciones html y no confundirlas con el contenido o información del documento.

Para indicar el final de una instrucción se utiliza la misma etiqueta pero añadiendo una diagonal (/) después del picoparéntesis que abre.

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

Atributos: son modificaciones que pueden presentar las etiquetas y permiten definir diferentes posibilidades de la instrucción html. Estos atributos se definen en la etiqueta de inicio y consisten generalmente en el nombre del atributo y el valor que toma separados por un signo de igual (=). El orden en que se incluyan los atributos es indiferente, no afecta al resultado. Cuando el valor que toma el atributo tiene más de una palabra se debe expresar entre comillas (“ ”).

Las etiquetas se agrupan según su tipo, por su extensión y su utilidad. Estos tipos son: formateo de texto, imágenes, enlaces, listas, tablas, formularios, frames.

Dentro de las etiquetas básicas están *<html></html>* que define el documento, *<head></head>* la cabecera del documento, *<title></title>* el título del documento y *<body></body>* el cuerpo del documento. Aparte de estas, hay etiquetas que se utilizan muy a menudo en los documentos html, por ejemplo:

La etiqueta *<pre></pre>*, dentro de ella el texto es interpretado por el navegador tal y como esta escrito en el código del documento, respetando espacios y saltos de línea, es una definición de texto sin formato o preformateado.

La etiqueta *
***,** indica un salto de línea o rompimiento de línea, es decir un punto y aparte sin separar el párrafo.

La etiqueta *<hr>* (separador horizontal), muestra una línea horizontal de un tamaño determinado. Esta cuenta con algunos atributos como son: *alin* define la alineación del separador, recibe los siguientes valores *left, right y center*, izquierda, derecha y centrado respectivamente; *size* y *width* reciben como valor un número, el cual define el tamaño y grosor del separador respectivamente.

La etiqueta * *, se utiliza para centrar líneas de texto, imágenes o cualquier otro elemento html. Todo lo que se encuentra entre las etiquetas de inicio y fin aparecerá centrado en el navegador.

La etiqueta *<!-- -->***,** se utiliza para incluir comentarios en el código html, los comentarios no son mostrados por el navegador, pero son útiles para realizar anotaciones que indiquen que se esta haciendo en determinada parte del documento.
Ejemplo:

<!-- Este es un comentario sobre un documento html -->

La etiqueta ** permite variar el tamaño, el color, y el tipo de letra del texto, utilizando los atributos *size, bgcolor* y *face*.

- *size* indica el tamaño del texto, su valor puede ser de 1 a 7. El valor 1 es el menor tamaño, el valor 7 el mayor tamaño. El texto normal equivale al tamaño 3.

- *bgcolor= codigo de color***,** define el color del texto en el documento
- *face* define el tipo de letra para el documento. Si el navegador no cuenta con el tipo de letra requerido, el documento será mostrado con el tipo de letra predeterminado del navegador.

*Estilo de fuentes***,** son atributos del texto (negrita, subrayado, etc.) se tienen varias etiquetas. Algunas de ellas no son reconocidas por determinados navegadores de Internet, es por ello que según el navegador que se este utilizando, se verá el resultado correctamente o no.

*Títulos o encabezamientos***,** se emplean al inicio de un documento. Son seis diferentes etiquetas.

Párrafos, se utiliza la etiqueta * * para definir un párrafo, la cual puede contener el atributo *align* y este tomar valores como: *left, right, center* y *justify* para alinear a la izquierda, derecha, centrar o justificar respectivamente. No es necesario cerrar la etiqueta.

3.2.3 Listas

Existen tres tipos de listas, *desordenadas, ordenadas* y de *definición*, así como listas anidadas de cada una de ellas.

Listas desordenadas: Representan los elementos de la lista con una viñeta que antecede a cada uno de ellos. La etiqueta * *, delimita este tipo de listas y *<li>* indica cada uno de los elementos. El tipo de viñetas se puede modificar por medio del atributo *type***,** con los valores: *circle, disc* o *square* (círculo, disco o cuadrado respectivamente).

Ejemplo de una lista desordenada:

Código en html	Así se verá
   Primer elemento Segundo elemento Tercer elemento 	• Primer elemento • Segundo elemento • Tercer elemento

Listas ordenadas: Representan los elementos de la lista, numerando cada uno de ellos según el lugar que ocupan en la lista. La etiqueta * ***,** delimita este tipo de listas y *<li>* indica cada uno de los elementos. Esta etiqueta puede emplear los atributos: *start=num***,** indica que número será el primero en la lista, si no se indica se entiende que comenzará por el número 1 y *type= tipo* indica el tipo de numeración utilizada, si no se indica se entiende que será una lista ordenada numéricamente**.** Los tipos posibles son:

- *1* = Numéricamente. (1, 2, 3, 4, ... etc.)
- *a* = Letras minúsculas. (a, b, c, ... etc.)
- *A* = Letras mayúsculas. (A, B, C, ... etc.)

- *i* = Números romanos en minúsculas. (i, ii, iii,... etc.)
- *I* = Números romanos en mayúsculas. (I, II, III, ... etc.)

Ejemplo de una lista ordenada.

Código en html	Así se verá
<ol type=1> Primer elemento Segundo elemento Tercer elemento Cuarto elemento Quinto elemento 	1. Primer elemento 2. Segundo elemento 3. Tercer elemento 4. Cuarto elemento 5. Quinto elemento

Listas de definición: Muestran los elementos tipo diccionario, es decir, término y definición. Las etiquetas son <*dl*><*/dl*> para la lista, <*dt* > para el termino y <*dd*> para la definición.

Ejemplo de una lista de definición:

Código en html	Así se verá
<dl> <dt >Término 1 <dd>Definición del termino 1 <dt>Término 2 <dd>Definición del termino 2 <dt>Término 3 <dd>Definición del termino 3 </dl>	Término 1 Definición del termino 1 Término 2 Definición del termino 2 Término 3 Definición del termino 3

3.3 Caracteres especiales

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

Estas secuencias de *escape* comienzan con el símbolo *ampersand (&)*, seguido de un texto (siempre en minúsculas) que define el carácter deseado y termina con el símbolo punto y coma (;).

Para expresar algunos de estos símbolos y otros del lenguaje html se usan las siguientes secuencias de escape:

Código en html	Símbolo
&lt;	Signo < (menor que)
&gt;	Signo > (mayor que)
&amp;	Signo & (ampersand)
&quot;	Se mostrará el signo “ (comillas)

Para los acentos, la representación es:

Código en html	Letra	Código en html	Letra
&aacute;	á	&Aacute;	Á
&eacute;	é	&Eacute;	É
&iacute;	í	&Iacute;	Í
&oacute;	ó	&Oacute;	Ó
&uacute;	ú	&Uacute;	Ú

Para la letra ñ se usa la secuencia ***tilde***:

Código en html	Letra
&ntilde;	ñ
&Ntilde;	Ñ

Para el acento circunflejo se usa el literal ***circ*** y para la diéresis se usa el literal ***uml***. Para expresar un carácter por su valor en el código ASCII, se usa el símbolo **#**, seguido de su equivalente numérico.

3.4 Ligas

Una liga o enlace es una conexión desde un recurso de la web a otro (documento, página web o cualquier tipo de archivo). Para definir la liga se utiliza la etiqueta *<a> *, si la liga está indicada por un texto, este aparecerá subrayado y en distinto color al documento, si se trata de una imagen, esta aparecerá con un borde rodeándola. Una liga puede definirse dentro de cualquier elemento html: listas, párrafos de texto, tablas, formularios, etc. Dicha etiqueta debe incluir el atributo *href ="URL"* para especificar el destino de la liga y un atributo (opcional) *name* que asigna un nombre al elemento para acceder al destino de la liga.

Los valores de *href* vienen dados por una *URL*, identificador universal de recursos (Uniform Resource Identifiers) y permite localizar o acceder de forma sencilla a cualquier recurso de la red desde el navegador de la WWW. Estos identificadores deben de ser únicos.

Las URL se utilizará para definir el documento de destino de las ligas, para referenciar los gráficos y cualquier otro archivo que se desee incluir dentro de un documento html. Cada elemento de Internet tendrá una URL que lo defina, ya sea que se encuentre en un servidor de la WWW, FTP, gopher o las News.
El formato de una URL será:

servicio://maquina.dominio/ruta/archivo

El *servicio* será alguno de los de Internet, estos pueden ser: *http*: (HyperText Transport Protocol), *https*: (HyperText Transport Protocol Secure), *ftp*: (File Transfer Protocol), *gopher, news, telnet, mailto*, etc.

La *maquina.dominio* indicará el servidor que proporciona el recurso, en este caso se utilizará el esquema IP para identificar la máquina será el nombre de la máquina y el dominio.

La *ruta* serán los directorios que hay que seguir para encontrar el documento que se desea referenciar.

El *archivo* es el nombre del documento que se desea ver.

En general los enlaces o ligas tienen la siguiente estructura:

* Texto o imagen de la liga*

Anclas

Las *anclas* permiten definir vínculos o saltos internos entre diferentes partes de un mismo documento de forma directa, esto es muy útil cuando el documento es muy largo.

En el documento se marcan las secciones en las que se divide, para ello se utilizara el atributo *name***,** la marca es texto que se coloca en el lugar al que se va acceder (El texto que define a la etiqueta no tendrá ningún tipo de resalte, y sólo se utiliza como punto de destino de la liga). A cada ancla se le dará un nombre distinto que será el que se utilice para referenciar.
La utilidad principal es la creación de índices en documentos largos, al comienzo del documento se especifica el índice con enlaces a las distintas anclas y dentro del documento se indica el comienzo de cada apartado con el ancla que lo define.

Liga a un documento dentro del sistema

Teniendo un directorio con tres páginas *uno.html, dos.html, tres.html*, poner una liga que permita saltar desde uno.html a tres.html. La URL se puede poner como ruta relativa o ruta

absoluta. En la página *uno.html* se pondrá: **, mientras que en la página *tres.html* se pondrá: *Ir a uno*

Un enlace interno (a una página del servidor) se escribiría como:

* enlace *

Liga a un documento externo del sistema

La URL debe escribirse en forma absoluta, tal y como se observa en la ventana de dirección del navegador o poner la IP, las cuales definirán el documento al que se accederá.

Ejemplo:

* Liga utilizando nombre *

* Liga utilizando IP *

Liga usando imágenes

Las imágenes se utilizan como ligas para saltar a la misma imagen con más resolución o salta a un documento con información de la imagen.

Ejemplo:

Liga de una imagen a una dirección de Internet

<img src="img/estrella.gif">

Liga de una imagen a otra imagen

<img src="imagen/luna.gif" >

Liga a una dirección de correo

Desde una liga se puede enviar un correo electrónico. Para ello se utiliza el atributo *mailto*, seguido de la dirección de correo a la cual va a enviarse el correo. Esta liga abre el programa de correo electrónico por defecto para enviar un correo a una dirección determinada.

Ejemplo:

* Envía tus comentarios *

3.5 Imágenes

Para incluir una imagen en un documento se utiliza la etiqueta *<img src=”URL”>* (no tiene etiqueta de cierre), donde URL es la dirección de la imagen. La etiqueta *<img>* tiene atributos como:

- **src =”imagen.jpg”**

Indica el nombre de la imagen que se incluirá en el documento. Se debe escribir la ruta de donde se encuentra la imagen en formato URL. Generalmente es mejor referenciar una imagen que se encuentre en el servidor local, ya que el acceso a imágenes en servidores externos puede ser más lento. Por tanto conviene copiar las imágenes e iconos que se usen al servidor local.

- **alt= “Texto”**

Indica un texto al situar el cursor sobre la imagen, también muestra este mismo texto en caso de que el navegador no muestre la imagen. Se usa para navegadores que no permitan mostrar imágenes, sean de sólo texto o tenga inhabilitado el mostrar imágenes. Se recomienda cuando existan en el documento imágenes usadas como botones, para mostrar un texto en lugar del botón en navegadores que no puedan mostrar gráficos.

- **align**

Indica la alineación del texto que sigue a la imagen con respecto a esta indica la posición de la imagen respecto del texto, con los siguientes valores:
- *top*: alinea el texto con la parte superior de la imagen.
- *middle*: alinea el texto con el punto medio (parte central) de la imagen.
- *bottom*: alinea el texto con la parte inferior de la imagen.
- *left*: alinea la imagen a la izquierda del documento, forzando al texto colocarse en la parte derecha y arriba.
- *rigth*: alinea la imagen a la derecha del documento, forzando al texto colocarse en la parte izquierda y arriba.

- **border = tamaño**

Indica el tamaño del borde de la imagen, si ésta se encuentra dentro de una liga, el borde será del color apropiado para indicarlo, en caso contrario el borde será invisible. Si no se desea mostrar el borde se utiliza la siguiente sintaxis: *border = 0*.

- **height = tamaño, width = tamaño**

Se utilizan para cambiar el tamaño de la imagen de forma que pueda ajustarse como se desee, pudiendo ampliar o disminuir su tamaño.
- *height*: define el alto de la imagen, se especifica en píxeles o en tanto por ciento sobre el tamaño del documento.
- *width*: define el ancho de la imagen, se especifica en píxeles o en tanto por ciento, escalándola al tamaño requerido.

No es necesario indicar el ancho y el alto, se puede especificar sólo uno de ellos, ajustándose el otro a la proporción de la imagen. Es recomendable indicar sólo uno de estos parámetros para que la imagen no se muestre deformada y usar el valor relativo en tanto por ciento para que la imagen guarde siempre una misma proporción con respecto al texto.

3.6 Tablas

Las tablas permiten la representación de datos (texto, imágenes, ligas, otras tablas, etc.) en filas y columnas, con el fin de tener la información ordenada y en determinado sitio del documento. Para crear una tabla se utilizan tres principales etiquetas *<table> </table>, <tr> </tr> y <td> </td> ó <th></th>* en donde:

- **<table></table>**

Etiqueta que define la tabla. En la etiqueta de inicio se definen una serie de atributos opcionales que afectarán a toda la tabla, como los siguientes:

- *border = num***,** dibuja un borde alrededor de la tabla y separa las celdas que presenta. El borde será indicado por un número que especificara el tamaño del borde, por defecto será

0, es decir, no se dibujará ningún borde. Aunque no se dibuje el borde se mantiene el espacio entre los elementos de la tabla.

- *cellspacing = num*, indica el espacio que debe existir entre las distintas celdas de la tabla. Por defecto será 2. Si se indica 0 no habrá ningún espacio entre las celdas.

- *cellpadding = num*, indica el espacio entre el borde de la celda y el contenido de esta, por defecto es 1. Si se indica 0 las celdas aparecerán sin separación.

- *width = num o num%, height =num o num%*, indica el ancho y alto de la tabla respectivamente, se puede indicar como valor absoluto o como porcentaje en función del ancho y alto de la ventana del navegador. Si no se indican los parámetros, el ancho y alto se adecuará al tamaño de los contenidos de las celdas.

- *bgcolor = código de color*, especifica el color de fondo de toda la tabla.

- <*caption*>, especifica el titulo de la tabla, el cual se muestra resaltado en la parte superior externa de la tabla. Siempre se mostrará centrado horizontalmente. Con el atributo *align* se indica si el titulo debe aparecer arriba (*top*) o debajo (*bottom*) de la tabla.

- **<tr> </tr>**

Permite insertar filas en la tabla, la etiqueta de cierre es opcional, si no se escribe se considera que una línea ha acabado cuando comienza otra o cuando acaba la tabla. En caso de tablas anidadas será necesario incluir la etiqueta de cierre. Puede utilizar atributos que afectarán a la fila, por ejemplo:

- *align*, indica la alineación horizontal de los datos dentro de la celda, con los valores *left, center, right***,** alinear a la izquierda, centrar o derecha respectivamente.

- *valing*, indica la alineación vertical de los datos dentro de la celda, con los valores: *top, middle o bottom*, alinea en la parte superior, en el centro o en la parte baja de la celda respectivamente.

- *bgcolor*, indica el color de fondo que tendrá la fila de la tabla.

- **<td> </td> o <th></th>**

Definen las celdas (columnas) que componen la tabla. <*td*> indica una celda normal, y <*th*> indica una celda de *cabecera*, es decir, el contenido será resaltado en negrita o subrayado, centrado y en un tamaño ligeramente superior al normal. Los atributos opcionales para dichas etiquetas son:

- *align, valign*, alineación horizontal y vertical respectivamente, se especifica individualmente para cada una de las celdas.

- *width*, indica el ancho que tendrá la celda (columna) de la tabla. Se especifica en valor absoluto o en tanto por ciento del tamaño de la tabla.

- *bgcolor*, indica el color de fondo que tendrá la columna de la tabla. El color se indica independientemente para cada una de las celdas de la columna.

- *rowspan*, indica el número de filas que ocupará la celda en la misma fila.

- *colspan*, indica el número de columnas que ocupará la celda.

3.7 Formularios

Los formularios son una herramienta que permite dentro de un documento Web solicitar información al visitante, procesarla y responder a ella. En un formulario se pueden solicitar diferentes datos (campos) cada uno de los cuales quedará asociado a una variable. Una vez que se hayan introducido los valores en los campos, el contenido de estos será enviado a la dirección (URL) donde se tenga el programa que procesara las variables.

Para procesar las variables (información) recibidas mediante el formulario se necesita realizar un programa especial externo en algún lenguaje de programación como PHP, ASP, PERL, C++, etc. A este programa externo se le llama *CGI* (Common Gateway Interface). Mediante este programa se pueden incorporar los datos recibidos del formulario a una base de datos, crear un registro, enviar un mensaje automático de respuesta al usuario, validar un pedido, confirmar el acceso a un sistema remoto, etc.

Los formularios están formados por una serie de controles distintos, cada uno de los cuales está asociado a un tipo concreto de datos o una acción predeterminada: botones de envío, borrado de datos, listas de selección, cajas de entrada de texto, etc. La etiqueta que se utiliza para crear un formulario es *<form></form>*. En el interior de la declaración se indican los elementos (variables) de entrada. Sus principales atributos son:

- **action = "programa"**

Indica el programa que va a *procesar* a las variables que se envíen con el formulario, envío de los datos mediante correo electrónico o por medio de una URL usando el método GET. Si se define un programa para el proceso de datos se deberá especificar su ruta relativa, si es mediante correo electrónico definir la dirección (mailto) o la URL completa.

- **method = POST / GET**

Indica el método según como se van a transferir las variables del formulario. Con POST produce la modificación del documento de destino (como en el caso de enviar por correo las variables). Con GET los datos del formulario son encadenados al URL especificado en *action* (utilizando como separador de las variables el signo "?"), utilizando el tipo de codificación especificado en el atributo *enctype*. Este método se utiliza cuando los datos no modifican la base de datos, por ejemplo, al realizar una búsqueda, y los caracteres a enviar tienen que pertenecer obligatoriamente al conjunto ASCII.

Dentro de la definición del formulario se deben incluir algunas etiquetas, las cuales, son las que van a contener los valores introducidos por el cliente, estas etiquetas son:

- **input**: la cual contiene el atributo **type** con valores como:

- *text*: Crea un control de entrada de texto de una línea.

- *password*: Crea una entrada, en la cual los dígitos serán representados por algún carácter que oculte el texto.
- *checkbox*: Las casillas de verificación son interruptores de encendido/apagado que pueden ser conmutados por el usuario. Una casilla de verificación está *marcada* cuando se establece el atributo checked del elemento de control.
- *radio*: Los radiobotones son como las casillas de verificación, excepto en que cuando varios comparten el mismo nombre de control, son mutuamente exclusivos.
- *submit*: Crea un botón de envío el cual manda la información al formulario.
- *image*: Crea un botón de envío gráfico. El valor del atributo *src* especifica el URL de la imagen que decorará el botón.
- *reset*: Crea un botón que reinicializa todos los valores a sus valores iniciales.
- *button*: Se utiliza en los scripts.

- **select**: Muestra una lista de opciones entre las que el usuario elige una. Cada opción se especifica mediante un elemento *option*. El elemento option puede contener varios atributos:

- *selected*: declara dicha opción preseleccionada (valor inicial).
- *value*: específica el valor de dicha opción, si no está, se toma como valor el contenido del elemento.

- **textarea**: Se utiliza para entradas de texto de más de una línea, en lugar del atributo *input*. Puede contener los atributos:

- *rows*: número de líneas de texto visibles.
- *cols*: número de caracteres por línea visibles.

3.7.1 Método GET

El método de consulta depende de los efectos que el formulario tenga en otro navegador, es decir, si el envío va a producir cambios en cualquier documento o programa que no sea el navegador que se este utilizando en ese momento. Si el proceso del formulario es independiente (no produce cambios), el método debe ser GET. Un ejemplo de este tipo de formularios son las consultas a bases de datos, que no tienen efectos laterales visibles.

Para procesar un formulario cuya URL de acción es una URL de tipo *http* y el método es GET, el visor genera una URL que comienza con el de la acción al que se le añade un signo de interrogación (?) y el conjunto de datos codificado con el formato application/x-www-form-urlencoded. Para acceder a la consulta el visor accede al URL de la misma manera que lo hace con los que aparecen en los anclajes.
En algunos casos, la codificación de los datos puede generar un URL extremadamente largo, lo que puede provocar un funcionamiento erróneo con algunos servidores de http antiguos. Por esta razón, algunos formularios que no tienen efectos laterales, se escriben usando el método POST.

3.7.2Método POST.

Para formularios con efectos laterales (como uno que modifique una base de datos) se emplea el método POST.

Para procesar un formulario cuyo URL de acción es de tipo http y el método es POST, el visor gestiona una transacción de tipo POST del protocolo http, usando el URL de la acción y el cuerpo de un mensaje de tipo application/x-www-form-urlencoded. El visor debe presentar la respuesta del http POST de la misma forma que la respuesta obtenida con el método GET.

3.8 Frames

Los frames (marcos o cuadros) permiten dividir la ventana del navegador en varias más pequeñas, de modo que en cada una de ellas se muestra un documento html distinto; cada una de estas ventanas se puede manipular por separado. De esta forma una de las principales utilidades de las frames es la creación de documentos con un índice o una cabecera estática, de esta manera se accede al índice del documento de una manera más rápida y efectiva.

Un documento html que describe frames se define de forma distinta a un documento html normal. Un documento con frames tiene un *head* y un *frameset* en lugar del *body*.

La etiqueta <*frameset*> </*frameset*>**,** especifica cómo se va a dividir la ventana principal. Se pueden incluir varias etiquetas <frameset> anidadas, con el objeto de subdividir una subdivisión. Los elementos que generalmente se definen en el *body* no deben aparecer antes del primer frameset, ya que si se insertan antes serán ignorados por el navegador.

Los atributos básicos de la etiqueta *frameset* son *row* y *cols* en función de si la división de la pantalla se realiza por filas (rows) o columnas (cols), definen el numero de subespacios. En estos se pueden utilizar porcentajes, en caso de no ser especificados, se asumirá un valor del 100%.

Los frames se crean de izquierda a derecha en columnas y de arriba a abajo en filas. Cuando se especifican los dos atributos, los frames se crean de izquierda a derecha en la primera columna, luego en la segunda, etc.

La etiqueta <*noframes*></*noframes*>, se utiliza para incluir una explicación de que el documento sólo es visible en un navegador que soporte frames o incluir una versión del documento a mostrar sin necesidad de utilizar frames.

La etiqueta <*frame*></*frame*> indica las propiedades de cada subventana. Es necesario indicar una etiqueta <frame> para cada subventana creada. Sus atributos son:

- *name*, asigna un nombre al frame, que será usado para enlazar los documentos.
- *src = URL*, indica la localización del documento o archivo que se mostrará en el frame definido, si no se especifica documento alguno se mostrará el frame vacío.
- *scrolling*, indica si el frame tendrá o no una barra de desplazamiento. Sus valores son: *auto* muestra la barra si es necesaria (valor por defecto), *yes* siempre muestra una barra y *no* nunca muestra una barra.
- *frameborder***,** indica si el frame tendrá un borde o no, sus valores son: *1* el navegador dibujará una separación entre los frames, es el valor por defecto, *0* no dibuja la separación entre los frames.

- *target* indica el frame de destino donde se abrirá el documento ya que no es necesario que sea en el frame del documento actual. Por defecto se mostrará en la ventana donde se encuentre el enlace. *target* cuenta con valores que permiten definir destinos distintos a los frames definidos.

4. Administración de Servidores WWW con Linux.

4.1 Servidores WWW

Un servidor WWW es un programa de código libre, robusto, que ofrece servicios dentro del World Wide Web en Internet (proporcionando los recursos que se soliciten), mediante el protocolo HTTP (Hypertext Transfer Protocol).

El protocolo http, basa su operación en la arquitectura "Cliente - Servidor". El cliente http consulta los recursos que el servidor ofrece, mientras que el servidor http es el encargado de publicar recursos electrónicos. El funcionamiento de un servidor Web es muy sencillo y consiste básicamente en enviar al cliente los archivos que este le solicita, es decir:

- El cliente HTTP abre una conexión.
- El server manda un "acknowledge" notificando que se ha abierto una sesión.
- El cliente envía su "request message" solicitando un recurso.
- El servidor responde con "response message" que contiene el recurso solicitado y cierra la conexión.

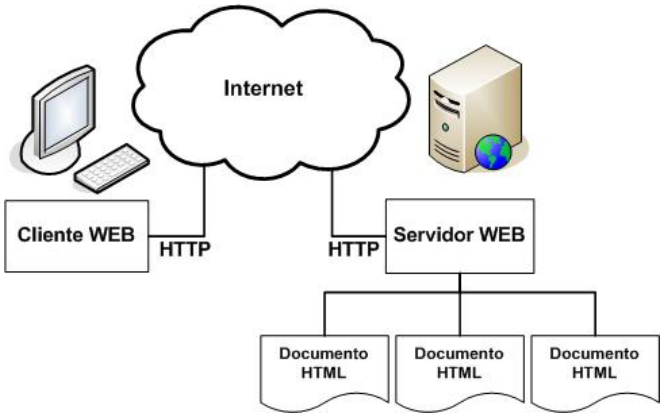


Figura 2. Esquema de peticiones a un servidor Web.

