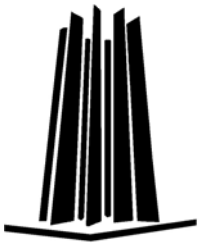
UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN

“INFORME DEL PROYECTO DEL SISTEMA DE CONTROL DE CERTIFICADOS DIGITALES EMITIDOS POR LA AUTORIDAD CERTIFICADORA DE LA UNAM EN SOFTWARE LIBRE”

T R A B A J O E S C R I T O
EN LA MODALIDAD DE SEMINARIOS Y CURSOS DE ACTUALIZACIÓN Y CAPACITACIÓN PROFESIONAL
QUE PARA OBTENER EL TÍTULO DE:
INGENIERO EN COMPUTACIÓN
P R E S E N T A:

G A B R I E L G O N Z Á L E Z G A R C Í A



ASESOR: ING. SILVIA VEGA MUYTOY

MÉXICO 2006

Agradecimientos

A mis padres y hermanos por darme ánimos y su ayuda incondicional en los momentos más difíciles.

A Lizbeth Barreto, Cristina Muzquiz y Roberto López por sus consejos en la elaboración de este trabajo.

Al equipo de Firma Electrónica de la UNAM por su apoyo en este proyecto.

A la Ing. Silvia Vega Muytoy por la orientación y guía brindada a lo largo de la planeación y desarrollo de este documento.

Índice

1. Informe General de la Línea de Especialización en Mantenimiento de Equipos de Cómputo

1.1 Taller de Mediciones Eléctricas y manejo de herramientas

1.1.1 Introducción

Actualmente el uso de los equipos de computo se ha vuelto muy común, diariamente tareas de nuestras actividades requieren el uso de esta herramienta. Al igual que el resto de los aparatos electrónicos, es necesario procurar darles mantenimiento para un correcto y óptimo desempeño.

Es de gran importancia conocer algunas de las especificaciones con las cuales nuestras computadoras trabajan como el voltaje, amperaje o potencia así como conocer de igual manera las técnicas para su medición e interpretación, identificación de sus elementos y sobre todo, poder precisar el origen de una falla y determinar si su origen se debió a causas de hardware o problemas con su cableado.

1.1.2 Parámetros Eléctricos

Como inicio de este módulo se observan las características de los parámetros eléctricos definidos en el Sistema Internacional de Unidades donde se especifican sus unidades. Estos parámetros son empleados por toda clase de dispositivos electrónicos, entre los que se encuentran las unidades mostradas en la tabla 1.0

Variable	Unidad	Símbolo	Equivalencia
Diferencia de potencial	Volt	V	J/c
Corriente Eléctrica	Amper	A	c/s
Potencia	Walts	W	J/s
Resistencia	Ohms	Ω	V/A
Frecuencia	Hertz	Hz	1/s

Tabla 1.0 – Unidades de Medición

Es necesario saber interpretar estas medidas cuando se intenta aislar las causas de un fallo en el equipo de cómputo para descartar problemas de alimentación o baja de potencia. Para ello hay que seguir las precauciones necesarias al tomar las lecturas del equipo para evitar dañar los diversos componentes.

1.1.3 Medidas de Seguridad

Es importante tener en cuenta las siguientes medidas de seguridad al reparar una computadora:

- El voltaje o corriente de algún componente puede causar daño o el voltaje o corriente de nuestro cuerpo puede dañar al componente.
- Al manejar elementos sensibles sería recomendable usar una pulsera antiestática o algo que permita hacer tierra.

- Al manejar instalaciones eléctricas sería mejor estar aislado de los elementos de la computadora y nuestra piel de cualquier conductor de tierra física.
- Si es posible, desconectar el elemento en revisión del suministro de energía.
- Procurar no usar artículos metálicos que puedan provocar un corto circuito como relojes, anillos, esclavas, etc.
- Tener precaución al manejar elementos como los capacitores que pueden contener grandes cantidades de voltaje a pesar de estar desconectados del suministro de energía.