Actividades del cliente

- Solicitar archivos al servidor.
- Interpretar y desplegar código HTML.
- Interpretar Lenguajes de *Scripting* y ejecutarlos.
- Visualizar Imágenes.
- Arrancar aplicaciones externas o *plug-ins.,*
- Controlar algunos aspectos de la presentación del documento (apariencia).

Actividades del servidor

- Enviar archivos al cliente.
- Esperar por peticiones de los clientes.
- Correr programas mediante CGI y enviar respuestas a cliente.
- Establecer conexión a SMBD
- Servir de "*gateways*" a otros servicios: telnet gopher, mail, B.D., ftp,etc.

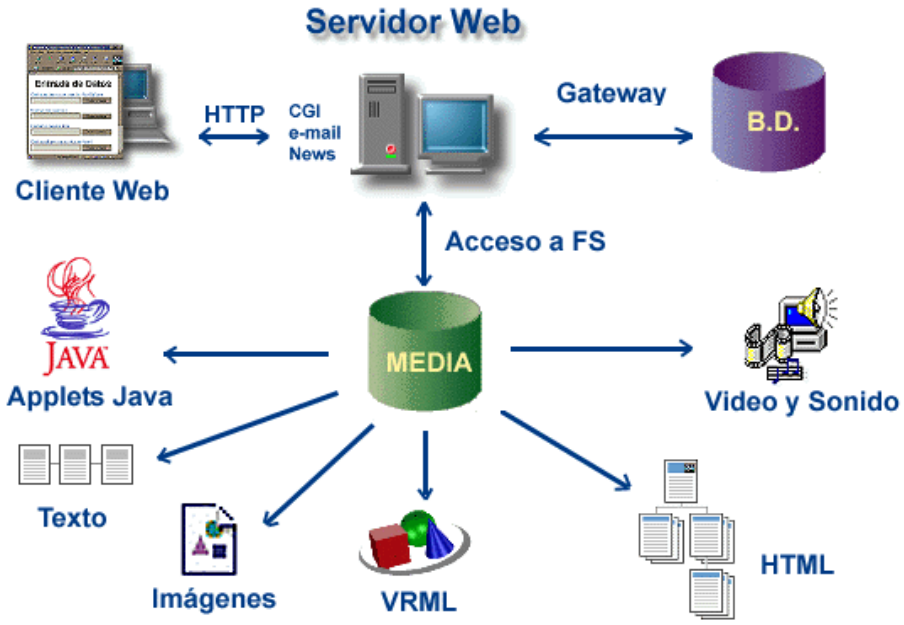


Figura 3. Elementos que intervienen en un servidor Web.

4.2 Instalación del servidor WWW

Apache es robusto, soporta un gran número de transacciones, cuenta con un alto nivel de seguridad, es configurable para diferentes entornos de trabajo y disponible para una gran variedad de plataformas, incluye el código fuente del servidor, es libre y gratuito, además cuenta con soporte para servicio de Proxy, granjas de servidores, Scripting languages integrados como módulos (por ejemplo PHP, mod_perl), accesos restringidos y para SSL. Algunos criterios de selección son:

- Función del Servidor de web
- Expertise de los administradores
- Plataforma de Hardware y Software disponible
- Número de conexiones concurrentes
- Número transacciones por segundo
- Costo computacional por transacción
- Proyección del crecimiento esperado.
- Soporte para la tecnología utilizada para el desarrollo.
- Análisis del retorno de Inversión.
- Robustez/confiabilidad

4.2.1 Revisión de requerimientos de Hardware y Software

Hardware: Los requerimientos para la instalación de Apache varían considerablemente en función de las opciones de configuración que se elija, de los módulos externos que se usen, así como del tráfico esperado. Apache ocupa aproximadamente 10 MB de espacio en disco después de la instalación

Software: Hay versiones de servidores Web para la mayoría de las plataformas de S.O existentes. Aunque necesitara una versión precompilada del servidor Web correspondiente al S.O que este ocupando, además de un Web Browser para probar el funcionamiento del servidor.

Conectividad: La máquina donde se instalara el software servidor, deberá estar conectado a la red y corriendo el protocolo TCP/IP.
La información sobre la Red y el Servidor necesaria para instalar y configurar el servidor es la siguiente:

- Dirección IP del servidor
- Nombre de dominio para el servidor
- Nombre del servidor DNS (opcional)
- Directorio donde van a residir los documentos (Document Root)
- Directorio donde va a residir el servidor (Server Root)
- Directorio donde residirán programas CGI (CGI-dir)

4.2.2 Desempaquetado e instalación

La instalación de apache en Linux, se puede realizar de dos formas: instalarlo a partir de un paquete binario apropiado para cada sistema operativo o compilar el código fuente.

- "The easy way" Por paquetes de alguna distribución, por ejemplo:

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

- "The best way" Compilación

El proceso de compilación es fácil y permite adaptar el servidor Apache a las necesidades requeridas. Para compilar Apache a partir de su código fuente, descargar la versión de Apache más reciente de la sección de descargas del sitio web de Apache http://www.apache.org , el cual contiene varios mirrors.

Después de la descarga, ejecutar lo siguiente:

```
$ tar -zxvf httpd-2.0.53.tar.gz      (Descomprime)
$ cd httpd-2.0.53
$ ./configure                        (Ejecuta el Script configure)
$ make                               (Compila)
# make install                       (Instala)
```

Al ejecutar el script configure (./configure) usa las opciones por defecto (valor por defecto: /usr/local/apache2). Para cambiar las opciones por defecto, *configure* acepta una serie de variables y opciones por la línea de comandos. La opción más importante es *--prefix* que es el directorio en el que Apache va a ser instalado después, porque Apache tiene que ser configurado para el directorio que se especifique para que funcione correctamente. Por ejemplo:

./configure --prefix=/usr/trabajo/apache

Una vez instalado Apache, en el directorio raíz de la instalación, se encuentran los siguientes directorios:

- *bin*: archivos ejecutables del Apache.
- *conf*: archivos de configuración del servidor.
- *error*: archivos con los mensajes de error del servidor, en varios lenguajes.
- *htdocs*: directorio raíz por defecto del servidor (Se guardan las páginas Web).
- *icons*: directorio donde se encuentran los iconos que utiliza el servidor (entre otras cosas para mostrar estructuras de directorios).
- *logs*: directorio donde se almacenan los registros de acceso y errores del servidor.
- *manual*: directorio donde se encuentra el manual del Apache.
- *proxy*: directorio con los archivos de la cache del servidor.

4.3 Configuración del servidor

Cuando se ha instalado apache, hay que configurarlo. Apache incluye por defecto una configuración que arranca el servidor en el puerto TCP, que es el puerto 80, y sirve los archivos del directorio que se han especificado mediante la directiva de configuración denominada *DocumentRoot*. Este archivo de configuración de Apache es el *httpd.conf*, localizado en el subdirectorio *conf* dentro del directorio de instalación. httpd.conf es un archivo de tipo ASCII que contiene las directivas de configuración.

4.3.1 Estructura del archivo de configuración httpd.conf

Apache es administrado por más de 200 directivas las cuales permiten que determinada funcionalidad pueda ser incluida. El administrador controla que directivas estarán disponibles de acuerdo a los módulos con los que se ha compilado Apache.

La configuración de apache se rige mediante directivas y están clasificadas en tres grupos:

- *Global Environment*: administra las directivas generales de operación para apache.
- *Main Server*: La sección Main Server administra las directivas del servidor principal o estándar de apache.
- *Virtual Servers*: administra las directivas donde los mismos procesos de apache soportan diversas Ip's o nombres de dominio.

Directivas de Global Environment del archivo httpd.conf

Los parámetros que se establecen dentro de este grupo son globales para el funcionamiento del servidor, por lo que no admiten estar dentro de ninguna directiva.

User: Define el user ID mediante el cual apache operará.
Valor por Default: User #-1
Se puede usar en: Server config, virtual host

Group: Define el group ID mediante el cual apache operará.
Valor por Default: User #-1

Listen (Port en apache 1.3.x): Define cual es el puerto en que operará el servidor de Web.

Valor por Default: 80

Permite especificar que direcciones IP operará, por defecto todas. Para operar dos direcciones IP distintas, con distinto puerto se utiliza: 80 y 8080

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

DocumentRoot: Define la ruta absoluta donde se almacenarán los archivos html que se desean publicar. Se puede usar en: Server config, virtual host.

Valor por Default: /usr/local/apache2/htdocs
/var/www/htdocs
prefix/htdocs

ServerRoot: Define la ruta absoluta donde los directorios *conf* y *logs* podrán ser encontrados. Se puede usar en: Server config.

Valor por Default: /usr/local/apache

*ServerName***:** especifica el nombre de host y el puerto que el servidor utiliza para identificarse, generalmente se determina automáticamente, pero es recomendable especificarlo explícitamente para que no haya problemas al iniciar el servidor. Si el servidor no tiene un nombre registrado en el DNS, se recomienda poner el número IP.
Sintaxis:

ServerName direccionIP:Puerto p.e. ServerName localhost:80

MinSpareServers: Define cual es el número mínimo de procesos servidores que son mantenidos en spare. Se monitorea el número de conexiones en espera, si este es menor a *MinSpareServers* se levantan los deamons necesarios.

Valor por Default: 5

MaxSpareServers: Define cual es el número máximo de procesos servidores que son mantenidos en spare. Se monitorea el número de conexiones en espera si es mayor a *MaxSpareServers* se eliminan los deamons necesarios, siempre y cuando no estén realizando alguna tarea.

Valor por Dafault: 10

StartServers: Define cual es el número máximo de servidores que se iniciarán cuando se arranca apache.

Valor por Default: 5

MaxClients: Define cual es el número máximo de procesos servidores que como máximo podrán ser levantados o iniciados.

Valor por Default: 150

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

1. Enviar un mensaje de error (default).

2. Enviar un mensaje personalizado.
3. Redirigir a un URL local para manejar el problema o error.
4. Redirigir a un URL externo quien manejará el problema o error.

PidFile: Define el lugar donde se almacenará el archivo que contiene el Process Id para el deamon "padre".

PidFile *file*

Default *file*: logs/httpd.pid

Nota: si este archivo se pierde es común tener problemas cuando arranca o se detiene apache, ver script en /etc.

HostnameLookups: Habilita la resolución de nombres en el DNS cuando se establece una conexión.

HostNameLookup on|off

Default: off

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

<Directory> y <DirectoryMatch>

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

Options: Controla que características del servidor están disponibles para un directorio en particular, es decir la forma en que se mostrará la página al usuario cuando requiere la lista de los archivos de un directorio

Options [+|-] option [[+|-]option] ...

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

Funcionalidades:

- *All*: Todas las opciones se activan excepto MultiViews, (Esta es la opción por default).
- *ExecCGI*: La ejecución de scripts CGIs es permitida.
- *FollowSymLinks*: Las ligas simbólicas son válidas dentro de este directorio. Aún cuando el servidor seguirá las ligas simbólicas este no cambiará de la ruta definida por la directiva <Directory>.
- *Includes*: Permite el uso de Server-side includes.
- *Indexes*: Si un URL mapea a un directorio solicitado y no hay DirectoryIndex (index.html) en este directorio, entonces el servidor devolverá un listado de directorio.

4.4 Ejecución del servidor

Cuando el servidor Apache se ha iniciado y ha completado algunas tareas preliminares, como abrir sus archivos *log*, lanzará varios procesos *hijo*, que hacen el trabajo de escuchar y atender las peticiones de los clientes. El proceso principal, httpd continúa ejecutandose como root, pero los procesos hijo se ejecutan con menores privilegios de usuario.

El comando llamado *apachectl* facilita el arranque y terminación de Apache.

- *./apachectl star*: Inicia el Servidor de Apache
- *./apachectl stop*: Termina el Servidor Apache
- *./apachectl restart*: Reinicia el proceso Apache

Al momento de ejecutar cualquier variación de *apachectl*, se lee el archivo principal de configuración de Apache, *httpd.conf*, ubicado en el directorio:

/usr/local/apache2/conf.

Si todo funciona correctamente durante el arranque, inmediatamente se debe retornar al *prompt* de comandos, lo que indicará que el servidor está levantado y esperando por peticiones de clientes. Si algo esta mal, en ese momento se recibe un mensaje de error. Si el servidor arrancó bien, se puede usar el navegardor (*browser)* y ver la página de prueba que hay en el directorio *DocumentRoot*. La dirección con la cual se consulta la página es: http://localhost/

4.5 Incorporación de módulos

En Apache los módulos son los encargados de agregar funcionalidad adicional incrementando el número de directivas disponibles. Los módulos son una manera de agrupar ciertos funcionamientos para el servidor. Una de las principales razones de emplear módulos en Apache, es que no toda instalación requiere de las mismas funcionalidades.

Para saber cuales son los módulos que se encuentran instalados en el sistema se puede ejecutar el comando *httpd -l*, este comando despliega los módulos instalados. Los módulos se clasifican en dos:

- *Módulos estáticos*: Se agregan al momento de compilar apache.
- *Módulos dinámicos*: Se agregan durante el ambiente productivo, permiten adicionar funcionalidad sin necesidad de recompilar apache.

4.6 Accesos restringidos

Para tener un nivel de seguridad y poder proteger un sitio dentro del servidor, Apache ofrece la funcionalidad ACL (Access Control List), con lo cual es factible determinar las direcciones IPs y los usuarios, así como un login y password por usuario los cuales tendrán acceso sobre recursos específicos del servidor Web.

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

- *Order*, define el orden en que allow o deny serán implementadas
- Allow, define los hosts que tendrán acceso al recurso.

  Allow from all|host|env=env-variable [host|env=env-variable]
- *Deny*, define los hosts que no tendrán acceso al recurso.

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

Allow y Deny permiten definir un conjunto de clientes a los que se les puede permitir o negar el acceso al servicio de Web.

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

- El nombre parcial de un dominio parcial.
  usuarios.com.ar
- Una dirección IP
  200.38.166.1
- La pareja Red y mascara
  10.2.0.0/255.255.255.0
- Un dirección de red definida por Classless Inter-Domain Routing (CIDR)
  10.2.2.110/24

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

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

login y password

AuthUserFile: Indica cual es el archivo que contiene la lista de usuarios y passwords para realizar la autenticación. Es importante que este archivo no sea accesible por los clientes web, pues tendrían acceso a la lista de passwords y podrían atacarla por métodos de diccionario o incluso fuerza bruta para conseguir nuevas claves del sistema.

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

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

AuthGroupFile /dev/null

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

AuthName "Nombre del recurso protegido"

require valid-user: Indica que se requiere de un usuario y una clave de acceso para un recurso determinado.

require valid-user

4.7 Registro de accesos

Apache cuenta con dos archivos donde residen las bitácoras:

access.log: Registra todos los accesos al sitio.

CustomLog filename: Define el nombre del archivo en el cual el servidor registrará los accesos que se presenten. Si *filename* no comienza con un "/" se considera que la ruta depende del ServerRoot.

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

ErrorLog filename: Define el nombre del archivo en el cual el servidor registrará los errores que se presenten (por defecto en la carpeta logs). Si *filename* no comienza con un "/" se considera que la ruta depende del ServerRoot.

Loglevel error: Define el nivel de mensajes de error que serán registrados en la bitácora de "ErrorLog" (nivel de menos a más: emerg, alert, crit, error, warn, notice, info, debug).

LogFormat format string nickname: Define el formato que se utilizará para almacenar los registros.

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

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

Existe una serie de *LogFormat* preestablecido.

- *Common*: Default
- *Refered*: Orientado a llevar un registro de las rutas de los usuarios dentro del sitio.
- *Agent*: Orientado a registrar el nombre de los navegadores de los clientes.
- *Combined*: Combina las características de los anteriores.

4.8 Manejo de sitios virtuales

Los servidores virtuales permiten que un mismo servidor Apache pueda responder a diferentes solicitudes, con lo cual es posible mantener múltiples sitios web con diferentes nombres y/o direcciones IPs. Para ello se configura la directiva *VirtualHost* del servidor Apache y el archivo host del Sistema Operativo.

Un sitio virtual es útil para usar el nombre del dominio real del proyecto como aparecería en Internet, por ejemplo: http://www.uno.com, http://www.dos.com. Porque cuando se tienen varios proyectos, éstos por lo general se guardan en carpetas separadas dentro de algún directorio por ejemplo /home/htdocs/ el cual sería el directorio raíz web, es decir, desde el cual se llama a la página http://localhost y desde ahí se llamarían a los demás proyectos de la siguiente manera:

http://localhost/uno
http://localhost/dos

Para configurar el archivo host
En Linux se encuentra en el directorio /etc/ . Este archivo asigna direcciones IP a los nombres de host, para que al colocar por ejemplo 127.0.0.1 o localhost en el navegador redireccione al mismo contenido. Se puede encontrar lo siguiente en el archivo host: 127.0.0.1 localhost, donde el primer valor es la dirección IP local y el segundo el nombre del host.

En este archivo se colocan las entradas que se necesiten para los proyectos, relacionando la IP local con los nombres de dominio. Por ejemplo:

#localhost
127.0.0.1 localhost
#uno
127.0.0.1 www.uno.com
#dos
127.0.0.1 www.dos.com

Para configurar el archivo httpd.conf de Apache
Dirigirse a la sección de *Virtual Host*, para agregar las siguientes líneas que permitirán que el servidor acepte los nombres de dominio agregados en el archivo host y los relacione con las carpetas de los proyectos.

Virtual Hosts

NameVirtualHost *:80

NameVirtualHost 192.168.40.5
<VirtualHost 192.168.40.5>
   ServerAdmin root@web.ejemplo.com
   ServerName www.uno.com
   DocumentRoot /home/uno/htdocs/
</VirtualHost>

<VirtualHost 192.168.40.5>
   ServerAdmin root@web.ejemplo.com
   ServerName www.dos.com
   DocumentRoot /home/dos/htdocs/
</VirtualHost>

5. Programación con PHP

5.1 Lenguaje PHP

PHP (HyperText Processor) es un lenguaje de scripting que permite la generación dinámica de contenidos en un servidor web. Entre sus principales características se pueden destacar su potencia, alto rendimiento y su facilidad de aprendizaje. Mediante PHP se puede trabajar con formularios, cookies y su mayor potencia es el trabajo con Bases de Datos, así como el trabajo con sockets y distintos protocolos como el manejo de emails.

PHP fue originalmente creado en 1995 por Rasmus Lerdorf quien desarrollo a través de Perl un conjunto de scripts para añadir dinamismo a las páginas web, al conjunto de scripts se les denomino PHP (Personal Home Pages), en 1997 se desarrolla la versión 2.0 denominada PHP/FI (FormsInterpreter); este conjunto de scripts ganó rápidamente popularidad y fue posteriormente rediseñado completamente en 1998 por ZeevSuraski y Andi Gutmans quienes lo llamaron PHP 3.0 y a partir de aquí se le denomina Hypertext Preprocessor. Mas tarde en 1999 Zeev y Andi rediseñan completamente al intérprete, añadiendole más potencia y nuevas funcionalidades (ZendEngine), y se libera la versión 4.0. En el 2004 se crea el motor Zend2.0 y se libera PHP 5.0, se introduce un modelo de orientación a objetos muy similar al de Java.

Las páginas PHP son interpretadas por parte del servidor y como resultado se devuelve al cliente código HTML que es lo único que sabe interpretar el navegador, es decir las páginas PHP dependen del servidor y no del navegador dado que este sólo recibe código HTML.

Cliente	Servidor	Servidor	Servidor	Cliente
Solicita una pagina PHP	Recibe la petición	Procesa la pagina PHP	Devuelve al cliente HTML	Visualiza la pagina el en Browser

Instalación de PHP

Descarga de la fuente

- http://www.php.net/

PHP 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: 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 (como módulo dinámico)

- tar-zxvfphp-4.3.x.tar.gz
- cdphp-4.3.x/
- ./configure --prefix=/usr/local/php4 --with-apxs2=/usr/local/apache2/bin/apxs--with-mysql
- make

- makeinstall
- cpphp.ini-dist/usr/local/php4/lib/php.ini
- Agregar las siguientes líneas al archivo httpd.conf:
  AddType application/x-httpd-php.php.phtml
  AddType application/x-httpd-php-source.phps
- Verificar que exista la linea:
  LoadModulephp4_module modules/libphp4.so
  Crear un archivo llamado info.php en el DocumentRoot de apache /usr/local/apache2/htdocs con el siguiente contenido:
  <? phpinfo(); ?>
- Reiniciar apache.
- Teclear en un navegador: http://localhost/info.php

5.1.1 Sintaxis básica

PHP permite intercalar fragmentos de código dentro de una página HTML, es decir, es posible incluir lenguaje PHP en un código HTML. Para ello es necesario delimitar cuál es el fragmento que contiene código PHP; y este puede delimitarse por cuatro etiquetas diferentes:

- *<?php ?>*
- *<scriptlanguage="php"> </script>*

Las dos anteriores se encuentran siempre disponibles, mientras que las siguientes deben ser configuradas en el archivo php.ini para ser o no ser aceptadas por el intérprete.

- *<? ?> (1)*
- *<% %> (2)*

Nota: Para utilizar (1) directiva short_open_tag=on
Para utilizar (2) directiva aspt_tags=on

Una página PHP funciona de la siguiente manera: El servidor reconoce la extensión correspondiente a la página PHP (php, php3, php4, phtml,...) y antes de enviarla al navegador se encarga de interpretar y ejecutar todo aquello que se encuentra entre las etiquetas de PHP. Así lo que esta fuera de las etiquetas de PHP, lo deja tal y como está.

Para separar las distintas instrucciones, se debe acabar cada instrucción con un punto y coma (;). Para la última instrucción (la que se encuentra antes de la etiqueta que cierra), esto no es necesario.

5.1.2 Variables

Una variable es el nombre que se le da a una posición de la memoria en la cual se almacena información. Es un contenedor de información, en el que se pueden almacenar datos (números enteros, decimales, caracteres, etc.) para, posteriormente recuperarlos cuando se necesiten. En PHP una misma variable puede almacenar distintos tipos de información (almacenar una información supone la pérdida de la anterior).

PHP es un lenguaje débilmente tipado, es decir, las variables no tienen asociada la naturaleza del tipo de información que almacenan. Si se quiere forzar a que una variable tenga un tipo de dato especifico se tiene que utilizar la función *settype()* que sirve para establecer el tipo de una variable.

Una variable se representa mediante el signo *$* seguido del nombre de la variable.

$nombre_de_variable

Un nombre de variable válido empieza con una letra o el carácter de subrayado (_), seguido por una serie letras, números o subrayados. Los nombres de variables en PHP no admiten espacios en blanco, signos de puntuación (acentos, ¡! , ¿?, etc.), ni caracteres especiales(%, ñ, etc.). No es necesario declarar variables en PHP. Las variables se definen por asignación y asumen el tipo del dato que contienen.

NOTA: PHP es *case sensitive*, por lo que diferencia entre minúsculas y mayúsculas.

Variables asignadas por referencia: En PHP también se pueden asignar variables por referencia. En este caso no se les asigna un valor, sino otra variable, de tal modo que las dos variables comparten espacio en memoria para el mismo dato. La notación para asignar por referencia es colocar un ampersand (&) antes del nombre de la variable.

&$nombre_de_variable

Nota: Algo importante a tener en cuenta es que sólo las variables con nombre pueden ser asignadas por referencia.

5.1.3 Tipos de datos

Dependiendo de la información que contenga, una variable puede ser considerada de uno u otro tipo. El tipo de una variable usualmente no es declarado por el programador, es decidido en tiempo de compilación por PHP dependiendo del contexto en el que es usada la variable. PHP soporta 3 tipos de *datos simples* (son aquellos cuyos valores no pueden dividirse en partes menores):

- *Integer*

Números enteros (positivos o negativos); la ocupación en memoria de dichos valores depende de la plataforma, pero en general suele ser de 32 bits, lo que implica que el máximo valor que pueden tomar varía entre -2 billones y 2 billones, se pueden representar en formato decimal, octal o hexadecimal.

- *Float* (sustituye a double desde v 4.2.0).

Números con decimales (El separador decimal es un *punto* y no una coma).

- *String*

Cadenas de caracteres, una cadena está formada por cero o más caracteres encerrados entre “ ” o ‘ ’. Las cadenas se pueden *concatenar* usando el operador punto (.).

Y 2 tipos de *datos compuestos* (se puede asignar cada uno de estos tipos de valor a variables o bien devolverse de funciones sin ningún tipo de limitación):

- *Array*

Es una colección de valores que comparten el mismo nombre y que pueden ser manipulados todos juntos de forma global o de forma individual a través de un índice que los diferencia. En php la estructura que compone el arreglo puede ser de distintos tipos. La primera posición del arreglo tiene el índice 0.

Arrays asociativos: Son arreglos especiales en los que el índice es una valor de tipo string, de modo que cada elemento del arreglo está definida por el par (clave, valor o key, value respectivamente).

- *Object*

Se trata de conjuntos de variables y funciones asociadas. Es una estructura que define características propias (denominadas propiedades) y sus funcionalidades (denominadas métodos). Para inicializar un objeto se utiliza el método *new*, y para acceder a cada uno de sus métodos se utiliza el operador ->

Además hace uso de un tipo *boolean* que no está en la sintaxis del lenguaje. Este es el tipo más simple, normalmente almacenan el resultado de evaluar expresiones lógicas, expresa un valor de verdad. Puede ser *TRUE* o *FALSE*.

Conversión de tipos: Es posible convertir las variables a un tipo específico si se desea. Se escribe entre paréntesis el tipo deseado antes de la variable, que puede ser integer, float, string, boolean, array, object.

Conversión automática de tipos: Cuando se opera con variables de distinto tipo, el intérprete de PHP tiende a homogeneizar sus diferentes tipos en función de la operación que se pretende realizar.

5.1.4 Operadores

Los operadores son componente esencial de cualquier lenguaje de programación. Con ellos se puede asignar, unir, cambiar o comparar valores de datos, cambiar el flujo del programa, etc. Los operadores son símbolos que representan operaciones sobre un valor. Los tipos de operadores que se pueden usar para manejar las variables son:

- *Aritméticos*

Son los operadores básicos para el trabajo con números. Los símbolos + - / * permiten realizar las operaciones de suma, resta, división y multiplicación. El símbolo % permite hallar el resto de una división. Se trata de operadores binarios en todos los casos ya que requieren dos valores para funcionar correctamente.

- *De asignación*

El operador básico de asignación es "=". No tiene el significado del operador de comparación "igual que", lo que realmente significa es que el operando de la izquierda toma el valor de la expresión a la derecha, es decir, se podría traducir por "toma el valor de"**.** En conjunto con el operador básico de asignación, existen *operadores combinados*

para todas las operaciones de aritmetica binaria y de cadenas, que le permiten usar un valor en una expresión y luego definir su valor como el resultado de esa expresión.

- *Cadena*

Se utiliza solo uno que es el operador punto (**.**) el cual une valores y el operador compuesto "**.=**" (asigna un dato al valor anterior) y su función es la de unir dos valores diferentes (concatenar).

- *Incremento y decremento*

Los operadores de incremento (++) permiten aumentar en una unidad el valor de una variable numérica, los de decremento (--) por lo contrario disminuyen la misma variable en una unidad. Estos operadores son operadores unarios por lo que solo reciben un operando al que incrementan o decrementan según el operador.

- *Comparación*

Los operadores de comparación son operadores en su mayoría binarios que nos permiten comparar variables devolviendo un valor booleano a 1 (TRUE) si se cumple la condicion que expresan y a 0 (FALSE) en el caso contrario. Son muy importantes para hacer cualquier tipo de construcción más compleja dentro del lenguaje.

- *A nivel de bit*

Los operadores bit a bit permiten activar o desactivar bits individuales de un entero. Si los parámetros tanto a la izquierda y a la derecha son cadenas, el operador bit a bit trabajará sobre los valores ASCII de los caracteres. Realizar operaciones a nivel de bits, es decir con ceros y unos.

- *Lógicos*

Los operadores de lógica permiten crear expresiones más complejas para evaluar las estructuras de control, permitiendo enlazar varias operaciones de comparación con los diferentes operadores booleanos. Estos operadores son mayoritariamente binarios, por lo que reciben 2 variables, dependiendo de estas variables (de valor 0 o 1) devolverá 0 (FALSE) o 1 (TRUE).

Precedencia de Operadores

Las operaciones se realizan siguiendo un orden de precedencia que PHP asigna a cada operando. Para cambiar estas precedencias se utilizan los paréntesis (), consiguiendo que se realicen primero las operaciones del interior del paréntesis y, después, el resto de operaciones siguiendo en cada caso el mismo orden de precedencia. Si la precedencia de los operadores es la misma, se utiliza una asociación de izquierda a derecha.

5.1.5 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.

- *Por referencia*

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

- *Por defecto*

Son parámetros opcionales en la llamada a las funciones, este tipo de parámetros toma un valor predefinido en caso que no se especifique el argumento en la llamada a la función.

Funciones para variables

- *gettype*(variable): Devuelve el tipo de dato del parámetro.
- *settype*(variable, tipo): Establece el tipo de dato a guardar en una variable, realiza conversiones de tipo de datos.
- *isset*(variable): Devuelve true si una variable ha sido inicializada con un valor, de lo contrario devuelve false.
- *empty*(variable): Devuelve true si la variable no ha sido inicializada, si tiene un valor 0 o si es una cadena vacia, de lo contrario devuelve false.
- *is_int*(variable): True si la variable es entera.
- is_float(variable) : True si la variable es flotante.
- *is_numeric*(variable): True si la variable es un número o una cadena numérica.
- *is_bool*(variable): True si la variable es de tipo lógico.
- *is_array*(variable): True si la variable es de tipo arreglo.
- *is_string*(variable): True si la variable es de tipo cadena.
- *is_object*(variable): True si la variable es de tipo objeto.

Funciones recursivas

Son funciones que en algún punto de su cuerpo se llaman a si mismas, con este tipo de funciones es necesario asegurarse que termine la recursión (condición de parada).

5.1.6 Constantes

Una constante es una variable que mantiene el mismo valor durante toda la ejecución del programa. Una vez definida, no puede ser modificada ni eliminada. Sólo se pueden definir como constantes valores escalares (bolean, integer, flota y string)

Constantes predefinidas de PHP:

- *PHP_VERSION*: Versión de PHP
- *PHP_OS*: Sistema operativo del cliente

- *TRUE*: Verdadero
- *FALSE*: falso
- *_FILE_*: Fichero que se está procesando
- *_LINE_*: Línea del fichero que se está procesando
- *E_ERROR*: Error sin recuperación
- *E_WARNING*: Error recuperable
- *E_PARSE*: Error no recuperable (sintaxis)
- *E_NOTICE*: Puede Tratarse de un error o no. Normalmente permite continuar la ejecución.

Sintaxis de definición de constantes:

define(constante,valor)

5.1.7 Comentarios

Los comentarios son porciones del programa que se ponen sólo para facilitar la comprensión de lectores, lo primero que hace el intérprete de PHP es quitar todos los comentarios del programa. La forma de los comentarios en PHP están heredados de varios lenguajes de programación muy usados en entornos UNIX.

PHP soporta tres tipos de comentarios:

/ Comentario de varias líneas*/*	como en el lenguaje C
//Comentario de una línea	como en el lenguaje C++
#Comentario de una línea	como en el scripting de shell y en el Perl.

Nota: no es recomendable mezclar distintos tipos de comentario en un archivo, sino elegir una sintaxis y quedarse con ella durante todo el documento. No se pueden poner comentarios dentro de otros comentarios.

5.1.8 Estructuras de control

Las estructuras de control o sentencias de control permiten modificar el flujo de ejecución de un programa, permitiendo que la ejecución no tenga que ser secuencial, sino que no permita bifurcar el flujo del programa (*estructuras condicionales*) o que determinado código se ejecuta determinado número de veces (*estructuras cíclicas*).

Estructuras condicionales: Son estructuras que permiten elegir diferentes caminos de ejecución, cuando se cumple una determinada condición. En PHP existen 2: *if* y *switch*.

- *IF*

Sintaxis: if (expresión){ sentencias; }

Expresión debe de ser una expresión lógica, es decir que devuelva verdadero o falso. Las sentencias if se pueden anidar indefinidamente dentro de otras sentencias if, lo cual proporciona una flexibilidad completa para ejecuciones condicionales en las diferentes partes del programa.

- *IF … ELSE*

Sintaxis: if(expresión){ sentencias; }
else{ sentencias; }

else extiende una sentencia if para ejecutar una sentencia en caso de que la expresión en la sentencia if se evalúe como FALSE. La sentencia else se ejecuta solamente si la expresión if se evalúa como FALSE, y si hubiera alguna expresión elseif sólo si se evalua también a FALSE.

- *IF … ELSEIF*

Sintaxis: if(expresión){ sentencias; }
elseif (expresión){ sentencias; }
else{ sentencias; }

elseif, es una combinación de if y else. Ejecutará una expresión alternativa solamente si la expresión condicional elseif se evalúa como TRUE. Puede haber varios elseifs dentro de la misma sentencia if. La primera expresión elseif que se evalúe como TRUE se ejecutara.

- *IF COMPACTO*

Sintaxis: <expresión1> : <expresión2> : <expresión3>

expresión1 es la condición lógica (true or false)
expresión2 se devuelve si es true
expresión3 se devuelve si es false

- *SWITCH*

Se utiliza para comparar una misma variable (o dato) con un conjunto de posibles valores. Un *case* especial es el *default case*; este *case* coincide con todo lo que no coincidan los otros *case*.

```
Sintaxis: switch($variable){
        case valor1:
                sentencias;
                break;
        case valor2:
                sentencias;
                break;
        case valorN:
                sentencias;
                break;
        default:
                sentencias;}
```

Estructuras cíclicas: Se utilizan para ejecutar una o más instrucciones un determinado número de veces, generalmente se utilizan para contar o para recorrer los elementos de un arreglo. En PHP existen 4 tipos: *for, foreach, while* y *do while*.

- *FOR*

Permite realizar un conjunto de instrucciones un determinado número de veces.

Sintaxis: for (inicialización; condición; incremento){
sentencias;}

Inicialización: Se ejecuta tan sólo al iniciar por primera vez la estructura for. En esta se coloca la variable que contara el número de veces que se repite la estructura for.
Condición: Es la condición que se evaluara cada vez que se inicie for. Esta condición es la que determina la duración de la estructura.
Incremento: Sirve para indicar los cambios que se quieren ejecutar en las variables cada vez que se ejecuta dicha estructura.

- *FOREACH*

Se utiliza para recorrer las estructuras de tipo arreglo, obteniendo en cada iteración uno de sus elementos. Tiene 2 sintaxis:

Generalmente para arreglos: foreach (nombre_arreglo $valor){
sentencias; }
Generalmente para arreglos asociativos: foreach (nombre_arreglos $clave=> $valor){
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 única diferencia es que por lo menos se ejecuta una vez, ya que la condición se evalúa al final del ciclo.

Sintaxis: do {sentencias;}
while(condición);

break: Se utiliza para forzar la terminación de un ciclo, o en el caso del switch para que no se sigan evaluando los case.

continue: Se utiliza dentro de los ciclos, cuando se requiere que no se efectúen una serie de instrucciones del ciclo y se quiere pasar a la siguiente iteración; acepta un parámetro opcional, el cual determina cuantos ciclos (niveles) saltara antes de continuar con la ejecución.

5.2 Herramientas elementales

5.2.1 Arreglos

Un array es una colección de valores que comparten el mismo nombre y que pueden ser manipulados todos juntos de forma global o de forma individual a través de un índice que los diferencia. La estructura de un array es la de un conjunto de celdas con un valor cada una, referenciadas por un índice y que tienen un nombre en común.

Una forma muy práctica de almacenar datos es mediante la creación de arrays multidimensionales (tablas). Al proceso de incluir una instrucción dentro de otra se llama anidar y es muy común en programación.

Con los arrays se puede utilizar toda una serie de funciones creadas para ordenarlos por orden alfabético directo o inverso, por claves, contar el número de elementos que componen el array además de poder moverse por dentro de él hacia delante o atrás.

5.2.2 Manejo de cadenas

Las cadenas de caracteres se utilizan delimitadas entre comillas, esta delimitación se puede hacer mediante las comillas simples o dobles, de forma indistinta. Si se necesita utilizar uno de los dos tipos dentro de la cadena, se usará el otro como delimitador. Una de las diferencias entre usar comillas dobles o simples radica en que si se usan las primeras, se puede incluir dentro de la cadena el nombre de una variable, que será sustituido por su contenido.

Además se pueden introducir variables dentro de una cadena lo cual ayuda mucho en el desarrollo de un script.

El tratamiento de cadenas es muy importante, existen bastantes funciones para el manejo de cadenas, las más usadas son:

1. **strlen(cadena)**. Devuelve el número de caracteres de una cadena.
2. **split(separador,cadena)**. Divide una cadena en varias usando un carácter separador.
3. **sprintf(cadena de formato, var1, var2...)**. Formatea una cadena de texto al igual que printf pero el resultado es devuelto como una cadena.
4. **substr(cadena, inicio, longitud)**. Devuelve una subcadena de otra, empezando por inicio y de longitud.
5. **chop(cadena)**. Elimina los saltos de línea y los espacios finales de una cadena.
6. **strpos(cadena1, cadena2)**. Busca la cadena2 dentro de cadena1 indicándonos la posición en la que se encuentra.
7. **str_replace(cadena1, cadena2, texto).** Reemplaza la cadena1 por la cadena2 en el texto.
8. **strtolower(cadena)** y **strtoupper(cadena).** Convierten todos los caracteres de la cadena a minúsculas o mayúsculas.

5.2.3 Inclusión de archivos

Se utilizan principalmente para la definición de librerías comunes a varios scripts, permitiendo la reutilización del código. Funciones para la inclusión de archivos:

- include("nombre_del_archivo"): esta 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.

Las operaciones que se pueden realizar con archivos: *Abrir, Cerrar, Leer, Escribir*.

- *Abrir un archivo*

Sintaxis: *fopen*(nombre_archivo,modo_apertura)

Valores para el parámetro modo apertura:
r :Lectura. El apuntador se coloca al inicio del archivo.
r+:Lectura y escritura. El apuntador se coloca al inicio del archivo.
w : Escritura. Si no existe el archivo se crea, si ya existe se borra su contenido.
w+: Lectura y escritura. Si no existe el archivo se crea, si ya existe se borra su contenido.
a: Escritura. Si no existe el archivo se crea, si ya existe se coloca al final del archivo para añadir datos.
a+: Lectura y escritura. Si no existe el archivo se crea, si ya existe se coloca al final del archivo para añadir datos.

NOTA: fopen() devuelve un apuntador al archivo, a través de este se recorre dicho archivo.

- *Cerrar un archivo*

Es recomendable cerrar un archivo cuando ya no se va a usar para no consumir recursos del sistema, si no se cierra PHP lo hará al terminar de ejecutar el script.
Sintaxis: *fclose*(apuntador)

- *Lectura de archivos*

- *fgetc*(apuntador)
Devuelve un carácter del archivo referenciado por apuntador, si se ha llegado al final del archivo devuelve false.

- *fgets*(apuntador,[total_car_a_leer])
Devuelve una cadena de total de caracteres a leer -1 o de menor longitud si se ha encontrado un cambio de línea que se incluiría en la cadena a devolver o si se ha llegado al final del archivo.

- *fread*(apuntador,[total_car_a_leer])
Igual a fgets() sólo que no deja de leer cuando encuentra un cambio de línea y devuelve total de caracteres a leer.

- *feof*(apuntador)
Devuelve true si se ha llegado al final del archivo.

- *file*(nombre_archivo)

Lee todo el contenido de un archivo y lo devuelve en forma de array: una línea en cada posición del arreglo.

- *readfile*(nombre_archivo)
Lee el contenido de un archivo y lo muestra por la salida estándar.

- *Escritura de archivos*

- *fwrite*(apuntador,cadena)
- *fputs*(apuntador,cadena)
Ambas funciones escriben la cadena pasada como parámetro, devuelven el total de caracteres escritos o false si se produjo algún error.

- *Desplazarse en archivos*

- *rewind*(apuntador)
Sitúa el apuntador de lectura/escritura al principio del archivo.

- *fseek*(apuntador,desp[,desde_pos])
Desplaza al apuntador *desp* posiciones a partir de su posición actual, el tercer parámetro puede tomar los valores SEEK_SET, SEEK_CURy SEEK_END, que le indican que se desplaza n *desp* a partir del principio, posición actual o final del archivo respectivamente (en este caso desp debe ser negativo).

- *ftell*(apuntador)
Devuelve la posición actual del apuntador.

5.3 Common gateway interface (CGI)
CGI "Common Gateway Interface" 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.

Lenguajes de programación CGI: Cualquier lenguaje capaz de tener una salida estándar es susceptible de ser utilizado para desarrollar programas CGI (script CGI), ya sea compilado o interpretado.

5.3.1 Acerca de las variables de ambiente

PHP permite trabajar con dos clases de variables: *locales* y *globales*. Una variable es local cuando se usa dentro del ámbito de una función, es decir, que sólo es accesible dentro del cuerpo de la función. Este tipo de variables existe mientras se está ejecutando la función, y su valor no se conserva una vez que la función ha finalizado su ejecución. Por otro lado, una variable es global cuando es accesible desde cualquier sentencia de un script que no éste dentro de una función.

Para que una función acceda a variables globales, debe estar declarada mediante la sentencia *global* dentro del código de la función. Esta sentencia hace que PHP sepa que se esta haciendo referencia a una variable global y que, por lo tanto, no tiene que crear ninguna variable local nueva. No hay ninguna limitación en cuanto al número de variables globales que puede utilizar una función. En el caso de que el nombre de la variable indicada por global no exista como variable global, ésta se crea y se inicializa con un valor nulo.

También se puede acceder a las variables globales a través de la matriz asociativa *$GLOBALS*. Dicha matriz asociativa almacena todas a las variables globales del script. Cuando se crean variables dentro de las funciones es posible asignar el mismo nombre a dos variables, uno dentro de la función y otro fuera.

Otra característica importante con respecto al ámbito de las variables es el concepto de variable estática (*static*). Una variable estática existe sólo en el ámbito local de una función, y permite que la variable no pierda su valor cuando la ejecución del programa abandona este ámbito. De esta forma cada vez que se vuelve a invocar a la función, todas las variables que se definieron como estáticas recuperan su valor anterior. Para definir una variable estática, en la declaración se antepone la palabra *static* ejemplo:

Stactic $contador = 0;

5.3.2 Lenguajes de programación compilados

Los lenguajes 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 un lenguaje de alto nivel al 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 interpretados, ya que el intérprete debe traducir mientras está en la fase de ejecución.

Ejemplos: Java, C, C++.

- *Ventajas*: Es más veloz que el interpretado ya que no ejecuta el código cada vez que se manda llamar, sino que posee un código objeto que es el que se ejecuta, se

protege la implementación, ya que sólo existe un binario y no se requiere del código fuente.

- *Desventajas*: El ciclo de edición requiere de más tiempo, ya que cada que se hace un cambio se tiene que volver a compilar, no es portable (a excepción de java) y se requiere del código fuente para las modificaciones.

5.3.3 Lenguajes de programación interpretados

Los lenguajes interpretados son aquellos que van siendo codificados por la computadora al tiempo en que se están ejecutando. Es decir, un *intérprete* es un traductor de lenguajes de programación de alto nivel, el cual realiza la operación de compilación paso a paso, los intérpretes ejecutan un programa línea por línea. Para cada sentencia que compone el texto inicial, se realiza una traducción, se ejecuta la sentencia y se vuelve a iniciar el proceso con la sentencia siguiente.

El programa siempre permanece en su forma original *(programa fuente)* y el intérprete proporciona la traducción al momento de ejecutar cada una de las instrucciones. Es decir, el programa será ejecutado sin necesidad de ser codificado antes, y de encontrarse un error la ejecución se detendrá en el comando o acción errónea, el programa puede modificarse sobre la marcha, sin necesidad de volver a comenzar la ejecución. No se genera un archivo binario.

Un intérprete permite utilizar funciones y operadores más potentes, como por ejemplo ejecutar código contenido en una variable en forma de cadenas de caracteres. Usualmente, este tipo de instrucciones es imposible de tratar por medio de compiladores. Los lenguajes que incluyen este tipo de operadores y que, por tanto, exigen un intérprete, se llaman interpretativos. Los lenguajes compilativos, que permiten el uso de un compilador, prescinden de este tipo de operadores.
Ejemplos: ASP, PHP, PERL.

- *Ventajas*: Se puede editar y probar de forma rápida ya que no requiere el proceso de volver a compilar (es más cómodo desarrollar 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 producen un código objeto y recorren el código fuente cada vez que son ejecutados, además de que no protegen la implementación, ya que se requiere el script para su ejecución.

5.3.4 Entrada de datos a partir de formularios

La entrada de datos a partir de formularios es una de las formas de enviar datos a un programa PHP de parte del usuario final, los formularios pueden generarse como HTML puro, sin necesidad de incluir un solo código en PHP, pero se recomienda guardar el archivo con extensión *php* para poder insertar posteriormente código. Ejemplo de un formulario:

```
<FORM action="informacion.php">
¿Cual es tu nombre?: <INPUT TYPE="text" NAME="nombre">
<INPUT TYPE="submit" VALUE=”enviar”>
</FORM>
```

En el ejemplo anterior se tiene un archivo llamado *formulario.php* con un formulario donde se muestra un nombre y un botón para poder enviar los datos escritos por el usuario, estos son enviados a un archivo llamado *informacion.php* definido por la forma con el parámetro *action*. Para poder realizar algo con la información que escribió el usuario (en este caso su nombre) se crea un archivo llamado *informacion.php* que contenga un programa en PHP como el siguiente:

```
<?PHP
print "Hola $nombre";
?>
```

La variable *$nombre,* contiene los datos que ha proporcionado el usuario en el campo "nombre" de tipo texto del formulario anterior, por lo que este ejemplo tiene como salida:

Hola *Nombre*

donde:

Nombre es el nombre que escribió el usuario en el formulario.

5.3.5 Un método diferente de entradas (GET y POST)

Los datos de un formulario se envían mediante el método indicado en el atributo *METHOD* de la etiqueta *FORM*, los métodos posibles son: *GET* y *POST*.

METHOD="GET", envía los datos proporcionados por el usuario a través de la URL (query strins - en la barra de direcciones aparece algo como lo siguiente: ?variablex=234234&otravariable=variable1), se debe tener cuidado al usar este método, ya que el mismo usuario puede modificar directamente la URL alterando los resultados del programa, además este método limita las variables y los nombres juntos hasta 1KB.

METHOD="POST", envía los datos proporcionados por el usuario a través de la entrada estándar *STDIO*, en este método los datos no se verán en la URL, con lo cual es más difícil que el usuario pueda modificar los datos.

El resultado final es el mismo, sólo que con el método *GET* se pueden ver los parámetros pasados ya que están codificados en la URL.

5.5 Algunas funciones útiles

5.5.1 Cookies

Son archivos de texto que almacenan información con la estructura *clave=valor*. Se utilizan para almacenar poca información por periodos prolongados de tiempo. Las cookies son almacenadas en los equipos cliente, para así poder identificar al equipo cliente cuando vuelva. Se manejan a través del arreglo asociativo *$_COOKIE[ ]*. Su tamaño es pequeño, no mayor a 4Kb. Cuando se van a establecer o borrar cookies debe de realizarse antes de mandar cualquier salida al cliente, de lo contrario marca un error.

Realmente las cookies no son más que cadenas de texto que son enviadas desde el servidor al cliente (navegador) y almacenadas en este, luego el navegador envía estas cookies al servidor permitiendo así la identificación del cliente en el servidor.

Creación de Cookies
int setcookie(string nombre [,string valor] [,int caducidad]
[,string ruta] [,string dominio] [,int seguro])
Donde:
- *nombre*: nombre de la cookie
- *valor*: valor que almacenará la cookie en el cliente
- *caducidad*: indica la hora en que se eliminará la cookie, generalmente se utiliza: *time() + N* segundos de duración, para especificar la duración de la cookie.
- *ruta*: subdirectorio en donde tiene valor la cookie (por default es raiz /).
- *dominio*: dominio en donde tiene valor la cookie.
- *seguro (0 o 1)*: Si su valor es 1 la cookie sólo será enviada en el caso que se tenga una conexión segura (https)

Todos los argumentos excepto el *nombre* son opcionales. Las cookies deben ser enviadas antes de cualquier etiqueta de html, por lo tanto se debe realizar la llamada a estas funciones antes de cualquier etiqueta *<HTML> o <HEAD>*. Esta es una restricción de las cookies no de PHP.

Eliminación de Cookies, no se pone ningún valor, se fija la fecha de expiración a 0 (cero) o segundos antes de la fecha actual.

5.5.2 Sesiones

Una sesión es el periodo de tiempo que transcurre desde que el usuario se conecta a un servidor hasta que sale de la aplicación, cierra el navegador o permanece un lapso de tiempo sin interactuar con la aplicación. Se manejan a través del arreglo asociativo *$_SESSION[]*

Las variables de sesión se almacenan en el servidor, cada una está vinculada a una única sesión/usuario a través de un identificador de sesión único. Este valor es almacenado en una *cookie* o bien se propaga en la URL y se hace visible al usuario a través del navegador.

- Para iniciar sesión:
  session_start()
- Para obtener el id de la sesión:
  session_id([string id])
- Para obtener el nombre de la sesión:
  sessionname([string nombre]
- Creación de variables de sesión:
  $_SESSION[‘nombre_variable’]=valor;
- Eliminar variables de sesión:
  unset($_SESSION[‘nombre_variable’]);
- Para cerrar una sesión
  session_destroy();

Esta última destruye todos los datos guardados en una sesión pero no destruye ninguna de las variables globales asociadas a la sesión o a las *cookies*. La función devuelve *TRUE* si se ha destruido la sesión correctamente y *FALSE* si ha habido algún problema al intentarlo.

5.5.3 Correo

Para el envió de correo se debe tener configurado algún servidor de correo en la máquina que esta ejecutando PHP. La sintaxis para el envió de correo es la siguiente:

```
<?php
$address="ejemplo@correo.com";
$subject="Información de Linux";
$body="Conceptos basicos, administracion, servidor Apache\r\entre otros, Pedro";
$mailsend=mail($address,$subject, $body);
print $mailsend;
?>
```

6. Interacción de WWW con bases de datos.

6.1 MySQL

MySQL es un sistema gestor de bases de datos relacionales cliente-servidor SQL. MSQL incluye un servidor SQL, programas cliente para acceder al servidor, herramientas administrativas y una interfaz de programación para escribir los programas. Algunas características de MySQL son: velocidad, facilidad de uso, amplio subconjunto del lenguaje SQL, disponibilidad en gran cantidad de plataformas y sistemas, transacciones y claves foráneas, capacidad de gestión de lenguajes de consulta, portabilidad, conectividad y seguridad, entre otras.

6.1.1 Ventajas y desventajas de acceder las bases de datos vía Web

El Web es un medio para localizar, enviar y recibir información de diversos tipos, utilizando para esto las bases de datos. En el ámbito competitivo, es esencial ver las ventajas que esta vía electrónica proporciona para presentar la información, reduciendo costos y el almacenamiento de la información, y aumentando la rapidez de difusión de la misma.

Internet provee de un formato de presentación dinámico para ofrecer campañas y mejorar negocios, además de que permite acceder a cada sitio alrededor del mundo, con lo cual se incrementa el número de personas a las cuales llega la información.