1.1.4 Herramientas de Medición

Para medir los diferentes parámetros eléctricos se recurre al uso de múltiples herramientas, unas más especializadas que otras.

Entre los dispositivos más comunes se encuentra el multímetro, este puede ser analógico o digital, ambos son capaces de captar las mismas medidas pero en el multímetro analógico se presenta la probabilidad de un error en la toma de lectura por cuestiones de apreciación y posicionamiento.

El multímetro tiene tres modalidades de medición: voltímetro, amperímetro y ohmmetro. Algunos modelos agregan algunas otras funciones como probadores de diodos, medidores de continuidad entre otros.

Para realizar las mediciones de voltaje en una fuente de energía se deben localizar la punta negativa y la positiva, de igual manera, se localizan en el multímetro para obtener el valor correcto de la medición. Si se tomara la medición con las puntas invertidas se obtendría un valor negativo en el caso de usar un multímetro digital y en un multímetro analógico la aguja se colocaría hasta el tope.

1.1.5 Medidas de Seguridad con el Multímetro

Al realizar medidas con el multímetro se debe tener presente lo siguiente:

- Qué variable eléctrica se medirá, en base a esto, seleccionar la función correspondiente en el multímetro.
- Seleccionar la escala más alta de voltaje o corriente pera evitar descargas al multímetro, cuando se tenga la certeza del rango de medición, este podrá ir siendo ajustado.
- Si se esta midiendo resistencia nunca se debe hacer con energía.
- No realizar mediciones con aparatos que no tengan la escala suficiente.

1.1.6 Medición de Voltaje

Para realizar medidas de voltaje se debe identificar el tipo de voltaje del que se trata, para medir voltaje alterno (contactos) se coloca el multímetro en la función ACV o V~ en el rango inmediatamente superior a la que se pretende medir. Al medir voltaje alterno no importa la manera en que se coloquen las puntas.

Para las medidas de voltaje directo (voltaje de un eliminador, fuente o baterías) se coloca el multímetro en la función DCV o V- en el rango inmediatamente superior al

que se pretende medir. En este caso, la punta negra del multímetro debe conectarse a la Terminal negativa y la roja en la terminal positiva.

En ambos casos el multímetro debe conectarse en paralelo con respecto al circuito que se está midiendo.

1.1.7 Medición de Corriente

Para realizar mediciones de corriente se coloca la punta de color rojo del multímetro en el conector etiquetado como ***A*** o ***10A*** y elegir la función A en un rango inmediatamente superior al que se pretende medir. La punta negra del multímetro debe conectarse a la punta cortada del cable y la punta roja a la continuación del mismo.

Para realizar medidas de corriente el multímetro debe conectarse en serie con respecto al circuito que se está midiendo.

1.1.8 Medición de Resistencia

Para realizar mediciones de resistencia se debe elegir la función Ω. A continuación, la punta negra se coloca en uno de los extremos a medir y la roja en el otro extremo del cable o pista, según la escala aplicada y la resistencia del elemento. Se obtendrá la medición en ohms o un valor de cero si la conducción es total o un valor de uno si no hay conducción lo que puede sugerir que el cable o pista se encuentran dañados.

1.1.9 Conectores más comunes de la computadora

Conector	Descripción	Ilustración
Conector de video VGA (Video Graphics Adapter)	Es un conector BD15, el conector macho es empleado en el cable del monitor mientras el hembra se coloca en la tarjeta de video.	
Conector Dim o Mini-Dim	Se emplea para la conexión de teclados al CPU, la transmisión es serial.	
Conector PS/2	Se emplea para la conexión del Mouse al CPU, la transmisión es serial.	
Conector USB (Universal Serial Bus	En la nueva generación de los puertos serie. Actualmente es el más utilizado ya que tiene una buena velocidad de	

transmisión (USB1.1, 11Mbps; USB2.0, 480Mbps) y permite la conexión de 128 dispositivos a la vez

1.2 Herramientas de Software preventivo y correctivo

1.2.1 Introducción

El intercambio de información es muy común entre los equipos de computo, este se da desde el uso de unidades de almacenamiento secundario como discos de 3 1/2 o CD's hasta las redes internas de una empresa o el propio Internet. Esta práctica lleva un riesgo latente de que nuestro equipo de cómputo sea atacado por programas mal intencionados como virus o spyware.