Pero, no sólo es una vía para hacer negocios, sino también una gran fuente de información, siendo éste uno de los principales propósitos con que fue creada. Una gran porción de dicha información requiere de un manejo especial, y puede ser provista por bases de datos.

Anteriormente, las bases de datos sólo se podían utilizar en el interior de las instituciones o en redes locales, pero actualmente es común encontrar aplicaciones de bases de datos que están migrando hacia la Web, por las ventajas y gran variedad de herramientas que existen en la actualidad para desarrollar estas aplicaciones. La Web permite acceder a bases de datos desde cualquier parte del mundo. Estas ofrecen, a través de la red, un manejo dinámico y una gran flexibilidad de los datos.

Así mismo, el software libre ha incursionado en todas las áreas de la informática por lo que se tiene una amplia gama de herramientas de desarrollo para bases de datos y aplicaciones propias que dan al usuario libertad y facilidad de diseño, desarrollo e implementación.

Con estos propósitos, los usuarios de Internet o Intranet pueden obtener un medio que puede adecuarse a sus necesidades de información, con un costo, inversión de tiempo, y recursos mínimos. Asimismo, las bases de datos serán usadas para permitir el acceso y manejo de la variada información que se encuentra a lo largo de la red.

6.1.2 Base de Datos

Una base de datos es una estructura para almacenar y manejar datos que pertenecen al mismo contexto. La manipulación de los datos involucra la definición de estructuras para el almacenamiento de la información y la provisión de mecanismos para la manipulación de la información. La mayoría de las bases de datos permiten hacer listados, consultas, crear pantallas de visualización de datos, controlar el acceso de los usuarios, etc.

Además un sistema de base de datos debe de tener implementados mecanismos de seguridad que garanticen la integridad de la información, a pesar de caídas del sistema o intentos de accesos no autorizados.

El *sistema de gestión de la base de datos (SGBD)* es una aplicación que permite a los usuarios definir, crear y mantener la base de datos, y proporciona acceso controlado a la misma. El SGBD es la aplicación que interacciona con los usuarios de los programas de aplicación y la base de datos.

Objetivos de las bases de datos

Independencia. La independencia de los datos consiste en la capacidad de modificar el esquema físico o lógico de una base de datos sin tener que realizar cambios en las aplicaciones que se sirven de ella.
Redundancia mínima. Un buen diseño de una base de datos logrará evitar la aparición de información repetida o redundante.
Consistencia. En donde no se ha logrado la redundancia nula, es conveniente vigilar que aquella información que aparece repetida se actualice de forma coherente, es decir, que todos los datos repetidos se actualicen de forma simultánea.
Seguridad. La información almacenada en una base de datos puede llegar a tener un gran valor. El SGBD debe garantizar que esta información se encuentra asegurada frente a usuarios malintencionados, frente a ataques que deseen manipular o destruir la información. Generalmente, el SGBD dispone de un sistema de permisos a usuarios y grupos de usuarios, que permiten otorgar diversas categorías de permisos.
Integridad. Se trata de adoptar las medidas necesarias para garantizar la validez de los datos almacenados.
Respaldo y recuperación. El SGBD debe proporcionar una forma eficiente de realizar copias de seguridad de la información almacenada, y de restaurar a partir de estas copias los datos que se hayan podido perder.
Tiempo de respuesta. Lógicamente, es deseable minimizar el tiempo que el SGBD tarda en dar la información solicitada y en almacenar los cambios realizados.

6.1.3 Modelos de Bases de Datos

Un *modelo de datos* es un conjunto de conceptos que describen la estructura de una base de datos: los datos, las relaciones entre los datos y las restricciones que deben cumplirse sobre los datos.

- ***Jerárquico***

Almacenan la información en una estructura jerárquica. En este modelo los datos se organizan en forma similar a la de un árbol, en donde un *nodo padre* de información puede tener varios *hijos*. El nodo que no tiene padres se le llama *raíz*, y a los nodos que no tienen hijos se les llama *hojas*. Una de las principales limitaciones de este modelo es su incapacidad de representar eficientemente la redundancia de datos.

- ***Reticulares*** (de red)

Este modelo permite que un mismo nodo tenga varios padres, opción que en el modelo anterior no se permite. Ofrece una solución eficiente al problema de redundancia de datos.

- ***Relacionales - MySQL***

Se utiliza para modelar problemas reales y administrar datos dinámicamente. Su estructura principal es la *relación,* es decir, una tabla bidimensional compuesta por filas (registros) y columnas (campos). Cada fila, en terminología relacional se llama *tupla*, representa una entidad que se quiere memorizar en la base de datos. Las características de cada entidad están definidas por las columnas de las relaciones, que se llaman atributos.

En este modelo, el lugar y la forma en que se almacenen los datos no tienen relevancia. La información puede ser recuperada o almacenada mediante "*consultas*" que ofrecen una amplia flexibilidad y poder para administrar la información.

El lenguaje más usado para construir las consultas a bases de datos relacionales es SQL (Structured Query Language o Lenguaje Estructurado de Consultas), estándar implementado por los principales sistemas de gestión de bases de datos relacionales. Está compuesto por comandos, cláusulas y funciones, estos elementos se combinan para crear actualizar y manipular todos los objetos en la base de datos.

De forma gráfica se representa a través de un DER (Diagrama Entidad -Relación). Una relación es la asociación que existe entre un par de entidades.

Tipos de relaciones:

1 : 1 Uno a Uno
1: M Uno a Muchos
M: 1 Muchos a Uno
M : M Muchos a Muchos

Llaves

PK (PRIMARY KEY): La llave primaria 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 brindan la integridad referencial.

- ***Orientadas a objetos***

Este modelo está representado por un conjunto de clases que definen las características y el comportamiento de los objetos que serán almacenados en la base de datos. Con una base de datos orienta a objetos, los objetos almacenados en la base de datos contienen tanto los datos como las operaciones posibles con tales datos. En este modelo se incorporan conceptos como:

Encapsulación - Propiedad que permite ocultar la información al resto de los objetos, impidiendo así accesos incorrectos o conflictos.
Herencia - Propiedad a través de la cual los objetos heredan comportamiento dentro de una jerarquía de clases.
Polimorfismo - Propiedad de una operación mediante la cual puede ser aplicada a distintos tipos de objetos.

6.1.4 Elementos que conforman una base de datos

- *Dato*: Es la unidad básica de almacenamiento para una base de datos.
- *Banco de datos*: Es un conjunto de datos.
- *Campo*: Componente que agrupa a los datos a partir de una misma característica (columnas).
- *Registro*: Componente que clasifica a los datos haciéndolos independientes y diferentes uno de otro.
- *Archivo*: Elemento lógico que almacena datos en formato tabular.
- *Información*: Conjunto de datos procesados y analizados para realizar descripciones.

6.2 Introducción a las etiquetas HTML de los formularios

Un *formulario* se emplea para que el usuario pueda enviar información hacia el servidor. Algunas de sus aplicaciones son: ordenar la compra de un producto, enviar comentarios al administrador de una página Web, obtener datos útiles para el usuario.

Un formulario se compone de los siguientes elementos:

- La etiqueta de inicio **<FORM ACTION='...'>**; en ACTION, se indica el programa ejecutable que procesará el formulario, es decir, la acción que se realiza cuando se procesan los datos.
- El cuerpo del formulario, que contiene texto y una serie de campos de ingreso.
- Botones de envío y borrado del mismo.
- La etiqueta de cierre **</FORM>**

Campos de ingreso de un formulario:

- Introducción de texto en una o varias líneas (**text**).
- Selección de una opción mediante un menú desplegable, o una lista de opciones en la que se puede elegir una y sólo una de ellas (**radio**).
- Selección de varias opciones por medio de los **checkbox**.

Los campos de ingreso tienen asociado un nombre, mediante el atributo **NAME**, con el cual se tiene una manera de referirse a la información ingresada en ese campo. La sintaxis para colocar un campo de ingreso es:

<INPUT TYPE='campo de ingreso' NAME='nombre' VALUE='valor'>
Donde:

- **campo de ingreso**, puede ser: **text, radio, checkbox**.
- **nombre:** es el nombre asignado a la variable de ingreso del dato.
- **valor:** es el valor (predefinido) del campo en cuestión. No se emplea en todos los formularios.

Implementando los campos de ingreso:

a) *text*

<INPUT TYPE='text' NAME='nombre' >

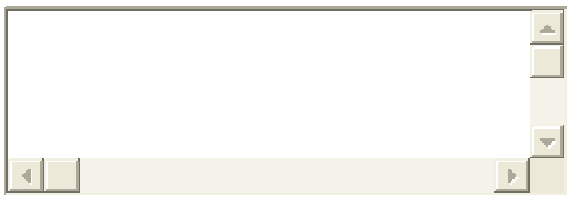
La longitud de este es de 20 caracteres por defecto, pero se puede variar incluyendo en la etiqueta el atributo **SIZE='numero'**.

Para que la información que se introduce no sea legible, es decir que los caracteres que se introducen se vean como asteriscos se utiliza la opción:
<INPUT TYPE='password'>

Nombre:

Cuando se requiere introducir texto grande (algún comentario) en un formulario, se utiliza la siguiente etiqueta:
<TEXTAREA NAME='nombre' ROWS='numero' COLS='numero'>
</TEXTAREA>

ROWS representa el número de filas del texto, y **COLS** la cantidad de columnas.



b) *radio*

<INPUT TYPE='radio' NAME='nombre' VALUE='valor'>

En esta etiqueta todos los botones comparten el mismo nombre, pero tienen distinto valor. Al enviar el formulario, lo recibido por el servidor es el nombre y el valor de la opción elegida.

La opción que aparece seleccionada por defecto se marca mediante el atributo **CHECKED**, pero si no indica ninguna, aparecerá elegida la primera de entre todas las que comparten el mismo nombre.

Elige una opción:

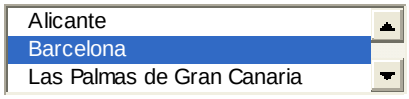
- opción 1
- opción 2
- opción 3

Una manera diferente para que un usuario escoja una opción entre varias es usando un menú desplegable, para ello se utiliza la siguiente etiqueta:
<SELECT NAME='nombre'>
<OPTION>
</SELECT>

Menú desplegable:

Si lo que se desea es mostrar varias opciones a la vez, se añade el atributo **MULTIPLE SIZE='numero'**, donde se especifica el número de opciones visibles para el usuario

¿Elegir opción del menú?:



c) *checkbox*

<INPUT TYPE='checkbox' NAME='nombre'>

Etiqueta que permite al usuario seleccionar varias opciones a la vez, los checkbox son pequeños cuadros que aparecen al costado del texto y se pueden seleccionar o deseleccionar de forma independiente, haciendo click sobre ellos.

 chekbox

En el momento del envío se mandará el nombre del checkbox, y su valor será **on** si la opción está seleccionada. Se puede agregar el atributo **CHECKED** a la opción que se requiera estar seleccionada por defecto.

 chekbox seleccionado por defecto

6.3 Instalación y configuración de la base de datos en linux

La base de datos MySQL, se puede obtener de http://www.mysql.com, o de cualquier otro sitio, es recomendable descargar la última versión estable y en versión fuente (Source Download) en formato tar.gz para compilarla en el propio LINUX que se tiene, de esta forma las herramientas quedaran configuradas con las características especificas del equipo en donde se instalara.

Instalación de mysql

```
tar –zxvf mysql-4.1.7.tar.gz
cd mysql-4.1.7
./configure --prefix=/usr/local/mysql4
make
make install
cp support-files/my-medium.cnf /etc/my.cnf
groupadd mysql
useradd -g mysql mysql
cd /usr/local/mysql4
/bin/mysql_install_db --user=mysql
chown-R root .
chown-R mysql var
chgrp-R mysql .
```

Para arrancar el servidor de MySQL, se emplea un comando como el siguiente:

$bin/mysqld_safe --user=mysql &

El comando anterior arranca el servicio (demonio) de MySQL, el parámetro *--user=mysql* indica que el servicio lo arranca con el usuario mysql y el & manda el proceso a background para que se mantenga funcionando aunque el usuario cierre su terminal. Para confirmar que el servicio de MySQL esta activo utilizar el siguiente comando:

ps –fea | grep mysql

6.3.1 Establecer y terminar conexión con el servidor

Para establecer una conexión con el servidor, se llama a mysql desde la shell tecleando la siguiente sintaxis:

$ mysql options o $mysql –h host_name –u user_name –p

Donde:

- ***-h*** host_name (--host=host_name), es el host del servidor al que se quiere conectar, si el servidor está ejecutándose en la misma máquina en donde se ejecuta mysql, generalmente esta opción puede omitirse.
- ***-u*** user_name (--user=user_name), es el nombre de usuario de MySQL
- ***-p*** (--password), indica a mysql que pida la contraseña del usuario.

$mysql –u Pedro –p
*Enter password: ******
Welcomw to the MySQL monitor. Commands end with; or \g.
Your MySQL connection id is 5 to server version: 4.1.11 –log

Type 'help' for help.

mysql>

Para terminar la sesión, después de establecer conexión con el servidor se utiliza la siguiente sintaxis:

mysql> QUIT
Bye

6.3.2 Introducción al SQL

SQL (Structured Query Language), es un lenguaje de consulta estructurado que permite homogeneizar el acceso a diferentes bases de datos.

SQL es un conjunto de comandos que permiten la definición, consulta y control de los datos de una base de datos relacional. La principal ventaja es que es intuitivo e independiente de la estructura de la base de datos.

- *Emitir consultas*

Para ejecutar los comandos de SQL se emplea la interfaz de comandos de MySQL empleando el comando:

 # bin/mysql

Para introducir una consulta en mysql se teclea el comando, al final de este, se teclea un punto y coma (;) y se pulsa enter. El punto y coma indica a mysql que la consulta esta completa (también se puede usar *\g* para terminar la consulta). Al introducir una consulta no importa si se utilizan mayúsculas, minúsculas o ambas. Las siguientes consultas son equivalentes:

mysql> SELECT VERSION();
mysql> select version();
mysql> SeLeCt vErSiOn();

Después de introducir una consulta, mysql la envía al servidor para que se ejecute. El servidor la procesa y envía los datos a mysql quien visualiza los datos para que el usuario los vea. Algunas consultas son:

- mysql> SELECT NOW();

Indica la fecha y la hora actual (la función NOW(), no es tan útil en si misma, pero puede usarse en expresiones, por ejemplo, para calcular la diferencia entre la fecha actual y otra.

- mysql> SELECT NOW(),
    -> USER(),
    -> VERSION()
    ->;

Como mysql espera un punto y coma antes de enviar la consulta al servidor, no es necesario introducirla en una línea, es decir, se puede introducir en varias líneas. Los prompts que mysql puede mostrar son:

a) mysql>, indica que mysql esta listo para una nueva consulta
b) ->, indica que mysql espera la línea siguiente de una consulta
c) '>, indica que mysql espera la siguiente línea para completar una cadena que comienza con comilla sencilla (').
d) ">, indica que mysql espera la siguiente línea para completar una cadena que comienza con comilla doble (").

- mysql> SELECT NOW(); SELECT USER();

Es posible introducir más de una sentencia por línea, siempre y cuando estén separadas por un punto y coma.

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

La anterior consulta utiliza la opción *\c*, la cual indica que la consulta es cancelada (no se realiza) y devuelve el prompt.

- mysql> STATUS;

Indica el estado y las características del servidor

- mysql> SHOW DATABASES;

Muestra una lista con los nombres de las bases de datos existentes en el servidor

- mysql> SHOW TABLES;

Muestra una lista con los nombres de las tablas existentes en el servidor

- mysql> USE nombredb;

Selecciona la base de datos con la que se va a trabajar. *Nombredb* es el nombre de la base de datos.

- mysql> DESCRIBE nombret;

Muestra la estructura de la tabla de *nombret.*

6.3.2.1 Tipos de datos de MySQL

MySQL reconoce varios tipos de datos o categorías generales cuyos valores se puedan representar.

- *Valores numéricos*

Los números son valores como 83 o 633.91. MySQL reconoce los números especificados como enteros (sin parte fraccionaria) o valores de punto flotante (tienen parte fraccionaria). Los enteros se pueden especificar de forma decimal o hexadecimal.

Un entero consiste en una secuencia de dígitos. Un entero hexadecimal esta compuesto por "0x" seguido de uno o más dígitos hexadecimales (0 hasta 9 y A hasta F). Por ejemplo: "0x0a" es 10 decimal y "0xffff" es 65535 decimal.

Un número de punto flotante consiste en una secuencia de dígitos, un punto decimal y otra secuencia de dígitos. Se puede omitir una de estas dos partes pero no ambas.

MySQL reconoce las notaciones científicas, van indicadas inmediatamente después de un número entero o de punto flotante con una "e" o "E", un signo ("+" o "-") y un exponente entero. Por ejemplo: 1.34E+12 y 47.29e-1. Los números pueden ir precedidos de un signo menos ("-") para indicar un valor negativo.

Cuando se crea una tabla, se especifica el tipo de dato deseado para cada columna, algunos de ellos son:

a. *TinyInt***:** un número entero muy pequeño.
b. *SmallInt***:** número entero pequeño, con o sin signo.
c. *MediumInt***:** número entero mediano, con o sin signo.
d. *Integer*, *Int:* número entero estándar, con o sin signo.
e. *BigInt***:** número entero largo, con o sin signo.
f. *Float***:** número único de precisión de punto flotante.
g. *Double***:** número doble de precisión de punto flotante.
h. *Decimal:* número de punto flotante, representado en una cadena.

- *Valores (caracteres) de cadena*

Las cadenas son valores como "Hola" o "La mesa es grande", se pueden usar las comillas dobles o simples. Se reconocen varias secuencias de escape en las cadenas que se pueden usar para indicar caracteres especiales. Cada secuencia comienza con una barra invertida (\), que significa un escape temporal de las reglas para la interpretación de los caracteres.

Algunos tipos de cadena, para especificar los datos de una columna son:

a. *Char(n):* almacena una cadena de longitud fija.
b. *VarChar(n):* almacena una cadena de longitud variable.

Dentro de los tipos de cadena se encuentran otros dos subtipos, los tipo *Text* y los tipo *BLOB* (Binary large Object). La diferencia entre un tipo y otro es el tratamiento que reciben a la hora de realizar ordenamientos y comparaciones. Mientras que el tipo *text* se ordena sin tener en cuenta las mayúsculas y minúsculas, el tipo *BLOB* se ordena teniéndolas en cuenta. Los tipos *BLOB* se utilizan para almacenar datos binarios como pueden ser archivos.

- *Valores de fecha y hora*

Fechas y horas son valores del tipo "2007-01-9" o "09:30:55", además reconoce la combinación de ambos valores como "2007-01-9 09:30:55". MySQL representa las fechas en el orden año-mes-día.

Para estos valores MySQL proporciona tipos con o sin la hora, sólo la hora y lapsos de tiempo (un tipo especial que permite investigar la última vez que se hicieron cambios en una grabación). Hay también un tipo para representar eficazmente valores de año cuando no se necesite una fecha completa. Estos tipos son:

a. *Date***:** valor de fecha en formato AAAA-MM-DD (año-mes-día)
b. *Time:* valor de hora en formato hh:mm:ss (horas:minutos:segundos)
c. *DateTime***:** valor de fecha y hora en formato AAAA-MM-DD hh:mm:ss
d. *TimeStamp***:** Valor de lapso de tiempo en formato AAAAMMDDhhmmss
e. *Year:* Valor de año en formato AAAA

- *Valor NULL*

NULL se puede considerar como un valor sin tipo. Normalmente quiere decir *sin valor, valor desconocido.* Se pueden insertar valores NULL en las tablas, o recuperarlos para ver si es o no NULL. No se pueden realizar operaciones aritméticas con estos valores.

6.3.2.2 Creando una Base de datos

Para crear una base de datos se utiliza el siguiente comando de SQL:

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

nombredb: es el nombre que se le asigna a la nueva base de datos. Para seleccionar la base de datos con la que se quiere trabajar se utiliza el comando:

mysql> USE nombredb
Database changed
mysql>

En Linux, los nombres de las bases de datos son sensibles al uso de mayúsculas y minúsculas (no como las palabras clave de SQL), por lo tanto se debe tener cuidado de escribir correctamente el nombre de la base de datos o el de una tabla.

La base de datos se crea sólo una vez, pero se debe seleccionar cada vez que se inicia una sesión con mysql. Por ello es recomendable que se indique la base de datos sobre la que se va a trabajar al momento de invocar al monitor de MySQL.

6.3.2.3 Creando tablas

Esta parte es un poco complicada, es decir, la estructura que debe tener la base de datos: qué tablas se necesitan y qué columnas estarán en cada tabla. Para crear una tabla se utiliza el comando:

mysql> CREATE TABLE nombretabla
-> (columna1 tipo_dato, columna2 tipo_dato, columnaN tipo_dato,
-> PRIMARY KEY(columna1, UNIQUE id(columna1));
Query OK, 0 rows affected (0.02 sec)

mysql>

La instrucción genera una tabla de nombre *nombredb,* con las columnas *columna1, columna2* y *columnaN. PRIMARY KEY* indica que la columna1 es la llave primaria de la tabla, *UNIQUE id* indica que no se permiten valores repetidos.

6.3.2.4 Manejo de datos

- *Insertar datos (INSERT)*

El comando para insertar nuevos datos a una base de datos es *INSERT,* se emplea de la siguiente forma:

INSERT INTO nombretabla (columna1, columna2, columna3) VALUES (valor1, valor2, valor3);
La instrucción inserta datos en la tabla *nombretabla,* columna1, columna2 y columna3 son las columnas a las que se les agrega datos y *VALUES* son los valores que toman dichas columnas. Por ejemplo:

INSERT INTO agenda (nombre, apaterno, amaterno, dirección, tel, email) VALUES ("Pedro", "Pérez", "Patiño", "San.Blas", "557979790", "pedro@gmail.com");

- *Actualizar datos (UPDATE)*

El comando para modificar o actualizar datos en una base de datos es *UPDATE*, se emplea de la siguiente forma:

UPDATE nombretabla SET campo1='valor1', campo2='valor2', campo3='valor3' WHERE condición;
La instrucción modifica datos en la tabla *nombretabla,* los *campos* que se listan son los que recibirán los nuevos *valores* para los datos, *WHERE* en este se especifica condición, la cual determina que datos cumplen con la condición para poder ser actualizados.

- *Listar o seleccionar datos (SELECT)*

El comando para listar o seleccionar datos de una base de datos es *SELECT*, se emplea de la siguiente forma:

SELECT campo1, campo2, campo3 FROM nombretabla WHERE condición;

La instrucción lista los datos que cumplen con la condición especificada de la tabla *nombretabla.*

- *Eliminar datos (DELETE)*

El comando para eliminar datos en una base de datos es *DELETE*, se emplea de la siguiente forma:

DELETE FROM nombretabla WHERE condicion;

La instrucción elimina datos de la tabla *nombretabla,* que cumplen la condición de *WHERE*. Es importante especificar la opción *WHERE*, de otra forma se borrarán todos los datos almacenados en la tabla especificada.

6.4 Introducción a los CGIs

CGI (Common Gateway Interface) es una interfaz al servidor Web, que consiste en un conjunto de normas que permiten a una página Web comunicarse y ejecutar con programas externos a la página, redirigiendo su salida al navegador. De esta forma se consigue una interactividad y capacidad de proceso que facilitan el desarrollo de aplicaciones Web.

Con un CGI se puede interactuar con los usuarios que acceden a un sitio en particular. En un nivel teórico, los CGI permiten extender las capacidades del servidor para interpretar las entradas obtenidas del browser (navegador) y regresar la información apropiada de acuerdo a la entrada del usuario. CGI es una interfaz que facilita la escritura de programas para que se comuniquen fácilmente con el servidor.

Actualmente, el uso de los CGIs ha decaído con la llegada de nuevas tecnologías (como el lenguaje Java y los ASPs). Sin embargo, para determinados propósitos son todavía el modo más simple y práctico (ejemplos: contadores, libros de visitas, envío de datos de un formulario, etc.).

El protocolo CGI define una forma estándar para que los programas se comuniquen con el servidor Web. Se puede escribir un programa en cualquier lenguaje de computación que interactúe con el servidor Web. Este programa trabajará con todos los servidores Web que entiendan al protocolo CGI.

La comunicación CGI esta manejada sobre la entrada y salida estándar, lo que significa que si se conoce como imprimir en pantalla y leer datos utilizando el lenguaje de programación, se puede escribir una aplicación sobre el servidor Web.

Sólo se puede utilizar CGIs si el servidor que aloja las páginas permite que se introduzcan y ejecuten CGIs en la zona del usuario (Para saber si se tienen esos permisos, preguntar al encargado de mantener el sitio Web). Esto no es frecuente en servidores de alojamiento gratuitos (comunidades virtuales, portales,...). La razón por la que no siempre se ofrece esta posibilidad es de importancia: la ejecución de un programa no controlado por el propietario de la máquina, en un servidor puede llevar asociados problemas de estabilidad, rendimiento y seguridad.

Si se tiene acceso a un servidor Web con posibilidad de albergar CGI's se deberá programarlo e incluirlo en un directorio dedicado a almacenar este tipo de programas. En general este directorio se llama *cgi-bin*. En lo que se refiere a la programación del CGI, es aconsejable tener experiencia anterior en programación.

6.4.1 Descripción del funcionamiento básico

Para su funcionamiento en general, es necesaria la presencia de dos elementos, una página Web en formato HTML con un formulario donde el usuario introduce sus datos, y un programa CGI en el servidor, que recibe y procesa los datos del usuario.

Los CGIs tienen algunas limitaciones, no es posible ejecutar cualquier programa desde una página Web. Por ejemplo no se puede ir a un servidor Unix cualquiera y ejecutar el interfaz gráfico X-Window. El propio interfaz CGI's impone severas limitaciones, que se derivan por el propio funcionamiento del sistema:

1) En una página Web se incluye la dirección URL de un CGI. En principio puede introducirse una dirección de un CGI, en cualquier lugar donde puede introducirse una dirección de una página Web.
2) Como consecuencia de una acción del usuario el navegador envía al servidor Web una petición de acceso a esa dirección (junto con información adicional).
3) El servidor se da cuenta de que la dirección es un programa CGI y lo ejecuta, usando como entrada al programa la información adicional enviada por el navegador. Al ejecutar un CGI desde una página se le puede mandar toda la información que se quiera, por ejemplo la información que el usuario ha introducido en un formulario.
4) El CGI procesará esta información y devolverá en general una página en formato HTML que crea sobre la marcha con los resultados obtenidos o bien una dirección de una página ya existente o bien una imagen.
5) El servidor recibe la respuesta del CGI y se la envía al navegador, quien la formatea adecuadamente y la muestra al usuario.

Por lo tanto, debe quedar claro que el CGI recibe unos datos, los procesa y devuelve otros datos. El usuario no puede interactuar directamente con el programa CGI. Para obtener ese efecto hay que usar apropiadamente el lenguaje HTML y hacer sucesivas llamadas a uno o varios CGI's.

La forma de interactuar de un programa CGI con el servidor Web, es muy sencilla, toma los datos de entrada por la entrada estándar (como si hubiera un usuario escribiéndolos en un teclado) y envía los datos de salida por la salida estándar (como si se mandaran a una pantalla).

6.4.2 Lenguajes de programación existentes para desarrollar CGIs

Un CGI puede programarse prácticamente en cualquier lenguaje, siempre y cuando dicho lenguaje implemente las funciones adecuadas, impuestas por el interfaz CGI. El único requerimiento externo al lenguaje es que el servidor web (y el sistema operativo) tengan autorizada la ejecución del código de la aplicación CGI. Los lenguajes de programación más usados son los siguientes:

- *Perl:* Lenguaje interpretado más usado para hacer CGI's, debido a las grandes facilidades que ofrece en el tratamiento de texto. Suele estar presente en todas las plataformas tipo Unix, como Linux.
- *Bash (sh) o cualquier otra shell de Unix*: Son usados por su increíble sencillez y facilidad de aprendizaje. Sin embargo tienen fallos de seguridad y no es aconsejable usarlos excepto para CGI's muy sencillos.
- *C o C++:* Son lenguajes muy usados y mucho más potentes, pero tienen inconvenientes como: son más difíciles de aprender, es necesario compilarlos y lleva más tiempo hacer un programa usando estos.

7. Introducción a la Seguridad en Cómputo

7.1 Qué es la Seguridad y Algunos Conceptos

La seguridad informática, generalmente consiste en asegurar que los recursos del sistema de una organización sean utilizados de la forma en que se espera que sean utilizados o que funcionen como deben funcionar.

Se puede entender como seguridad una característica de cualquier sistema (informático o no) que indica que ese sistema está libre de peligro, daño o riesgo. Se entiende como peligro o daño todo aquello que pueda afectar su funcionamiento directo o los resultados que se obtienen de éste. Para la mayoría de los expertos el concepto de seguridad informática es utópico porque no existe un sistema 100% seguro. Para que un sistema se pueda definir como seguro se debe de dotar de cuatro características principales:

- ***Confidencialidad***

La información puede ser accedida únicamente por las personas que tienen autorización para hacerlo. Por ejemplo, cuando se habla de internet (una red de redes), se habla de que la información que se envía a través de esta red puede ser interceptada por otro usuario y se debe asegurar de que sólo el destinatario pueda ser el que la pueda leer.

- ***Integridad***

Se dice que se esta totalmente seguro de que la información no ha sido borrada, copiada o alterada, no sólo en su trayecto, sino también desde su origen.

- ***Disponibilidad***

Este término hace referencia al método de precaución contra posibles daños tanto en la información como en el acceso a la misma: ataques, accidentes o simplemente descuidos, pueden ser factores que obliguen a diseñar métodos para posibles bloqueos.

- ***Autenticidad***

Algunos profesionales de la seguridad informática no incluyen este término, sino que nombran los tres anteriores. Particularmente, creo que no se puede soslayar este concepto, debido al hecho de la *integridad* se informa que el archivo, por ejemplo, no ha sido alterado, y *autenticidad* nos informa que el archivo en cuestión es real.

La autenticación de una computadora difiere con los términos de los humanos: para una computadora, autenticar no es lo mismo que identificar. Por ejemplo, en un sistema de seguridad donde se verifica la voz, el sistema se encarga de buscar un patrón en la voz para poder distinguir quién es. Este reconocimiento es de identificación, pero todavía falta la parte en que el usuario dice una frase o palabra clave, y es aquí donde la autenticación tiene efecto. Los métodos de autenticación para verificación de identidad pueden clasificarse en tres categorías:

Categoría 1: algo que el usuario sabe.

Un dato esencial, puede tratarse de algo de su persona o bien de un simple o complejo password (contraseña).

Categoría 2: algo que el usuario lleva consigo.
Puede ser un documento de identidad, una tarjeta o cualquier otro elemento que lleve consigo.

Categoría 3: propiedad física o acto involuntario.
La pupila, la voz y la huella dactilar son ejemplos de propiedades físicas de un individuo y firmar es un acto involuntario, ya que no está pensando en hacer cada trazo, sino que se realizan en conjunto.

7.1.1 Modelos de Seguridad

En gran medida, el éxito o fracaso de la seguridad depende del esfuerzo realizado en implementar los controles de seguridad que diseñamos para una organización. Antes de poder implementar dichos controles, se necesita comprender los requerimientos de seguridad específicos del sistema. Y este es el objetivo de un modelo de seguridad, especificar con precisión dichos requerimientos. El modelo que se escoja debe ser fácil de comprender, implementable, carente de ambigüedad y capaz de incorporar las políticas de la organización.

En organizaciones que requieran un alto nivel de seguridad, es esencial incorporar al modelo de seguridad una especificación formal de requerimientos y mecanismos de verificación. La especificación describirá el comportamiento que se propone para el sistema de forma precisa, sin ambigüedad, facilitando la verificación, que inicialmente permitirá probar que la especificación cumple el modelo y, posteriormente, que la implementación cumple la especificación.

El comportamiento que se propone para el sistema en cuanto a seguridad se ve fuertemente influenciado por las amenazas que el sistema debe afrontar y el tipo de sistema de que se trate. Es muy diferente plantear un modelo para un entorno de alta seguridad, más preocupado por la posible filtración de información confidencial, que para un entorno comercial, más preocupado por asegurar la integridad y actualidad de la información. Son estos tipos de detalles los que podrán guiar a la hora de elegir el modelo.

Una referencia importante en esta área es el denominado libro naranja1 (orange book), que durante años ha procurado un estándar para la evaluación de sistemas. Divide los sistemas en cuatro categorías (algunas se subdividen en clases) en función de los requerimientos que satisfacen. La categoría más baja es la D, que corresponde a sistemas que tras ser evaluados, no satisfacen los requerimientos de las categorías superiores. La siguiente categoría es la C, se divide en dos clases C1 y C2, e introduce controles de acceso discrecionales. La categoría B, con tres clases B1, B2 y B3, introduce controles de acceso obligatorios. La categoría A se reserva a sistemas que, aunque son funcionalmente equivalentes a los de la categoría B, disponen de completas especificaciones formales de diseño y en los que se han aplicado técnicas de verificación.

Aunque el orange book describe cuatro clases, realmente los divide en dos categorías: sistemas que incorporan controles de acceso discrecionales (categoría C), con seguridad limitada a un único nivel de seguridad, y sistemas que incorporan controles de acceso

obligatorios, que permiten procesar información de diferentes niveles de seguridad (categorías A y B).

Los controles de acceso discrecionales exigen un proceso de identificación y verificación del usuario a la entrada al sistema. Una vez verificado que se trata de un usuario autorizado, este podrá acceder a toda la información que no tenga definidas restricciones de acceso. Un ejemplo de este tipo de controles es el mecanismo de permisos de UNIX: cuando un usuario crea un objeto, puede o no definir un conjunto de permisos de manipulación para él, su grupo y los restantes usuarios.

Los controles de acceso obligatorios requieren múltiples niveles de seguridad. Los objetos y sujetos del sistema deben, obligatoriamente, tener un nivel de seguridad asociado. El sistema no debe permitir la creación de objetos o usuarios sin definir su nivel de seguridad, o que sean duplicados total o parcialmente con diferente nivel de seguridad. Cuando un usuario desea acceder a un objeto, el sistema comprueba los niveles de ambos y evalúa si se permite o no el acceso.

7.1.2 Tipos de Seguridad

Principalmente existen dos tipos de seguridad, física y lógica.

Seguridad Física

Es uno de los aspectos más olvidados a la hora del diseño de un sistema informático. Uno de los objetivos de la seguridad física es, por ejemplo, evitar que un usuario ajeno a la empresa u organización tenga acceso a una área restringida y/o que ese usuario pueda tomar una cinta de los respaldos del sistema.

Así, la seguridad física consiste en la "aplicación de barreras físicas y procedimientos de control, como medidas de prevención y contramedidas ante amenazas a los recursos e información confidencial". Se refiere a los controles y mecanismos de seguridad dentro y alrededor del Centro de Cómputo así como los medios de acceso remoto al y desde el mismo; implementados para proteger el hardware y medios de almacenamiento de datos.
Este tipo de seguridad está enfocado a cubrir las amenazas ocasionadas tanto por el hombre como por la naturaleza del medio físico en que se encuentra ubicada la organización o centro de cómputo.

Las principales amenazas que se prevén en la seguridad física son:

1. Desastres naturales, incendios accidentales, tormentas e inundaciones.
2. Amenazas ocasionadas por el hombre.
3. Disturbios, sabotajes internos y externos deliberados.

A veces basta recurrir al sentido común para darse cuenta que cerrar una puerta con llave o cortar la electricidad en ciertas áreas siguen siendo técnicas válidas en cualquier entorno.

Evaluar y controlar permanentemente la seguridad física del edificio es la base para comenzar a integrar la seguridad como una función primordial dentro de cualquier organismo. Tener controlado el ambiente y acceso físico permite:

- disminuir siniestros
- trabajar mejor manteniendo la sensación de seguridad
- descartar falsas hipótesis si se produjeran incidentes
- tener los medios para luchar contra accidentes

Seguridad Lógica

Luego de ver como el sistema puede verse afectado por la falta de seguridad física, es importante recalcar que la mayoría de los daños que puede sufrir un sistema no será sobre los medios físicos sino contra información almacenada y procesada.

Así, la seguridad física, sólo es una parte del amplio espectro que se debe cubrir para no vivir con una sensación ficticia de seguridad. Como bien se sabe el activo más importante de un sistema es la información, y por lo tanto deben existir técnicas, más allá de la seguridad física, que la aseguren. Estas técnicas las brinda la seguridad lógica.

Es decir que la *seguridad lógica* consiste en la "aplicación de barreras y procedimientos que resguarden el acceso a los datos y sólo se permita acceder a ellos a las personas autorizadas para hacerlo". Existe un viejo dicho en la seguridad informática que dicta que "todo lo que no está permitido debe estar prohibido" y esto es lo que debe asegurar la seguridad lógica.

Los objetivos que se plantean serán:

1. Restringir el acceso a los programas y archivos.
2. Asegurar que los usuarios puedan trabajar sin una supervisión minuciosa y no puedan modificar los programas ni los archivos que no correspondan.
3. Asegurar que se estén utilizando los datos, archivos y programas correctos en y por el procedimiento correcto.
4. Que la información transmitida sea recibida sólo por el destinatario al cual ha sido enviada y no a otro.
5. Que la información recibida sea la misma que ha sido transmitida.
6. Que existan sistemas alternativos secundarios de transmisión entre diferentes puntos.
7. Que se disponga de pasos alternativos de emergencia para la transmisión de información.

7.2 Criptoogía

La criptología es el estudio de los criptosistemas: sistemas que ofrecen medios seguros de comunicación en los que el emisor oculta o cifra el mensaje antes de transmitirlo para que sólo un receptor autorizado (o nadie) pueda descifrarlo.

En el año 500 a.C. los griegos utilizaron un cilindro llamado "scytale" alrededor del cual enrollaban una tira de cuero. Al escribir un mensaje sobre el cuero y desenrollarlo se veía una lista de letras sin sentido. El mensaje correcto sólo podía leerse al enrollar el cuero nuevamente en un cilindro de igual diámetro.

Durante el Imperio Romano Julio Cesar empleo un sistema de cifrado consistente en sustituir la letra a encriptar por otra letra distanciada a tres posiciones más adelante. Durante su reinado, los mensajes de Julio Cesar nunca fueron desencriptados.

Durante la segunda guerra mundial en un lugar llamado Bletchley Park (70 Km al norte de Londres) un grupo de científicos trabajaba en Enigma, la máquina encargada de cifrar los mensajes secretos alemanes.

En este grupo se encontraban tres matemáticos polacos llamados Marian Rejewski, Jerzy Rozycki, Henryk Zygalski y "un joven que se mordía siempre los dedos alrededor de las uñas, iba con ropa sin planchar y era más bien bajito". Este joven retraído se llamaba Alan Turing y había sido reclutado porque unos años antes había creado una computadora "binaria". Probablemente poca gente en los servicios secretos ingleses sabía lo que era una computadora (y mucho menos binaria)... pero no cabía duda que sólo alguien realmente inteligente podía inventar algo así, cualquier cosa que eso fuese.

Sería Turing el encargado de descifrar el primer mensaje de Enigma y, cambiar el curso de la guerra, la historia y de la Seguridad Informática actual.

7.2.1 Criptografía

La palabra Criptografía proviene etimológicamente del griego Kruiptoz (Kriptos-Oculto) y Grajein (Grafo-Escritura) y significa *"arte de escribir con clave secreta o de un modo enigmático"*.

Cabe aclarar que la criptografía hace años que dejó de ser un arte para convertirse en una técnica (o conjunto de ellas) que tratan sobre la protección (ocultamiento ante personas no autorizadas) de la información. Entre las disciplinas que engloba cabe destacar la Teoría de la Información, la Matemática Discreta, la Teoría de los Grandes Números y la Complejidad Algorítmica.

Es decir que la criptografía es la ciencia que consiste en transformar un mensaje inteligible en otro que no lo es (mediante claves que sólo el emisor y el destinatario conocen), para después devolverlo a su forma original, sin que nadie que vea el mensaje cifrado sea capaz de entenderlo (figura 4). El mensaje cifrado recibe el nombre Criptograma.

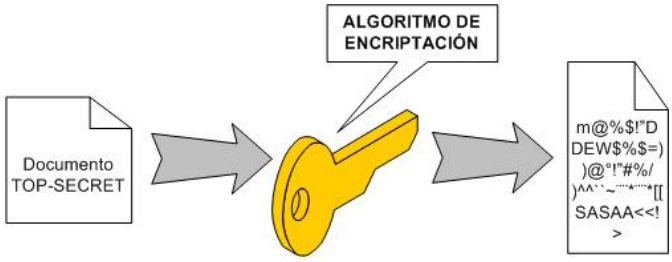


Figura 4. Esquema de cifrado

La importancia de la criptografía radica en que es el único método actual capaz de hacer cumplir el objetivo de la seguridad informática: *"mantener la privacidad, integridad,*

autenticidad..." y hacer cumplir con el No Rechazo (o no repudio), relacionado a no poder negar la autoría y recepción de un mensaje enviado.

7.2.2 Criptosistema

Un *criptosistema* se define como la quíntupla (m,C,K,E,D), donde:

m, representa el conjunto de todos los mensajes sin cifrar (texto plano) que pueden ser enviados.
C, representa el conjunto de todos los posibles mensajes cifrados, o criptogramas.
K, representa el conjunto de claves que se pueden emplear en el criptosistema.
E, es el conjunto de transformaciones de cifrado o familia de funciones que se aplica a cada elemento de *m* para obtener un elemento de *C*. Existe una transformación diferente E_k para cada valor posible de la clave *K*.
D, es el conjunto de transformaciones de descifrado, análogo a *E*.

Todo criptosistema cumple la condición $D_k(E_k\ (m))=m$ es decir, que si se tiene un mensaje *m*, se cifra empleando la clave *K* y luego se descifra empleando la misma clave, se obtiene el mensaje original *m*.

Existen dos tipos fundamentales de criptosistemas utilizados para cifrar datos e información digital y ser enviados posteriormente por medios de transmisión libre.

a) ***Simétricos o de clave privada***: se emplea la misma clave *K* para cifrar y descifrar, por lo tanto el emisor y el receptor deben poseer la clave. El mayor inconveniente que presentan es que se debe contar con un canal seguro para la transmisión de dicha clave.

b) ***Asimétricos o de llave pública***: se emplea una doble clave conocidas como K_P (clave privada) y *Kp* (clave pública). Una de ellas es utilizada para la transformación *E* de cifrado y la otra para el descifrado *D*. En muchos de los sistemas existentes estas clave son intercambiables, es decir que si emplea una para cifrar se utiliza la otra para descifrar y viceversa.

Los sistemas asimétricos deben cumplir con la condición que la clave pública (al ser conocida y sólo utilizada para cifrar) no debe permitir calcular la privada. Como puede observarse este sistema permite intercambiar claves en un canal inseguro de transmisión ya que lo único que se envía es la clave pública.

Los algoritmos asimétricos emplean claves de longitud mayor a los simétricos. Así, por ejemplo, suele considerarse segura una clave de 128 bits para estos últimos pero se recomienda claves de 1024 bits (como mínimo) para los algoritmos asimétricos. Esto permite que los algoritmos simétricos sean considerablemente más rápidos que los asimétricos.

En la práctica actualmente se emplea una combinación de ambos sistemas ya que los asimétricos son computacionalmente más costosos (mayor tiempo de cifrado). Para realizar dicha combinación se cifra el mensaje *m* con un sistema simétrico y luego se encripta la

clave *K* utilizada en el algoritmo simétrico (generalmente más corta que el mensaje) con un sistema asimétrico.

7.2.3 Esteganografía

Consiste en ocultar en el interior de información aparentemente inocua, otro tipo de información (cifrada o no). El texto se envía como texto plano, pero entremezclado con mucha cantidad de "basura" que sirve de camuflaje al mensaje enviado. El método de recuperación y lectura sólo es conocido por el destinatario del mensaje y se conoce como "separar el grano de la paja".

Los mensajes suelen ir ocultos entre archivos de sonido o imágenes y ser enormemente grandes por la cantidad extra de información enviada (a comparación del mensaje original).

7.3 Administración básica de la seguridad, políticas de cuentas y contraseñas

7.3.1 Políticas

Una política de seguridad es un plan que se basa en prioridades. Estas prioridades se apoyan en jerarquías de importancia, empezando desde lo menos importante a proteger y terminando, obviamente, en lo más sensible. Las prioridades no consisten únicamente en ataques, sino que abarcan todo tipo de rubros relacionados, por ejemplo, se habla de hardware, software, información, recursos y también de personas. Para realizar también este tipo de plan, no se puede dejar de recordar los pilares fundamentales de la seguridad. Una política de seguridad que no hable de proteger la autenticidad, disponibilidad, integridad y confidencialidad de los datos o cualquier elemento sensible no es una política correcta. Obviamente, no siempre se usan estos pilares, pero siempre debe existir la protección de por lo menos uno de ellos.

Además, se deben considerar varios factores. No olvidar que las reglas las pone el creador del método y es relativo a cada empresa ya que cada una tiene sus prioridades. Así mismo, hay métodos esquematizados que pueden seguirse.

Para empezar a armar una política de seguridad, primero hay que determinar ***"qué es lo que se desea proteger"*** y cuáles son las propiedades de los elementos. Por ejemplo, una comunicación contra un servidor puede estar autenticada. Es por eso que no se le da mucha importancia, por ende, la seguridad de ese mecanismo para llegar al destino va a ser baja, debido a que el túnel por donde viaja ***ya tiene seguridad***. Pero los elementos a los cuales acceden tal vez tienen otra vía para ser vistos, o por las propiedades de lectura/escritura/ejecución hace que el nivel de seguridad por el cual acceden se convierta en inconducente.

El segundo paso es saber ***"de quién hay que protegerse".*** Si bien el grado de paranoia dependerá de cuán importante sea la empresa y qué tan sensible sea la información, no olvidar que lo primordial es el *cómo*. Se puede poner una pared frente a la computadora para que no espíen, pero si no se sabe que hay gente que sabe trepar, jamás se entenderá cómo proteger la red. Al no saber con certeza quién será el atacante, se usa una técnica efectiva: *las restricciones*. Por ejemplo, a un ejecutivo de cuentas sólo se le da privilegios

para ver los archivos que necesita, ni más ni menos, y si precisa otra información, deberá seguir un procedimiento para obtenerla.

El tercer paso a considerar es ***"el riesgo"***. Aquí tiene mucha importancia el enfoque que se utiliza. Es en vano preocuparse por la seguridad de la base de datos de clientes, si a éstos no se les da la seguridad necesaria para que crean que la empresa es segura.

En el cuarto procedimiento, se debe implementar medidas de seguridad para ***"proteger los recursos de la empresa"***: firewalls, IDS, reglas de caducación de contraseñas, encriptación, etc., son ejemplos de métodos de protección.

El último paso es el de ***"mejorar".*** Cuando se habla de mejorar, se refiere a que un recurso de la empresa puede funcionar en forma más eficiente. Para saber en qué momento un recurso tiene fallas o si salió al mercado una versión más nueva (por consiguiente, pueden existir fallas conocidas), es necesario realizar un método para la verificación y enumeración de los recursos, con su versión y fecha, a fin de efectuar una revisión periódicamente (figura 5).

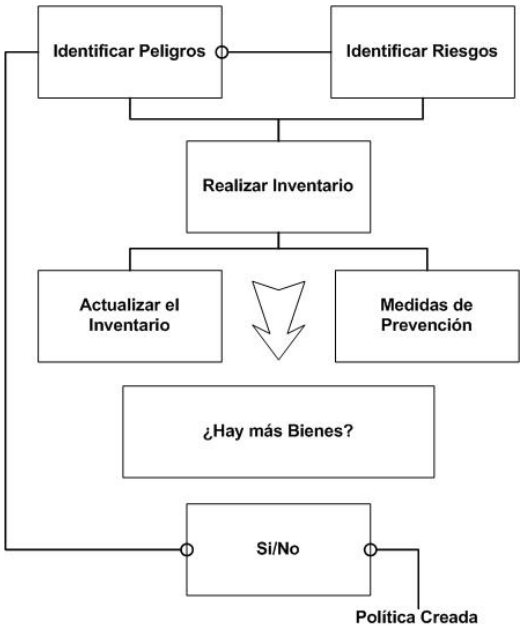


Figura 5. Proceso para la creación de una política.

A continuación se muestra un ejemplo de una política de contraseñas que puede ser usada por cualquier institución u organización.

Política de Contraseñas

1. *Objetivo y Ámbito de la Política*

Las contraseñas son un aspecto fundamental de la seguridad de los recursos informáticos, es la primera línea de protección para el usuario. Una contraseña mal elegida o protegida puede resultar en un agujero de seguridad para toda la organización. Por ello, todos los

usuarios son responsables de velar por la seguridad de sus contraseñas seleccionadas por ellos mismos para el uso de los distintos servicios ofrecidos.

La seguridad provista por una contraseña depende de que la misma se mantenga siempre en secreto, todas las directrices suministradas por esta política tienen por objetivo mantener esta característica fundamental en las contraseñas de los recursos de la organización. El objetivo fundamental de esta política es establecer un estándar para la creación de contraseñas fuertes, la protección de dichas contraseñas, y el cambio frecuente de las mismas.

El ámbito de esta política incluye a todos aquellos usuarios de los servicios y recursos informáticos de la organización que tienen o son responsables de una cuenta (o cualquier otro tipo de acceso que requiera una contraseña) en cualquiera de los sistemas.

2. *Política General*

Todas las contraseñas de cuentas que den acceso a recursos y servicios de la organización deberán seguir las siguientes directrices generales:

- Todas las contraseñas de sistema (root, administradores, cuentas de administración de aplicaciones, etc.) deben ser cambiadas al menos una vez cada seis meses.

- Todas las contraseñas de usuario (cuentas de bases de datos, cuentas de email, cuentas de servicios web, etc.) deben ser cambiadas al menos una vez cada doce meses. Sin embargo, se recomienda cambiarla con mayor frecuencia y también siempre que el usuario sospeche que la seguridad de su contraseña pueda haber sido comprometida.

- Las cuentas de usuario que tengan privilegios de sistema a través de su pertenencia a grupos o por cualquier otro medio, deben tener contraseñas distintas del resto de las demás cuentas mantenidas por dicho usuario.

- Las contraseñas no deben ser incluidas en mensajes de correo electrónico, ni ningún otro medio de comunicación electrónica. Tampoco deben ser comunicadas las contraseñas en conversaciones telefónicas.

- En la medida de lo posible, las contraseñas serán generadas automáticamente con las características recomendadas en esta política y se les comunicará a los usuarios su contraseña siempre en estado *expirado* para obligar al usuario a cambiarla en el primer uso que hagan de la cuenta o servicio.

- Las contraseñas por defecto asociadas a los sistemas o aplicaciones nuevas deberán ser cambiadas antes de poner estos sistemas en producción. También se desactivarán aquellas cuentas *por defecto* que no sean imprescindibles.

- Todas las contraseñas de sistema y de usuario de recursos y servicios deben respetar las recomendaciones descritas en la presente política. Algunos servicios en los que

sea crítico el mantener la seguridad de la contraseña podrán determinar medidas adicionales de protección de la misma.

3. *Selección y custodia de contraseñas*

Las contraseñas son usadas con múltiples propósitos en la organización, como pueden ser las contraseñas de sistema de los recursos de la organización, servicios web, cuentas de correo electrónico, protectores de pantalla en los recursos de los usuarios, administración de dispositivos remotos, etc. Se debe poner especial atención en la selección de contraseñas seguras para la autentificación en todos los recursos y servicios de la organización.

La seguridad de este tipo de autentificación se basa en dos premisas:

a. La contraseña personal sólo la conoce el usuario.
b. La contraseña es lo suficientemente ***fuerte*** para no ser descifrada.

Dentro de la asignación de contraseñas se pueden considerar dos niveles de seguridad:

a) *Seguridad completa*: Una contraseña de, al menos, 15 CARACTERES
b) *Seguridad dependiente*: Contraseña de 13 CARACTERES como máximo.

Las contraseñas ***débiles*** (no seguras) tienen alguna de las siguientes características:

- Contiene menos de 13 caracteres.
- Es una palabra que se encuentra en el diccionario (español o extranjero).
- Es una palabra de uso común, como por ejemplo:

    a. Nombres de la familia, animales, compañeros de trabajo, amigos, personajes, etc.
    b. Términos y marcas informáticos, comandos, compañías comerciales, hardware, software.
    c. Fechas de nacimiento y cualquier otra información personal.
    d. Patrones de letras o números, como 'aaabbb', 'qwerty', '123321', etc.
    e. Cualquiera de lo anterior escrito al revés.
    f. Cualquiera de lo anterior precedido o seguido por un dígito, por ejemplo *secreto1* o *1secreto*.

En cambio, las contraseñas ***fuertes*** (seguras) tienen las siguientes características:

- Más de 13 caracteres (quince para seguridad completa).
- Utiliza caracteres de tres de los cuatro grupos siguientes, SIEMPRE QUE UNO DE ELLOS SEA EL DE SÍMBOLOS:
  1. Letras minúsculas.
  2. Letras mayúsculas.
  3. Números (por ejemplo, 1, 2, 3).
  4. Símbolos (por ejemplo, ¡, @, Ñ, =, -, etc.).
- No ser ni derivarse de una palabra del diccionario o de un dialecto.
- No derivarse del nombre del usuario o de algún pariente cercano.

- No derivarse de información personal del usuario o de algún pariente cercano.

Finalmente, si prevé viajar al extranjero o utilizar teclados que no sean en español o latinoamericano, tenga en cuenta que los símbolos que utiliza en su contraseña pueden estar en sitios diferentes del teclado; por ejemplo no use letras ***ñ***.

Las contraseñas no deben ser almacenadas por escrito nunca. Intente crear contraseñas que pueda recordar fácilmente. Una forma de recordarlo con facilidad es crear una contraseña basada en una frase fácilmente recordable. Por ejemplo:

La frase: “hola pianola”
Sugiere la contraseña: 'H01@.piaN0L@'

Recomendaciones para la Protección de la Contraseña

No utilice la misma contraseña que utiliza para las cuentas de recursos y servicios en otras cuentas (acceso a su proveedor de servicios de internet, acceso a servicios de su banco, cuenta de correo, etc.).

No comparta las cuentas y contraseñas con nadie, incluyendo administrativos, secretarias, etc. Todas las contraseñas deben ser tratadas como información sensible y confidencial.

A continuación se presenta una lista de cosas que NO se deben hacer:

- No revele su contraseña por teléfono a NADIE, incluso aunque le hablen en nombre del servicio de informática o de un superior suyo en la organización.
- No revele la contraseña en mensajes de correo electrónico ni a través de cualquier otro medio de comunicación electrónica.
- Nunca escriba la contraseña en papel y lo guarde. Tampoco almacene contraseñas en archivos sin encriptar o proveerlo de algún mecanismo de seguridad.
- No revele su contraseña a sus superiores, ni a sus colaboradores.
- No hable sobre una contraseña delante de otras personas.
- No revele su contraseña en ningún cuestionario o formulario, independientemente de la confianza que le inspire el mismo.
- No comparta la contraseña con familiares.
- No revele la contraseña a sus compañeros cuando salga de vacaciones.
- No utilice la característica de "Recordar Contraseña" existente en algunas aplicaciones (Outlook, Netscape, Internet Explorer).

Si alguien le pide la contraseña, refiérale a este documento o pídale que se comunique con el Área de Sistemas de Información y Comunicaciones de la organización. Si sospecha que una cuenta o su contraseña pueden haber sido comprometidas, comuníquelo al ASIC y cambie las contraseñas de todas sus cuentas.

4. *Medidas a Aplicar*

El incumplimiento de la presente Política puede llegar a comprometer la seguridad de la totalidad de la red corporativa de la organización.

Será la Comisión de Informática de la organización la que decida las acciones a tomar en el caso de incumplimiento de la presente política una vez establecidas las repercusiones que sobre los recursos y servicios informáticos de la organización haya podido tener la violación de la misma. Todo ello sin perjuicio de las acciones disciplinarias, administrativas, civiles o penales que en su caso correspondan, a las personas presuntamente implicadas en dicho incumplimiento.

7.3.2 Monitoreo

- ***Registro***

Se deben mantener guardados los datos necesarios para poder efectuar análisis e investigaciones futuras. Algunos ejemplos son los ID's del usuario, la fecha y hora de inicio, la finalización de la conexión, la identidad o ubicación de la terminal, etc. También es importante para la estadística y la prevención guardar los intentos exitosos/fallidos de acceso al sistema, a los datos y a otros recursos.

- ***Monitoreo***

Se deben crear procedimientos para el monitoreo de las instalaciones de datos. Los mismos se utilizan para garantizar que los usuarios tengan intenciones legítimas y que no estén intentando acceder a información no correspondida. El nivel de monitoreo se determina a través de una evaluación de riesgos.

Entre las áreas que hay que tener en cuenta, se tienen las siguientes:

a) Acceso no autorizado, incluyendo detalles como:
- ID de usuario.
- Fecha y hora de eventos clave.
- Tipos de eventos.
- Archivos a los que se accede.
- Utilitarios y programas utilizados (procesos, memoria, servicios, etc.).

b) Todas las operaciones con privilegios como:
- Utilización de cuenta de superusuario.
- Inicio y cierre (startup y stop) del sistema.
- Conexión y desconexión de dispositivos i/o.

c) Intentos de acceso no autorizado:
- Intentos fallidos.
- Violaciones de la política de accesos y notificaciones para "gateways y firewalls".
- Alertas de sistemas patentados para detección de intrusos.

d) Alertas o fallas del sistema, como:
- Alertas y mensajes de consola.
- Excepciones del sistema de registro.
- Alarmas del sistema de administración de redes.

Se recomienda revisar periódicamente el resultado del monitoreo. La frecuencia es relativa a los riesgos involucrados. Los factores de riesgo más importantes son la criticidad de los

procesos de las aplicaciones y la valoración de la información involucrada, y en realidad, también los datos.

- ***Revisión***

Habitualmente, los registros (logs) del sistema contienen información (datos) innecesaria para la comprensión del monitoreo. Para saber cuáles eventos son significativos, es necesario utilizar herramientas que se encarguen de pasar la información cruda a algo entendible y significativo (información). También es factible redireccionar la salida de los registros para crear un segundo registro con la información entendible. Un "script" podría hacer fácilmente esta tarea.

Es importante cuidar la seguridad de los logs, debido a que un atacante podría vulnerar la confidencialidad y alterar la integridad.

7.4 Clasificación del los tipos de ataques o amenazas a los sistemas informáticos

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

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

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

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

Interrupción: un recurso del sistema es destruido o se vuelve no disponible. Este es un ataque contra la disponibilidad. Ejemplos de este ataque son la destrucción de un elemento hardware, como un disco duro, cortar una línea de comunicación o deshabilitar el sistema de gestión de archivos.

Intercepción: una entidad no autorizada consigue acceso a un recurso. Este es un ataque contra la confidencialidad. La entidad no autorizada podría ser una persona, un programa o una computadora. Ejemplos de este ataque son colgarse de una línea para hacerse de datos que circulen por la red y la copia ilícita de archivos o programas (intercepción de datos), o bien la lectura de las cabeceras de paquetes para descubrir la identidad de uno o más de los usuarios implicados en la comunicación observada ilegalmente (intercepción de identidad).

Modificación: una entidad no autorizada no sólo consigue acceder a un recurso, sino que es capaz de manipularlo. Este es un ataque contra la integridad, ejemplos de este ataque son el cambio de valores en un archivo de datos, alterar un programa para que funcione de forma diferente y modificar el contenido de mensajes que están siendo transferidos por la red.

Fabricación: una entidad no autorizada inserta objetos falsificados en el sistema. Este es un ataque contra la autenticidad. Ejemplos de este ataque son la inserción de mensajes espurios en una red o añadir registros a un archivo. Estos ataques se pueden asimismo clasificar de forma útil en términos de ataques pasivos y ataques activos.

7.4.1 Ataques Pasivos

En los ataques pasivos el atacante no altera la comunicación, sino que únicamente la escucha o monitoriza, para obtener información que está siendo transmitida.

Sus objetivos son la intercepción de datos y el análisis de tráfico, una técnica más sutil para obtener información de la comunicación, que puede consistir en:

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

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

7.4.2 Ataques Activos

Estos ataques implican algún tipo de modificación del flujo de datos transmitido o la creación de un falso flujo de datos, pudiendo subdividirse en cuatro categorías:

Suplantación de identidad: el intruso se hace pasar por una entidad diferente. Normalmente incluye alguna de las otras formas de ataque activo. Por ejemplo, secuencias de autenticación pueden ser capturadas y repetidas, permitiendo a una entidad no autorizada acceder a una serie de recursos privilegiados suplantando a la entidad que posee esos privilegios, como al robar la contraseña de acceso a una cuenta.

Reactuación: uno o varios mensajes legítimos son capturados y repetidos para producir un efecto no deseado, por ejemplo ingresar dinero repetidas veces en una cuenta dada.

Modificación de mensajes: una porción del mensaje legítimo es alterada, o los mensajes son retardados o reordenados, para producir un efecto no autorizado. Por ejemplo, el mensaje "Ingresa un millón de pesos en la cuenta A" podría ser modificado para decir "Ingresa un millón de pesos en la cuenta B".

Degradación fraudulenta del servicio: impide o inhibe el uso normal o la gestión de recursos informáticos y de comunicaciones. Por ejemplo, el intruso podría suprimir todos los mensajes dirigidos a una determinada entidad o se podría interrumpir el servicio de una red inundándola con mensajes espurios. Entre estos ataques se encuentran los de denegación de servicio, consistentes en paralizar temporalmente el servicio de un servidor de correo, Web, FTP, etc.

7.4.3 Análisis de Vulnerabilidades

Se dice que la seguridad informática es un rompecabezas compuesto por varias piezas pequeñas, que por sí solas no son capaces de ser eficaces. La política de seguridad, por ejemplo, sería la pieza que justifica y da coherencia a todas las demás, una solución de antivirus corporativo aporta su grano de arena en la protección de los equipos finales, un IDS trata de mantener la seguridad de la red etc.

No obstante, de todas las piezas que componen el rompecabezas de la seguridad, yo destacaría una como el pilar central de todo el conjunto. A saber, el análisis de vulnerabilidades. Es tan importante porque actúa en los dos niveles esenciales de toda política de seguridad. Por un lado, en el nivel operacional, ya que asegura que la configuración de los servidores, firewalls y equipos en general sea la adecuada; colaborando con técnicos y administradores para garantizar en su operativa diaria que los recursos de la organización están protegidos. Por otro lado, apoya al nivel de gestión de la seguridad al proporcionar información referente a uno de los pilares básicos de toda gestión de riesgos, esto es, conocer los puntos débiles o vulnerables de la infraestructura.

Y no sólo eso, cada vez más empresas se ven en el compromiso de demostrar su grado de seguridad frente a terceros. Ya no sólo basta con ser seguros, ahora los clientes exigen demostrar que la infraestructura posee un nivel aceptable de seguridad.

Todo administrador de sistemas debe hacerse las siguientes preguntas, para ver hasta que punto necesita un análisis de vulnerabilidades.

- ¿Conozco los puntos débiles de mi infraestructura?
- Puede un atacante externo entrar en mis servidores? ¿A cuáles?
- ¿Puede bloquearlos?
- ¿Cómo podría hacerlo?
- ¿Y un empleado malintencionado?
- ¿Existen en alguno de mis sistemas (SO, BBDD, protocolos, etc.) cuentas por defecto o mal configuradas, sin password o con uno demasiado débil?
- ¿Cuánto le costaría a un atacante llevar a cabo las anteriores acciones?
- ¿Qué puedo hacer para remediar los problemas detectados?
- ¿Cómo evoluciona la seguridad de mi infraestructura con el tiempo?
- ¿Cómo afectan a la seguridad los últimos cambios que he realizado?

La realización de análisis de vulnerabilidades periódicos, internos y externos, con herramientas automáticas, puede proporcionar la respuesta a las anteriores preguntas en muy poco tiempo y con muy poco esfuerzo. Lo que es aún más importante, puede proporcionar la información de base para actuaciones más ambiciosas en el ámbito de la seguridad corporativa.

NOTA. Es importante conocer lo que puede llegar a suceder antes de que suceda porque entonces, probablemente, ya sea demasiado tarde.

7.5 Seguridad perimetral

Cuando se habla de seguridad perimetral, más que nada se hace referencia a asegurar la frontera entre la red interna y el resto de Internet. El objetivo es restringir o controlar que datos entran a la organización o salen de ella. La principal ventaja de este tipo de seguridad es que permite al administrador concentrarse en los puntos de entrada. No es necesario, por tanto, que se preocupe de todos y cada uno de los sistemas de la red interna.

Su principal inconveniente es que da una falsa sensación de seguridad total y conduce, a menudo, a descuidar otros aspectos que también son importantes.

El ejemplo más destacable de seguridad perimetral lo constituyen los firewalls. Aunque los hay de muchos tipos, básicamente lo que hacen es un filtrado de paquetes y/o contenidos (figura 6).

7.5.1 Firewall

Un sistema básico de seguridad, que se debe utilizar para la conexión a Internet, es la instalación de un *Firewall*. Un firewall es un sistema de defensa que se basa en la instalación de una "barrera" entre la computadora y la red, por la que circulan todos los datos. Este tráfico entre la red y la computadora es autorizado o denegado por el firewall, siguiendo las reglas que se le hayan configurado.

El funcionamiento de un firewall se basa principalmente en el "filtrado de paquetes". Todo dato o información que circule entre la computadora y la red es analizado por el firewall con la misión de permitir o denegar su paso en ambas direcciones (Internet -->Computadora o Computadora--->Internet).

El comprender esto último es muy importante, ya que si se autoriza un determinado servicio o programa, el firewall no va a avisar que es correcto o incorrecto, o incluso, que siendo correcto los paquetes que están entrando o saliendo, éstos contienen datos perniciosos para el sistema o la red, por lo que hay que tener buen cuidado en las autorizaciones que se otorgue.
Como ejemplo de esto último se puede poner el *correo electrónico*. Si se autoriza en el firewall a que determinado programa de correo acceda a Internet, y al recibir el correo, en un mensaje recibido viene un adjunto con un virus, por ejemplo tipo gusano, el firewall no va a defender de ello, ya que se le ha autorizado a que ese programa acceda a la red. Lo que si va a hacer es que si al ejecutar el adjunto, el gusano intenta acceder a la red por algún puerto que no esté previamente aceptado en las reglas, no lo va a dejar propagarse. Ahora bien, si hace uso por ejemplo del mismo cliente de correo, si va a propagarse. La misión del firewall es la de aceptar o denegar el tráfico, pero no el contenido del mismo, aunque si hay firewalls que hacen un análisis de contenidos.

Una buena política debería ser, ante la duda, no aceptar nunca cualquier acceso hasta comprobar que es necesario para un correcto funcionamiento del servicio que se pretenda usar y no es potencialmente peligroso para el sistema. Si se deniega el acceso y el sistema sigue funcionando bien, no es necesario, por lo que se debe denegar.

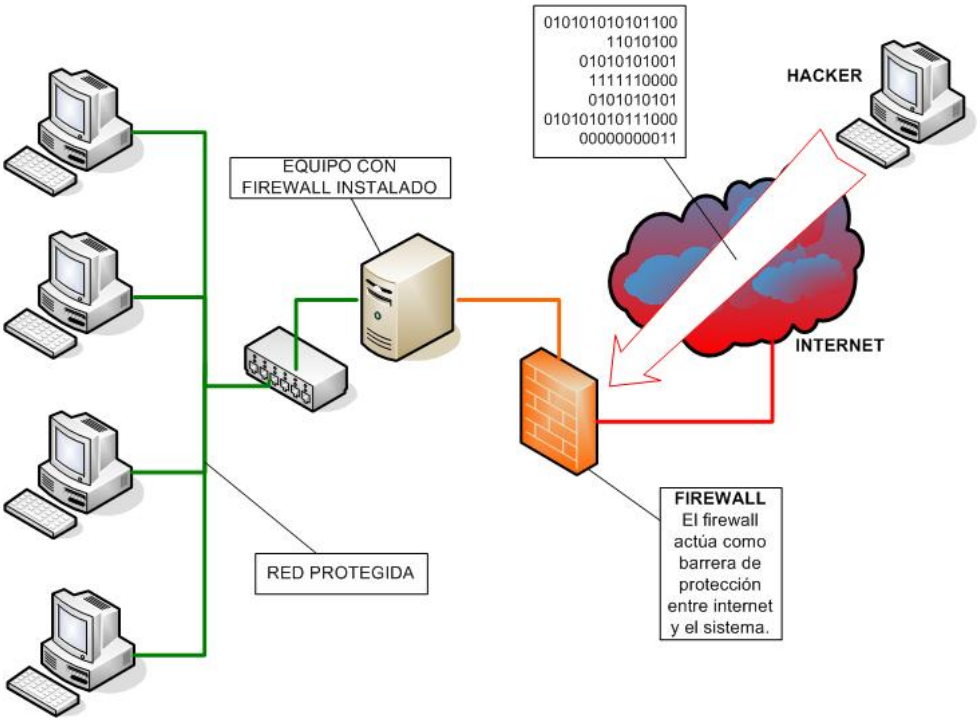


Figura 6.Esquema básico de seguridad de una red.

7.5.1.1 Firewalls en Linux

Desde la series 1.1, el kernel de Linux posee en mayor o menor medida capacidad para filtrar tramas. Originalmente (1994), ipfwadm era la herramienta proporcionada con Linux para la implementación de políticas de filtrado de paquetes en este clon de Unix; derivaba del código de filtrado en BSD (ipfw), y debido a sus limitaciones (por ejemplo, sólo puede manejar los protocolos TCP, UDP o ICMP) ipfwadm fue reescrito para convertirse en ipchains a partir del núcleo 2.1.102 (en 1998). Esta nueva herramienta (realmente, todo el subsistema de filtrado de los núcleos 2.2) introdujo bastantes mejoras con respecto a la anterior, pero seguía careciendo de algo fundamental: el stateful, era difícil ver a un sistema tan potente como Linux sin una herramienta de firewalling decente, libre, y con el sistema, mientras otros clones de Unix, también gratuitos hacía tiempo que la incorporaban, como es el caso de FreeBSD e IPFilter.

De esta forma, no es de extrañar que a partir del núcleo 2.3.15 (por tanto, en todos los kernels estables, de la serie 2.4, desde mediados de 1999) ipchains fuera sustituido por iptables, que de nuevo introducía importantes mejoras con respecto a su predecesor. Sin duda la más importante era que ya incorporaba el stateful no presente en ipchains, pero no era la única; además, iptables ofrece, un sistema de NAT (Network Address Translation) mucho más avanzado, incorpora mejoras en el filtrado (llegando incluso a filtrar en base a la dirección física de las tramas) e inspección de paquetes, y presenta un subsistema de log mucho más depurado que ipchains. Por tanto, iptables es en la actualidad el software de firewalling en Linux IPv4; aunque todas las versiones de Linux lo incorporan por defecto, se puede descargar una versión actualizada desde http://netfilter.samba.org/.

Históricamente, todos los sistemas de firewalling nativos de Linux han sido orientados a comando. Esto significa, muy por encima, que no leen su configuración de un determinado

archivo, por ejemplo durante el arranque del sistema, sino que ese archivo de arranque ha de ser un script donde, línea a línea, se definan los comandos a ejecutar para implantar la política de seguridad deseada, esta es una importante diferencia con respecto a otros firewalls, como IPFilter, orientados a archivo: en estos la política se define en un simple archivo ASCII con una cierta sintaxis, que el software interpreta y carga en el sistema.

7.5.2 Sistemas Detectores de Intrusos (IDS)

Un IDS o Sistema de Detección de Intrusiones o Intrusos es una herramienta de seguridad que intenta detectar o monitorizar los eventos ocurridos en un determinado sistema informático o red en busca de intentos de comprometer la seguridad del sistema.

- Los IDS buscan patrones previamente definidos que impliquen cualquier tipo de actividad sospechosa o maliciosa sobre la red o host.
- Los IDS aportan a la seguridad una capacidad de prevención y de alerta anticipada ante cualquier actividad sospechosa. No están diseñados para detener un ataque, aunque sí pueden generar ciertos tipos de respuesta ante éstos.
- Los IDS aumentan la seguridad del sistema, vigilan el tráfico de la red, examinan los paquetes analizándolos en busca de datos sospechosos y detectan las primeras fases de cualquier ataque como pueden ser el análisis de la red, barrido de puertos, etc.

7.5.2.1 Tipos de IDS

- *HIDS (Host IDS)*

Protege contra a un único Servidor, computadora o host. Monitorizan gran cantidad de eventos, analizando actividades con una gran precisión, determinando de esta manera qué procesos y usuarios se involucran en una determinada acción. Recaban información del sistema como archivos, logs, recursos, etc, para su posterior análisis en busca de posibles incidencias. Todo ello en modo local, dentro del propio sistema. Fueron los primeros IDS en desarrollar por la industria de la seguridad informática.

- *NIDS (Net IDS)*

Protege un sistema basado en red. Actúan sobre una red capturando y analizando paquetes de red, es decir, son *sniffers* del tráfico de red. Luego analizan los paquetes capturados, buscando patrones que supongan algún tipo de ataque.
Bien ubicados, pueden analizar grandes redes y su impacto en el tráfico suele ser pequeño. Actúan mediante la utilización de un dispositivo de red configurado en modo promiscuo (analizan, "ven" todos los paquetes que circulan por un segmento de red aunque estos nos vayan dirigidos a un determinado equipo). Analizan el tráfico de red, normalmente, en tiempo real. No sólo trabajan a nivel TCP/IP, también lo pueden hacer a nivel de aplicación.

7.5.3 IDS en Linux "SNORT"

Snort es un IDS o Sistema de detección de intrusiones basado en red (NIDS). Implementa un motor de detección de ataques y barrido de puertos que permite registrar, alertar y responder ante cualquier anomalía previamente definida como patrones que corresponden a

ataques, barridos, intentos aprovechar alguna vulnerabilidad, análisis de protocolos conocidos, etc. Todo esto en tiempo real.

Puede funcionar como *sniffer* (podemos ver en consola y en tiempo real que ocurre en la red, todo el tráfico), registro de paquetes (permite guardar en un archivo los logs para su posterior análisis, un análisis offline) o como un IDS normal (en este caso NIDS).

8. Desarrollo de Aplicaciones con PostgreSQL y PHP.

8.1 PostgreSQL

PostgreSQL es un manejador de base de datos relacional libre, orientado a objetos, incorpora casi todas las funcionalidades de SQL, incluyendo subconsultas, transacciones, tipos y funciones definidas por el usuario. Para el desarrollo de bases de datos se tiene la herramienta Pgaccess, que brinda una interfaz gráfica que acorta los tiempos de desarrollo y facilita la generación de reportes, tarea fundamental cuando se administran bases de datos.

Los sistemas de mantenimiento de Bases de Datos relacionales tradicionales (DBMS,s) soportan un modelo de datos que consisten en una colección de relaciones con nombre, que contienen atributos de un tipo específico. Postgres ofrece una potencia adicional sustancial al incorporar los siguientes cuatro conceptos adicionales básicos en una vía en la que los usuarios pueden extender fácilmente el sistema: clases, herencia, tipos y funciones.

Otras características que aportan potencia y flexibilidad adicional son: *restricciones* (constraints), *disparadores* (triggers), *reglas* (rules), *integridad transaccional*. Estas características colocan a Postgres en la categoría de las Bases de Datos identificadas como *objeto-relacionales*. Postgres tiene algunas características que son propias del mundo de las bases de datos orientadas a objetos. De hecho, algunas bases de datos comerciales han incorporado recientemente características en las que Postgres fue pionera.

PostgreSQL es un manejador de base de datos objecto-relacional, altamente escalable, compatible con SQL y con más de 15 años de desarrollo. Surge en la Universidad de Berkeley, basado en Ingres y toma el nombre de Postgres bajo el liderazgo de Michel Stonebraker. En 1996 cambia de nombre de postgres95 a PostgreSQL.

8.1.1 Instalación de PostgreSQL

Esta se puede realizar de dos formas:

Por paquetes de su distribución favorita.

Compilación

```
$ tar zxvf postgresql-8.0.3.tar.gz
$ cd postgresql-8.0.3
$ ./configure --prefix=/usr/local/pgsql
$ make
# make install
#adduser postgres
#passwd postgres
#mkdir /usr/local/pgsql/data
# chown postgres /usr/local/pgsql/data
# su - postgres
$ /usr/local/pgsql/bin/initdb -D /usr/local/pgsql/data
$ /usr/local/pgsql/bin/postmaster -D /usr/local/pgsql/data >logfile 2>&1 &
$ /usr/local/pgsql/bin/createdb test
$ /usr/local/pgsql/bin/psql test
```

8.2 Programación Orientada a Objetos

La Programación Orientada a Objetos (POO), es una metodología de programación que se fundamenta en el concepto de objeto. Es una filosofía de programación donde la realidad se

modela mediante objetos y la interacción entre ellos.

Ventajas de la POO
• La forma de programar pasa a un segundo plano, permite agregar funcionalidad sin incrementar en la misma proporción la complejidad.
• Los programas no son sólo líneas que se ejecutan unas tras otras.
• Permite identificar entidades que conviven en un sistema dado.
• La herencia permite la reutilización de código.
• Favorece la generación de bancos de componentes.

Características de la POO
• *Abstracción*: Surge de reconocer las similitudes entre ciertos objetos, situaciones o procesos del mundo real, es decir, en un sistema el objeto sirve de modelo abstracto el cual realiza trabajo, informa, cambia de estado y se comunica con otros objetos pero sin revelar la implementación de estas características.

• *Encapsulamiento*: Almacena en un mismo lugar los elementos de una abstracción que constituyen su estructura y funcionamiento, para proteger las propiedades de un objeto contra su modificación por quien no tenga derecho a acceder a ellas, solamente los propios métodos internos del objeto pueden acceder a su estado. Esto asegura que otros objetos no pueden cambiar el estado interno de un objeto de manera inesperada.

• *Modularidad*: Se refiere a fragmentar un programa en componentes para reducir su complejidad. Repercute en el acoplamiento o integración de los componentes.

• *Jerarquía*: Define una estructura mediante la cual se clasifican los objetos. Permite acceder más fácilmente a la información.

8.2.1 Objetos y Clases

Una *clase* es una entidad que declara conjuntos de objetos similares; es una plantilla (template) para un tipo particular de objetos donde cada uno de estos objetos tendrá un estado propio, aunque pueden hacer operaciones comunes. La definición de la clase debe realizarse dentro del mismo bloque de código de PHP. Para crear una clase se utiliza la sentencia ***class.*** Estructura mínima de una clase:

```
class NombreClase {
}
```

Un *objeto* es una entidad que contiene las características que describen el estado de un objeto del mundo real y las acciones que se asocian con el objeto. Se designa por un nombre o identificador del objeto y se describe mediante una notación gráfica que varia de acuerdo a la metodología que se este aplicando.

```
objeto =new class();
```

Atributos de un Objeto: contenedor de un tipo de datos asociados a un objeto (o a una clase de objetos), que hace los datos visibles desde fuera del objeto, y cuyo valor puede ser

alterado por la ejecución de algún método.

Método de un objeto: Es una rutina que define el comportamiento del objeto o una clase de objetos, cuya ejecución sólo puede desencadenarse a través de un mensaje recibido por éste o por sus descendientes. Los métodos pueden heredarse de unos objetos a otros. El principal de todos los métodos de un objeto se denomina constructor.

nombre_objeto->nombre_metodo (parametros);

8.2.2 Constructores e Instancias

Los *constructores* son funciones o métodos, que se encargan de realizar las tareas de inicialización de los objetos al ser instanciados. Es decir, cuando se crean los objetos a partir de las clases, se llama a un constructor que se encarga de inicializar los atributos del objeto y realizar cualquier otra tarea de inicialización que sea necesaria.

El constructor se define dentro de la propia clase, de la misma forma que un método y requiere tener el mismo nombre de la clase. La sintaxis para la creación de un constructor es*: __construct()* (dos guiones bajos antes de la palabra *construct*).

Los destructores son funciones que se encargan de realizar las tareas que se necesita ejecutar cuando un objeto deja de existir. Cuando un objeto ya no está referenciado por ninguna variable, el objeto se debe destruir para liberar espacio. Para ello se llama a la función *__destruct(),* se define de la misma forma que un método.

8.2.3 Implementación de Herencia

La herencia se define a partir de una clase ya existente, para tomar sus características en clases más especializadas o derivadas de ésta, al crear una herencia se reutiliza el código. Las clases que heredan incluyen tanto los métodos como las propiedades de la clase a partir de la que están definidos.

Tipos de herencia: *Múltiple* (más de un padre) y *Simple* (sólo se hereda de un padre). La herencia en PHP es simple, no se soporta la herencia múltiple. Se utiliza la palabra reservada ***extends*** en el encabezado de la clase. Se heredan métodos, variables y constructores.

class nombre extends cart

8.2.4 Sobrescritura de Métodos y Atributos

La sobrescritura o sustitución de métodos, es un mecanismo por el cual una clase que hereda puede modificar la funcionalidad de los métodos que está heredando. La sobrescritura de métodos es común en mecanismos de herencia, puesto que los métodos que fueron creados para una clase *padre* no tienen porque ser los mismos que los definidos en las clases que heredan.

Referencias a métodos de la clase base, al sobreponer un método, se pierde el acceso directo a los métodos sobrepuestos de la clase base, pero es posible llamar directamente a un método de la super clase, ejemplo:

- Cuando se trata de un constructor
  parent::__construct();

- Cuando se trata de un método
  parent::metodo();

Atributos y métodos *static*, el declarar un atributo de forma estática (*static)*, permite que el atributo pueda ser invocado desde fuera de la clase sin la necesidad de instanciar un objeto. La variable *$this* no es posible utilizarla para acceder a atributos *static*, ni tampoco es posible incluirla en un método definido como *static*, a esté tipo de métodos y atributos se les conoce como "De Clase"

Visibilidad es el nivel de protección para acceder a un atributo o método de una clase.
- *Public*: Pueden ser accesados desde cualquier parte.
- *Private*: El acceso está limitado sólo a la clase que define el objeto.
- *Protected*: Limitan el acceso a las clases heredadas (y a la clase que define el elemento)

8.3 Uso de templates en PHP

Los *templates* (plantillas) son archivos HTML, que facilitan el proceso de edición, eliminando el código PHP incrustado, sin eliminar la funcionalidad. Permite separar totalmente la programación y el diseño de una aplicación Web.

Ventajas
- El programador no se preocupa por la presentación, está en un archivo distinto.
- El diseñador minimiza el riesgo de que al modificar alguna imagen se pierda la funcionalidad por una modificación involuntaria.
- Se puede rediseñar el sitio, de forma más sencilla sin modificar la funcionalidad.

Desventajas
- La tarea de desarrollo es ligeramente más complicada para el programador.
- Se requiere de un preproceso para generar el html final, lo cual consume tiempo de procesamiento.
- El diseñador debe considerar aun algunas etiquetas y trabajar con frecuencia con archivos fraccionados.

Existen una gran variedad de motores de templates *Engines*, por ejemplo: Smarty, NokTemplate, FastTemplate, PHP Savant, entre otros. Pero todos permiten separar la capa lógica de la presentación, con sus diferentes ventajas e inconvenientes.

La idea fundamental de un *PHP Template Engine* es muy sencilla:
1. Se diseña sobre un html convencional.
2. Se insertan *tags* particulares que representan las secciones dinámicas de un sitio.
3. El *Engine* busca por los tags y los reemplaza por código dinámico, definido previamente.
4. Se genera el código HTML que se mostrará al visitante.

Algunos *Engines* incorporan cache o buffers con el fin de evitar regenerar una página cada vez que se solicita, la intención es optimizar el desempeño.

8.3.1 La clase Class.NokTemplate

NokTemplate es una clase escrita en PHP, para manejar templates. Básicamente lo que hace es interpolación de variables, es decir, dada una variable definida en el template, el script lo que hace es intercambiar esa variable por su correspondiente valor (asignado en tiempo de ejecución). NokTemplate se puede descargar de:

http://www.jpw.com.ar/descargas.php

Características de NokTemplate

- Modo Debug: Posibilita la optimización del script.
- Sistema de cache temporal en archivo externo: Lo que incrementa considerablemente la velocidad de procesamiento.
- Velocidad de interpolación. Utiliza expresiones regulares compatibles con Perl.
- Soporta definición de bloques anidados dentro de las plantillas.
- Fue desarrollada para PHP 4.x

Lógica que utiliza NokTemplate

- Carga templates.
- Asigna valores a las variables.
- Expande templates (analizar el contenido).
- Muestra el resultado.

Las variables dentro de las plantillas se definen entre llaves ('{' y '}') y puede contener letras mayúsculas, minúsculas, números y guión bajo (_), las variables son sensibles a las mayúsculas y minúsculas, por lo que no es lo mismo {MiVariable} que {MIVARIABLE}.

Para crear un ejemplo se utilizaran 2 templates: *cuerpo.html y contenido.html*

cuerpo.html

```
<html>
<head>
<title>{TITULO} - Soportado por NokTemplate</title>
</head>
<body>
Hola {NOMBRE}!!<br>
</body>
</html>
```

contenido.html

```
Hola, mi nombre es {NOMBRE}.
```

Index.php

```
<?php
```

```
include ('Class.NokTemplate.php');
// Incluir la clase
$html = new NokTemplate('./templates');
// Crear el objeto html, instancia de NokTemplate y se define el lugar donde se //encuentran los templates
$html->cargar('tCuerpo','cuerpo.html');
// Cargar el contenido del archivo cuerpo.html en la variable tCuerpo
$html->asignar('TITULO','Ejemplo numero 1');
// Asignar a la variable TITULO el valor "Ejemplo numero 1"
$html->asignar('NOMBRE','Juan');
// Asignar a la variable NOMBRE el valor "Juan"
$html->expandir('FINAL', 'tCuerpo');
// Se procesa el contenido de tCuerpo, reemplazando las variables NOMBRE Y
TITULO, el resultado se almacena en FINAL
$html->imprimir('FINAL');
// Imprimir el contenido de la variable FINAL
?>
```

8.3.2 Template Smarty

Smarty, uno de los Engines de templates para PHP ampliamente utilizado. Se puede descargar de: *http://smarty.php.net/download.php*
Smarty, no solamente separa el código PHP del código HTML, también sirve para mantener un orden en la aplicación ya que de esta manera es más fácil rastrear un error y/o revisar un módulo de la aplicación;

Características de Smarty

- Es rápido, eficiente y dispone de cache.
- Arquitectura Plugin.
- Compila solo una vez, ya que no analiza gramaticalmente desde arriba el template. Además solo recompila los archivos del template que han sido cambiados.
- Se pueden crear funciones habituales y modificadores de variables customizados, de modo que el lenguaje del template es altamente extensible.
- Sintaxis de etiquetas delimitadoras para configuración del template, se puede usar como: {}, {{}}, <!--{}-->, etc.
- Los construtoress if/elseif/else/endif son pasados por el interpretador de PHP, así la sintaxis de la expresión {if ...} puede ser compleja o simple de la forma que se quiera.

8.4 Patrones de diseño en PHP

Los Patrones de Diseño son soluciones simples basadas en *mejores prácticas* para problemas recurrentes de la programación orientada a objetos. Los patrones de diseño son también conocidos como *phpPatterns*, se pueden clasificar en las siguientes categorías:

- *Patrones de Creación*: Muestran la guía de cómo crear objetos cuando sus creaciones requieren tomar decisiones. Estas decisiones generalmente serán resueltas dinámicamente decidiendo que clases instanciar o sobre que objetos un objeto

delegará responsabilidades.
- *Patrones Estructurales*: Describen la forma en que diferentes tipos de objetos pueden ser organizados para trabajar unos con otros.
- *Patrones de Comportamiento*: Se utilizan para organizar, manejar y combinar comportamientos.

Dentro de las tres categorías anteriores se encuentran una serie de patrones de diseño, aunque los patrones de diseño más ampliamente usados son:

- *Factory*: Permite cargar nuevas clases e instanciar nuevos objetos en tiempo de ejecución.
- *Singleton*: Obliga a que una clase sólo pueda tener una instancia de ella y en consecuencia un único punto de interacción.
- *Estrategia*: Permite que sea posible utilizar un algoritmo independientemente de los clientes que lo utilicen.
- *Facade*: Permite la integración de interfaces, con el fin de facilitar el uso del sistema.
- *Cadena de responsabilidad*: Permite a un objeto enviar un comando sin necesidad de conocer que objeto lo realizará, dado que el comando se transmite mediante la cadena de objetos que interactúan

8.5 Trabajando bases de datos de PostgreSQL en PHP

Acceder a Postgres desde PHP, para ello se debe compilar php habilitando el soporte para postgres, mediante la opción: --with-pgsql. Se utilizan funciones del tipo "pg_<action>" donde cada una de ellas define una acción en específico.

Para facilitar el proceso de conexión a la base de datos se suelen utilizar las siguientes variables:

```
PGHOST=pgsql.example.com
PGPORT=7890
PGDATABASE=web-system
PGUSER=web-user
PGPASSWORD=secret
PGDATESTYLE=ISO
PGTZ=JST
PGCLIENTENCODING=EUC-JP

export PGHOST PGPORT PGDATABASE PGUSER PGPASSWORD
PGDATESTYLE PGTZ PGCLIENTENCODING
```

String de conexión, algunas de las variables definidas conforman el string de conexión necesario para conectarse a Postgres.

```
$conn_string = "host=localhost  port=4523  dbname=midb user=pedro  password=gato";
```

Con el string de conexión definido se puede proceder a establecer una conexión a la base de datos.

$con = pg_connect($conn_string);

Funciones de uso común.

- Conectar a la Base de datos
  $con = pg_connect($con);
- Realizar un Select sobre la base de datos
  *$result = pg_query ($con, "SELECT * FROM prueba order by id desc");*
- Realizar el insert del contenido del arreglo *$arreglo* en la tabla *tabla*
  $res = pg_insert($con, 'tabla', $arreglo);
- Recuperar el contenido del arreglo *$result*
  $arr = pg_fetch_array($result);
- Cerrar la conexión a la base de datos
  pg_close($con);

8.5.1 Creación de bases de datos en PostgreSQL

Iniciando Postgres

$ su - postgres

Las bases de datos se crean dentro de Postgres con el comando *createdb*. El usuario que teclea el comando debe ser el *super-usuario* de Postgres, o haber obtenido privilegio por parte del super-usuario para crear bases de datos. Para crear una base de datos a partir de la línea de comandos, se utiliza la instrucción:

$ CREATEDB uno

Para obtener el mismo resultado dentro de *psql* se utiliza la instrucción:

$CREATE DATABASE uno;

Para acceder a la base de datos que se ha creado, mediante la línea de comandos de PostgreSQL *psql*, se utiliza la instrucción:

$ psql uno

El comando CREATE USER requiere sólo un parámetro: el nombre del nuevo usuario. Aunque contiene una variedad de opciones que pueden ser establecidas, incluyendo una contraseña, un ID de sistema explícito, grupo, y un juego de permisos que pueden ser específicamente definidos.

Para realizar una conexión a PostgreSQL a través del cliente *psql* se utiliza la instrucción:

user@local$ psql -h host –p 4523 database

Cuando el usuario registrado ya cuenta con privilegios sobre algún objeto de base de datos en la máquina local se utiliza la instrucción:

postgres@local$ psql prueba

Para terminar con una sesión de psql se utiliza la instrucción:

prueba=# \q

Para obtener ayuda acerca de los comandos de sesión del entorno utilizar la instrucción:

prueba=# \?

Para la ayuda con respecto a los comandos de sql se utiliza la instrucción:

prueba=# \h

Para crear una tabla, se utiliza la siguiente sintaxis:

CREATE TABLE nombre (columna tipo [DEFAULT valor] [NOT NULL], ... [INHERITS (hereda, ...)] [CONSTRAINT nom_cons CHECK (prueba), CHECK (prueba)]);

Donde:

- Nombre: Es el nombre que se le da a la tabla.
- Columna: Es el nombre de la columna de la tabla.
- Tipo: Es el tipo de dato (varchar, char, int, date, time, etc),
- Valor: El valor que tendrá por defecto.
- Hereda: Define una herencia de otra tabla, es decir, creará una entidad que contiene las columnas de la tabla que se crean y las heredadas.
- Nom_cons: Define una regla de integridad a respetar cada vez que se modifica una tupla.
- Prueba: Condición a comprobar

CREATE TABLE uno (product_no integer PRIMARY KEY, name text, price numeric);

Algunos tipos de datos

Numericos

- *Smallint*: entero de dos bytes con signo
- *int, integer:* entero de cuatro bytes con signo
- *real*: número de punto flotante de doble precisión
- *numeric*: número de precisión múltiple
- *decimal (p[,q]):* número decimal con signo de *p* dígitos de precisión, asumiendo *q* a la derecha para el punto decimal. ($15 \geq p \geq q$q ≥ 0). Si *q* se omite, se asume que vale 0.
- *float:* número de punto flotante con precisión

Caracteres

- *char(n)*: cadena de caracteres de longitud fija, de longitud *n.*
- *varchar(n)*: cadena de caracteres de longitud variable, de longitud máxima *n*.
- *text*: cadena de caracteres, de longitud variable ilimitada.

Fechas

- *date:* dato de fecha que contendrá año, mes, día, AAAA/MM/DD
- *time:* dato de tiempo que contendrá horas, minutos, segundos, centésimas**,** HH:MM:SS:CCC
- *timestamp:* dato fecha y hora, AAAA/MM/DD:HH:MM:SS:CCC

Para ver las tablas de una base de datos utilizar la instrucción:

prueba=# \d

Para revisar la estructura de una tabla se utiliza la instrucción:

prueba=# \d tabla

Para procesar un script de instrucciones sql se utiliza la instrucción:

prueba=# \i script.sql

Para la eliminación de una base de datos utilizar la instrucción:

$ DROPDB prueba

Algunos comandos para trabajar con las bases de datos son:

- *INSERT INTO*: inserta datos en una tabla, la ausencia de algún campo implica que éste recibirá el valor *NULO*.
- *UPDATE*: actualiza datos en una tabla.
- *DELETE*: borra datos de una tabla.
- *ALTER*: modifica una tabla de la base de datos.
- *SELECT*: selecciona información de la base de datos; este comando utiliza operadores como and, or, not, etc.

8.5.2 Respaldar una base de datos

Los comandos *pg_dump* y *pg_dumpall* se utilizan para respaldar las tablas, en archivos de texto simple que contienen las ordenes de SQL requeridas para reconstruir la base de datos tal y como se encontraba antes de hacer el respaldo, este respaldo se puede reconstruir en el mismo sistema, cualquier otro sistema, en otra arquitectura, etc. Cuando se actualiza la versión de PostgreSQL es necesario primero respaldar, para posteriormente volverlas a cargar.

- ***pg_dump*** se utiliza para respaldar una base de datos o una tabla en particular
  Para respaldar una base de datos se utiliza la sintaxis:
  pg_dump option dbname > respaldodb.sql
  Para cargar nuevamente la base de datos:
  psql –d database -f respaldo.sql
  Para respaldar una base de datos en un archivo *.tar*.
  pg_dump - Ft midb > db.tar

- ***pg_dumpall*** respalda todas las bases de datos en el sistema.

- ***pg_restore*** [*option...*] [*filename*]
  Para cargar el respaldo en una base de datos existente llamada *newdb*
  pg_restore -d newdb db.tar

CAPITULO II
Proyecto de un Sistema de Facturación en Línea

1. Proyecto de un sistema de facturación en línea

En este capítulo describiré tanto especificaciones como el funcionamiento del Sistema de Facturación en línea versión 0.1 describiendo cada uno de sus módulos que lo componen.

Todo sistema surge de la o las necesidades que un determinado usuario puede tener, la cual en este caso, fue satisfecha con software libre.

2. Justificación

Para el desarrollo del sistema sólo se hizo uso de software libre, por las siguientes causas:

- Los costos de implementación de software libre son completamente nulos.
- Cada aplicación que fue seleccionada para el desarrollo del sistema, ofrece una amplia portabilidad y robustez.
- Demasiadas empresas a nivel mundial están optando por el uso de software libre por la confiabilidad que éste ofrece.

Las aplicaciones que fueron seleccionadas para el desarrollo del sistema son:

a. *Sistema Operativo*: GNU Linux, las causas de haber seleccionado este sistema operativo, fueron:
   - es seguro
   - es libre
   - es portable
   - es multiusuario
   - aunque no ofrece soporte por alguna empresa, existe una amplia documentación en línea, entre otras.

b. *Servidor de Páginas WEB*: Apache, este servidor de páginas Web es hasta la fecha el más utilizado a nivel mundial.

c. *Manejador de Base de Datos*: MySQL, este DBMS ha ido creciendo demasiado rápido. En su página principal ofrece una amplia documentación. Ofrece un rendimiento óptimo y una amplia conectividad con el lenguaje de programación PHP.

d. *Lenguaje de Programación*: PHP, las causas de haber seleccionado este lenguaje son:
   - Ofrece un amplio conjunto de funciones para la conectividad con DBMS, en este caso, para MySQL.
   - Es fácil de programarlo.
   - Se interpreta a nivel de Servidor.

3. Objetivo

El principal objetivo del Sistema de Facturación en Línea, es el de generar las facturas en un formato en el cual pueda almacenarse una copia y a su vez pueda imprimirse una para el cliente, y este formato es: '***PDF***'.

4. Requisitos para la Implementación del Sistema de Facturación en Línea

a) Sistema Operativo GNU Linux

- Versión de kernel: 2.4 o superior

b) Sistema Manejado de Bases de Datos MySQL

- Versión: 4.1.11 o superior.
- Configuración:
  Puerto de Conexión: 3306
  Usuario: root.

c) Servidor de Páginas WEB Apache

- Versión: 2.0.53 o superior
- Configuración:
  - En la parte de la compilación de apache se debe incluir las directivas para que tenga soporte a base de datos MySQL y a PHP como módulo.

d) Lenguaje de Programación PHP

- Versión: 4.3.11 o superior.

5. Descripción y Definición del Sistema

En este apartado se describirá el diseño de la base de datos que fue utilizada para este sistema, que es el de una tienda de ventas de Computadoras portátiles y el Manejo del Sistema.

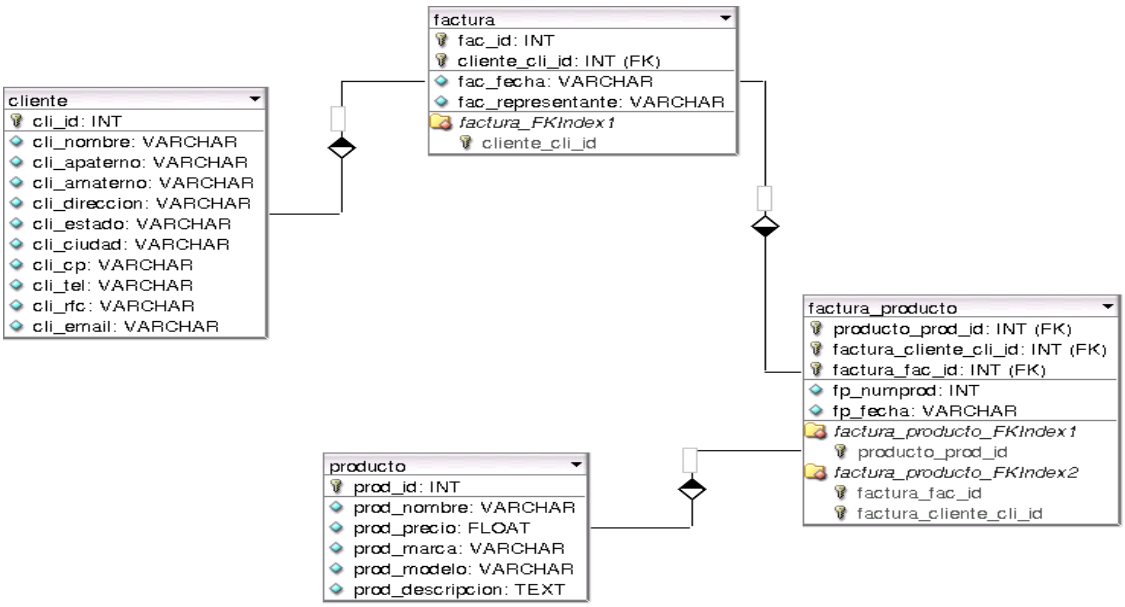


Figura 7. Diagrama de la base de datos.

El sistema está desarrollado por módulos, los cuales son descritos a continuación.

- ***Módulo de Autenticación***.

Este módulo consiste en la autenticación de un usuario (Administrador) ante el sistema para poder hacer uso de él.

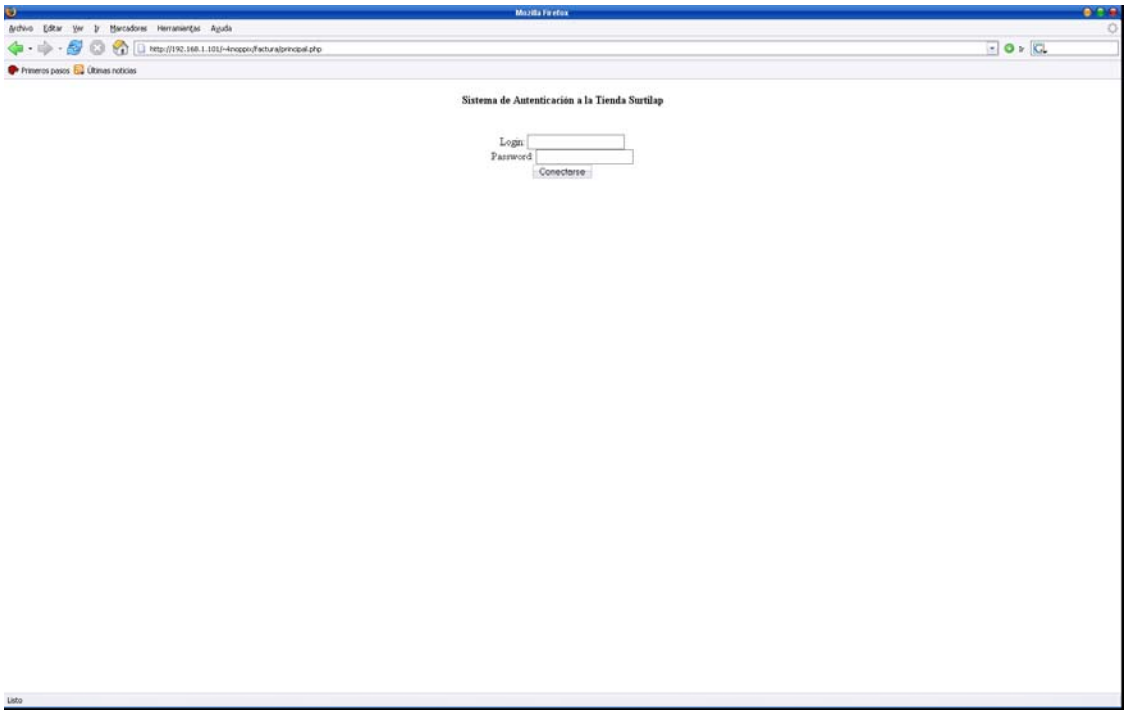


Figura 8. Autenticación al sistema

Al momento de ingresar el usuario y password o contraseña correctos, direcciona al módulo que muestra el menú principal del sistema.

- ***Módulo de Menú Principal***

Este módulo sólo despliega tres frames los cuales son:

- En la parte superior, el título de la empresa o logo.
- En la parte izquierda, se muestra el menú de las opciones que se pueden realizar en el sistema:
  - Alta de Clientes.
  - Borrado de Clientes.
  - Actualización de Clientes.
  - Compras.
  - Alta de Productos.
  - Borrado de Productos.
  - Reportes Por Día.
  - Salir
- Y el que sobra es el que despliega cada uno de los módulos antes nombrados.

Figura 9. Menú principal

- ***Módulo de Altas de Clientes***

El objetivo de este módulo es el de dar de alta nuevos clientes para poder tenerlos en la base de datos y con ello llevar un mejor control de los clientes y poder así identificarlos por claves.

Los datos solicitados para un nuevo cliente son:

- Nombre
- Apellido Paterno
- Apellido Materno
- Dirección
- Estado
- Ciudad
- Código Postal
- Teléfono
- RFC
- email

Y sólo queda dar click en el botón [continuar] para poder registrar al usuario.

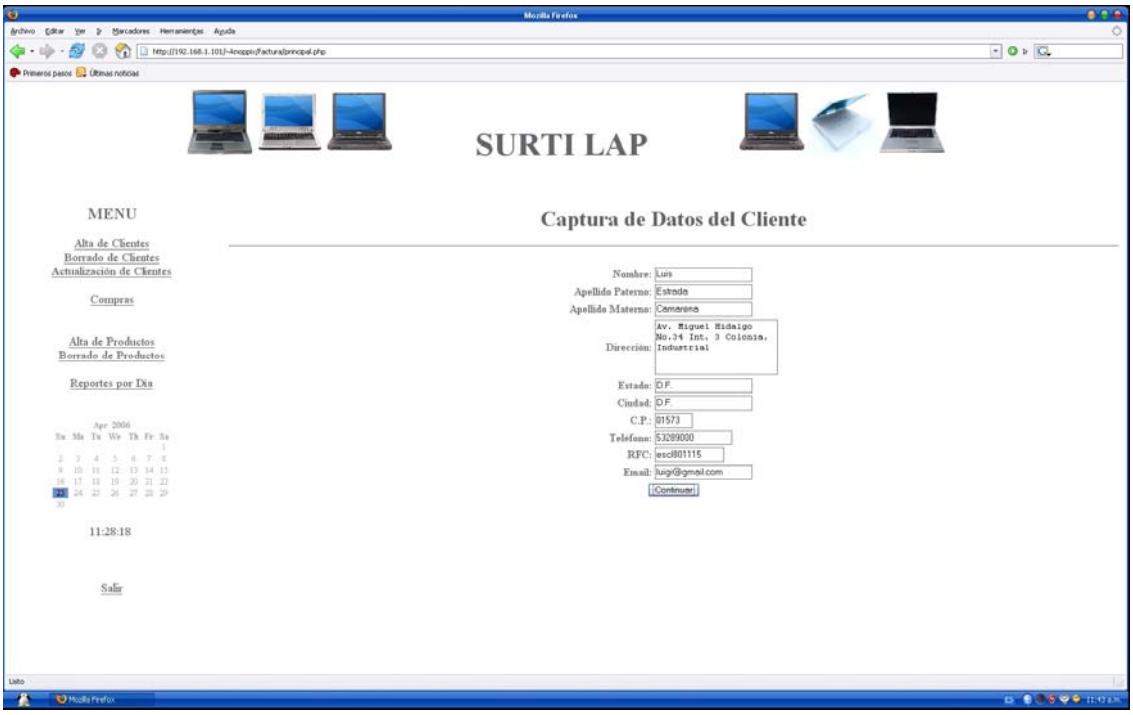


Figura 10. Captura de datos del cliente

- ***Módulo de Borrado de Clientes***

El objetivo de este módulo es el de borrar clientes de la base de datos.

Los pasos para poder borrar un cliente son:

Del Menú principal seleccionar la liga [Borra Cliente] la cual despliega la lista de clientes que se tienen almacenados, al lado de cada cliente se encuentra un radio botón, el cual es necesario activar y dar click en el botón [Borrar Cliente] para poder borrar al Cliente, saliendo una ventana de confirmación para borrar el cliente y si se realizó correctamente el borrado del cliente despliega una segunda pantalla la cual muestra un mensaje informativo que muestra que se ha realizado con éxito el borrado del cliente.

Figura 11. Selección del cliente a borrar

Figura 12. Mensaje de confirmación

Figura 13. Mensaje cuando el cliente se ha borrado

- ***Módulo Actualización de Clientes***

El objetivo de este módulo es el de actualizar los datos de un cliente (en caso de que se hayan introducido erróneamente).

El modo de funcionamiento de este módulo es:

Seleccionar la liga del menú principal que dice Actualización de Clientes la cual despliega la lista de clientes con un radio botón del lado derecho, el cual para poder actualizar a un cliente debe ser seleccionado y posteriormente dar click en el botón [Actualizar Cliente], el cual despliega un formulario con los datos del cliente seleccionado de forma que estos puedan ser modificados, al ser actualizados los datos ya sólo queda dar un click en el botón [Actualizar] y si no ocurrió algún error se despliega un mensaje informando de que los datos del cliente fueron actualizados de forma correcta.

Figura 14. Selección del cliente para actualizar datos

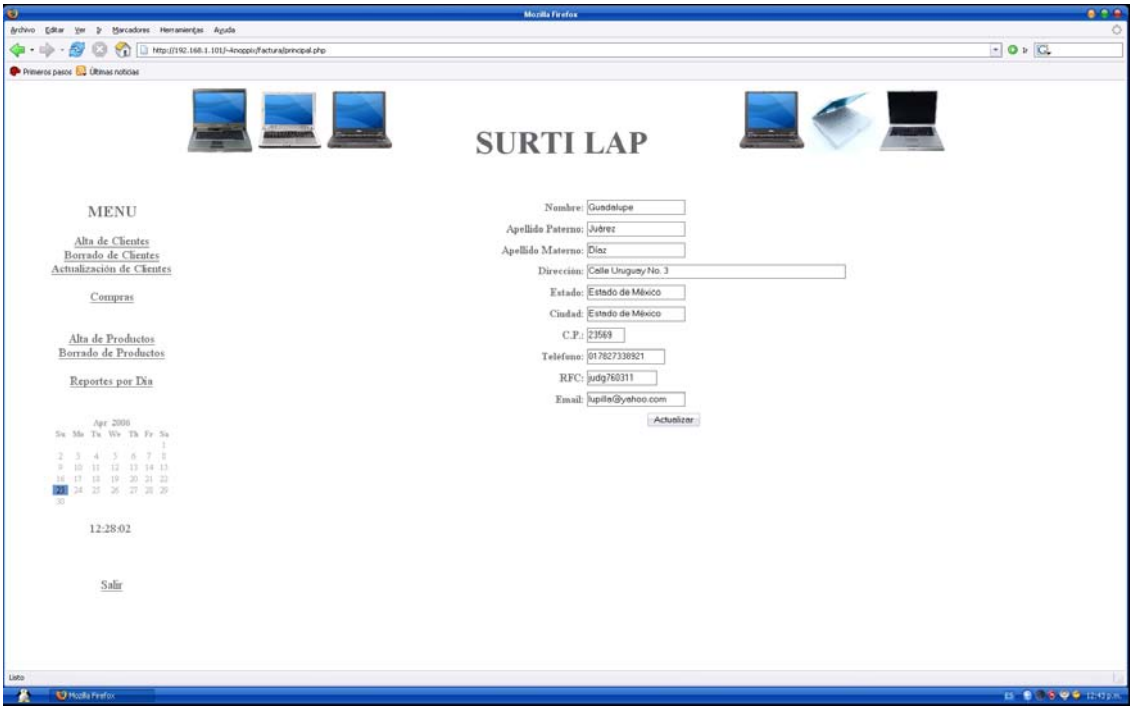


Figura 15. Actualizando datos

Figura 16. Mensaje de la actualización del cliente

Figura 17. Resultado de la actualización de un cliente

- ***Módulo de Compras***

El objetivo de este sistema es el de realizar una compra por parte de un cliente y entregarle su factura.

Este módulo está formado por dos módulos, que es el de la compra y el de la generación de la factura. Y su funcionamiento es:

Del módulo principal se selecciona la liga Compras, la cual despliega un listado de clientes, de la cual se selecciona al cliente que va a realizar la compra. Al ya haber seleccionado a un cliente y dar click en el botón [Continuar con la Compra], el sistema redirecciona al listado de productos con los que cuenta la empresa de los cuales solicita un número y activación de compra del o de los productos y finalmente se da click en el botón [Comprar], en este momento se ha realizado la compra y se crea la factura en formato PDF, ahora sólo queda guardarla, imprimirla y entregársela al cliente.

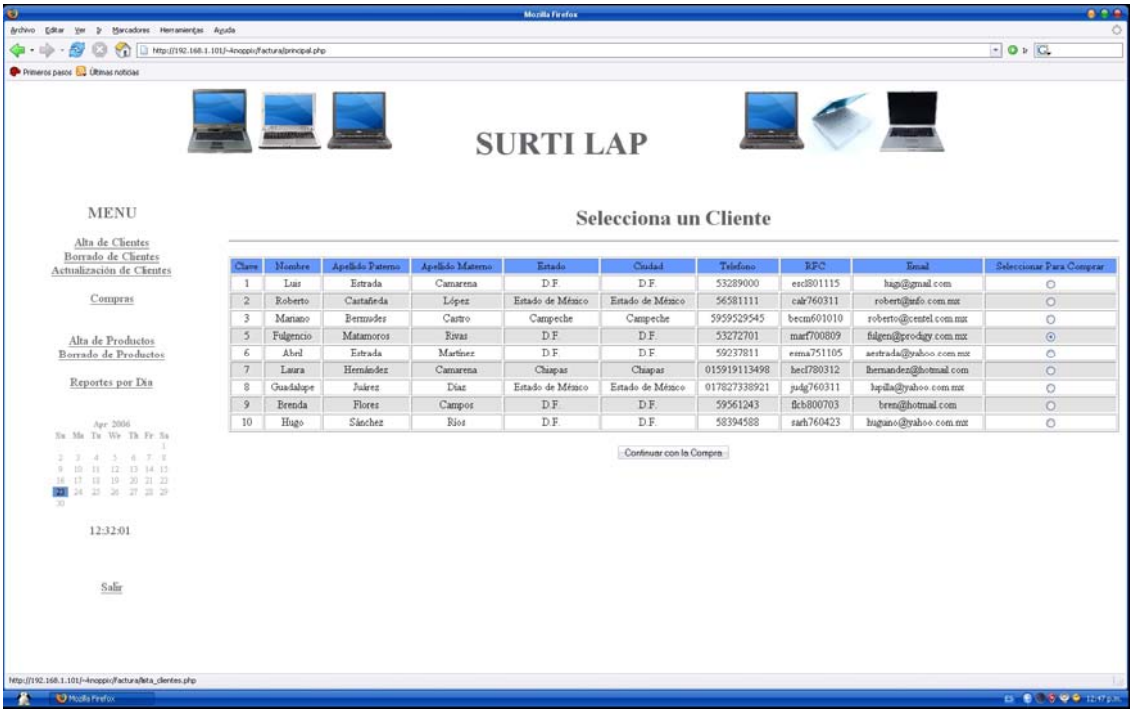


Figura 18. Selección del cliente

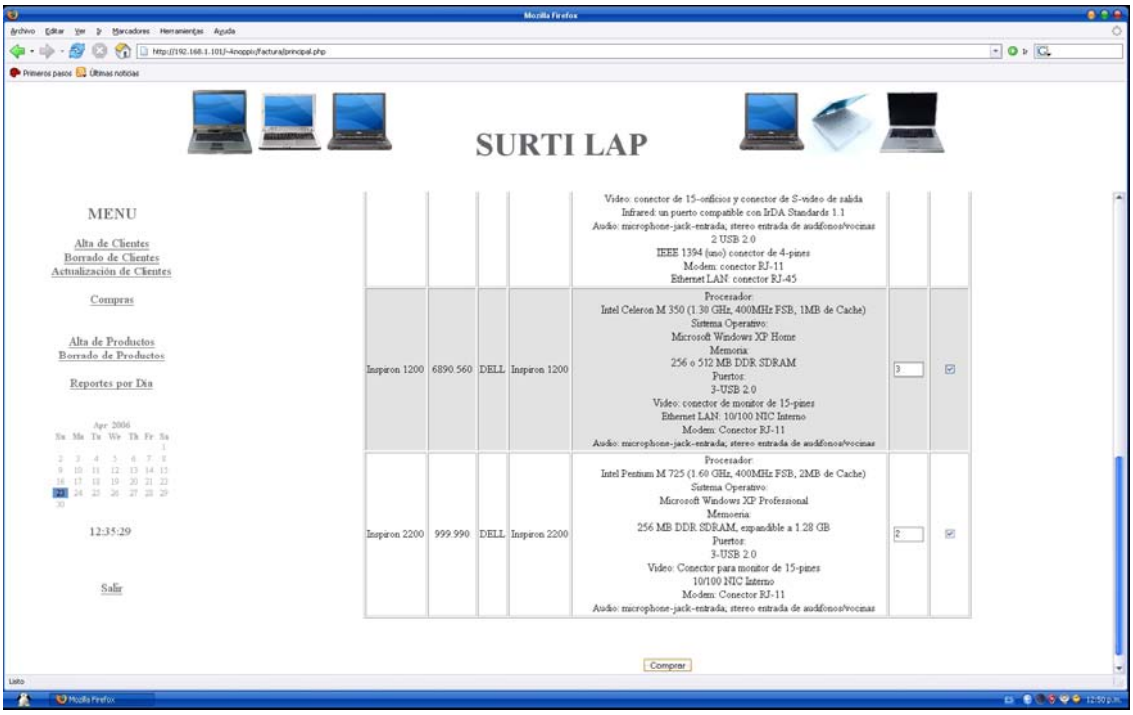


Figura 19. Selección de los productos ha comprar

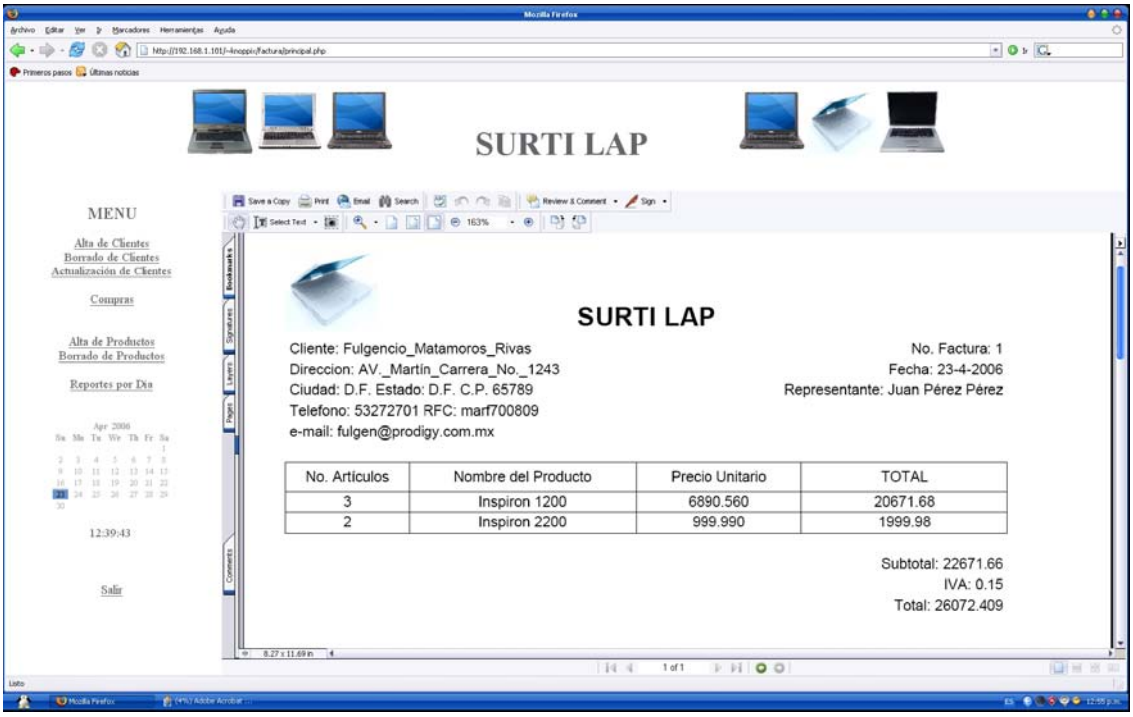


Figura 20. Factura de la compra

- ***Módulo de Alta de Productos***

El objetivo de este módulo es el de dar de alta nuevos productos.

Los datos solicitados para un nuevo producto son:

- Nombre del Producto.
- Precio.
- Marca.
- Modelo.
- Descripción.

Figura 21. Captura de datos de un producto nuevo

El funcionamiento de este módulo es muy similar al del Módulo de "Alta de Clientes".

- ***Módulo de Borrado de Productos***

El objetivo de este módulo es el de borrar o dar de baja alguno de los productos existentes de la lista.

El modo de funcionamiento de este módulo es similar al modo de funcionamiento del módulo del "Borrado de Clientes"

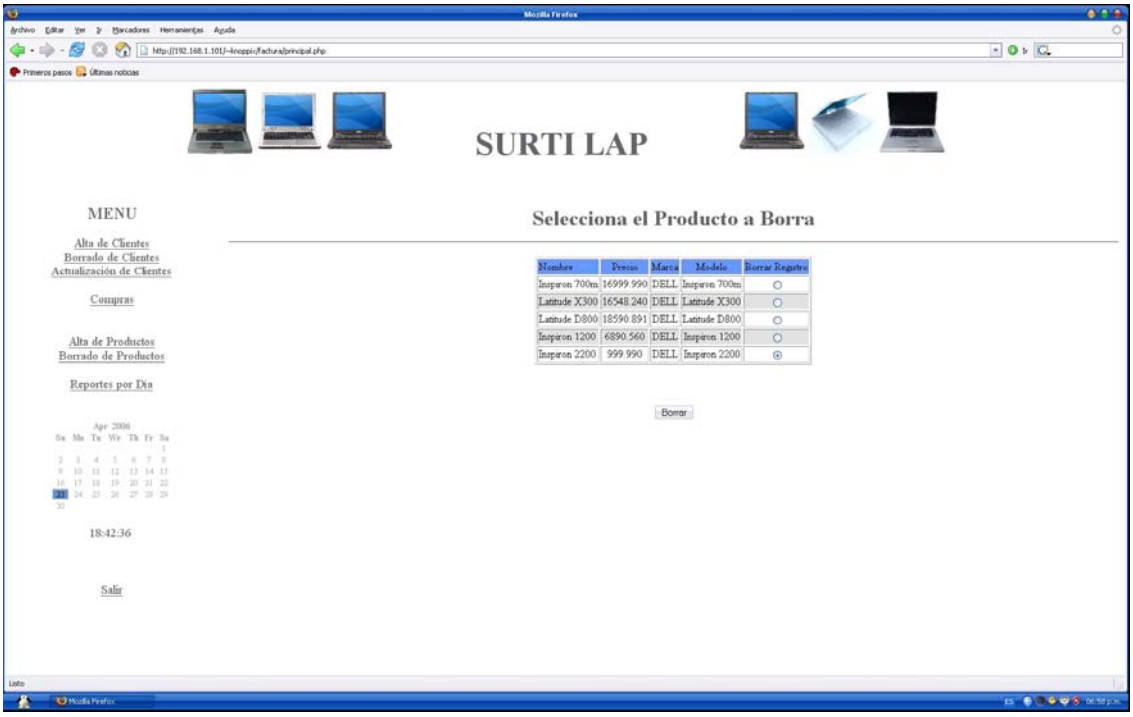


Figura 22. Selección del producto ha borrar

- ***Módulo de Reportes***

El objetivo de este módulo es el de generar los reportes de las ventas de un determinado día.

El funcionamiento de este módulo es:

Dar un click en la liga [Reportes por Día] del menú principal la cual desplegará tres listas de selección de modo que se pueda escoger una fecha, ya seleccionada la fecha y dar click en el botón [Generar Reporte], genera un reporte de todas las ventas que fueron realizadas esa fecha en formato PDF.

Figura 23. Selección de la fecha a consultar

Figura 24. Reporte de la fecha selecciona para consultar

- ***Módulo de Salir***

El objetivo de este módulo es el de salir del sistema.

El funcionamiento es simple, sólo se da click en la liga del menú principal que dice "Salir", la cual redirecciona al sistema de autenticación, para que si se requiere entrar al sistema ingrese nuevamente el nombre de usuario y contraseña.

CONCLUSIÓN

Hablando de software informático, existía una sola tendencia, software propietario, la cual consistía en el uso de una licencia que extendía la empresa dueña de ese software, pero surgió el software libre. Lo cual revolucionó el desarrollo de software a nivel mundial.

Uno de los mitos que se llegan a escuchar por parte de la empresas que desarrollan software propietario, es que, “el software libre, por ser libre es más propenso a errores e inseguro”, pero esto simplemente es un mito, puesto que como bien sabemos el sistema operativo más conocido y usado a nivel mundial es el más vulnerable y con más fallos en su funcionamiento.

Como nota; una empresa dedicada al desarrollo de software (Borland), desarrollo un software dedicado al manejo de bases de datos llamado *interbase*, el cual, por más de seis años fue vendido a las empresas. Años más tarde se descubrió que este software contenía una puerta trasera, la cual fue incrustada desde el momento que fue desarrollada la aplicación, esta puerta trasera permitía a la empresa desarrolladora de esta aplicación conectarse remotamente a todas las máquinas con este software instalado. Como se puede ver no todas las empresas laboran de la forma más correcta.

En la actualidad varias empresas están optando por el uso de software libre, por una simple razón, "ahorro de costos en licencias". Y es que en el software libre estas simplemente no existen.

El software libre ha demostrado que puede competir en rendimiento, portabilidad y robustez contra cualquier software propietario. A parte de que este software ofrece los fuentes, lo que conlleva a una implementación de este software completamente personalizada.

En lo personal los conocimiento que tenía de linux antes de tomar el diplomado eran mínimos, y con el diplomado de desarrollo de aplicaciones con software libre, estos conocimientos se ampliaron y con ello el panorama de posibilidades de brindar un ambiente más seguro y portable.

Puedo decir que con el módulo de introducción a la seguridad pude ver que tan vulnerable puede ser un sistema y qué tan comprometido puede estar éste en caso que no se lleve a cabo un plan o un régimen de políticas.

Con el desarrollo de un sistema, tan simple como es el del “sistema de facturación en línea” pude constatar todas las consideraciones que se deben tomar en cuenta para el desarrollo de éste, por ejemplo una buena administración del servidor de páginas Web, determinar cuántos usuarios pueden acceder al sistema a la vez, así mismo del DBMS MySQL, administrar los usuarios que pueden tener acceso a la base de datos del sistema y del lenguaje de programación la portabilidad que este puede tener.

Cada uno de los módulos me proporcionó los conocimientos básicos como para poder desempeñar mis labores con software libre sin necesidad de recurrir al uso de software propietario y poder con ello satisfacer las necesidades de clientes que requieren de algún sistema.

BIBLIOGRAFÍA Y FUENTES

LIBROS

- MORITSUGO, Steve y DTR Business System, Inc. Unix. Prentice Hall. 2000.
- Ben Laurie & Peter Laurie. Apache The Definitive Guide. O'Reilly & Associates, Inc.
- GIL RUBIO, Fco. Javier, TEJEDOR CERBEL, Jorge A. YAGÜE PANADERO, Agustín. VILLAVERDE, Santiago Alonso. GUTIÉRREZ RODRÍGUEZ, Abraham. Creación de sitios web con PHP4. Osborne MacGraw-Hill. 2001
- WELLING, Luke. THOMSON, Laura. PHP and MySQL Web Development. Sams Publishing. 2001.
- Tim Converse and Joyce Park with Clark Morgan PHP5 and MySQL® Bible. Wiley Publishing, Inc. 2004
- DuBois, Paul. Edición Especial MySQL. Prentice Hall. 2001.
- Randy Jay Yarger, George Reese, and Tim King. MySQL & mSQL. O'Reilly. 1999.
- Firtman, Sebastián J. Seguridad Informática (Manuales users). MP ediciones. 2005.
- Korry Douglas, Susan Douglas. PostgreSQL. Sams Publishing. 2003.
- Bruce Momjian. PostgreSQL Introduction and Concepts. Addison-Wesley. 2001.

FUENTES

- The Slackware Linux Project
  http://www.slackware.org/
- Linux Hardware Compatibility HOWTO
  http://www.tldp.org/HOWTO/Hardware-HOWTO/
- Curso de Html
  http://atena.mascarones.unam.mx/~alf/html/
- The Apache Software Foundation.
  http://www.apache.org
- PHP
  http://www.php.net/
- PHP: Hackers Paradise
  http://www.e-gineer.com/v1/articles/php-hackers-paradise.htm
- Curso de Programación con PHP
  http://andromeda.mascarones.unam.mx/notas/php/
- Curso de Interacción de WWW con Bases de Datos
  http://andromeda.mascarones.unam.mx/notas/mysql/
- PostgreSQL
  http://www.potsgresql.org