Si este fuera el caso, se debe contar con los medios necesarios para diagnosticar y restaurar el equipo a su correcto funcionamiento así como poder recuperar la parte más importante que es la información.

1.2.2 BIOS

Una vez que se ha encendido la computadora se podrá verificar su configuración al entrar al BIOS (Basic Input/Output System), la manera en como se accesa a esta utilidad dependerá del fabricante de la tarjeta madre.

Dentro del BIOS se pueden encontrar diferentes secciones de acuerdo a la función de cada una de ellas. De entrada se vera la hora del sistema, la configuración de los periféricos conectados a ella como son discos duros y su configuración (maestro o esclavo), unidades de 3 1/2, CD-ROM, etc.

Una vez dentro del BIOS, habrá que buscar la sección de configuración de los dispositivos de arranque. Ahí se elegirá la forma en que la computadora iniciara, es decir, desde que unidad de almacenamiento se cargara el sistema operativo. De esta manera es posible volver a iniciar el sistema operativo en caso de que los archivos de arranque sean dañados a causa de un virus o una mala configuración.

En el caso de reemplazar o agregar un nuevo disco duro, hay que cerciorarse que la computadora lo ha reconocido con la capacidad de almacenamiento adecuada. Esto se puede lograr mediante la utilería de autodetección que puede venir dentro de la opción de configuración básica o bien, con una detección de discos IDE independiente. Al correr la utilería de auto detección se sugieren parámetros al usuario los cuales al ser aceptados pasan a ser parte de la configuración básica.

1.2.3 Formas de Arranque del Sistema Operativo

Normalmente Windows inicia cargando todos los archivos de inicio y de registro pero cuando el inicio del sistema falla, es común que este presente un menú de opciones

de arranque mismo que puede activarse presionando la tecla F8 al encender el equipo. Entre las opciones de arranque están:

- Modo Seguro.
- Modo Seguro con funciones de Red
- Modo Seguro con Símbolo del Sistema
- Ultima Configuración Buena Conocida

Cada uno de estos niveles carga una serie de módulos “paso a paso” para aislar problemas en caso de conflictos en el sistema a causa de una mala configuración.

1.2.4 Desfragmentador de disco duro y scandisk

A medida que se instalan o eliminan programas el rendimiento del sistema se ve disminuido.

Estas dos herramientas implementadas desde Windows 98 permiten mantener el sistema funcionando correctamente.

El comando scandisk permite crear y mostrar un informe del estado de un disco, basado en el sistema de archivos, también muestra y corrige los errores del disco. Actualmente el comando **chkdsk** es empleado en vez de scandisk.

Es recomendable ejecutar esta utilería después de que el sistema operativo se apaga sin hacer un correcto cierre del mismo, por ejemplo, cuando se va la luz y no hay un suministro eléctrico de emergencia como un UPS, la computadora se apaga sin que el sistema operativo pueda terminar con sus procesos, esto puede ocasionar inconsistencia en archivos y en los directorios donde se almacenan.

El desfragmentador de disco es una utilidad del sistema que permite analizar volúmenes (particiones del disco duro) locales, y encontrar y consolidar carpetas y archivos fragmentados, es decir, trata de colocar las localidades de memoria que forman los archivos más cerca unas de otras.

La ejecución de esta utilidad mejora considerablemente el desempeño del sistema operativo ya que reduce la cantidad de movimientos que los cabezales del disco duro tienen que hacer para leer las partes que conforman un archivo que se encuentra distribuido en varias zonas del disco duro.

1.2.5 Copia de seguridad

El programa Copia de seguridad ayuda a crear copias de la información del disco duro. Si por alguna razón se borraran o sobrescribieran accidentalmente los datos del disco duro o fuera imposible tener acceso a ellos por un error en el sistema de archivos, es posible utilizar esta copia para restaurar los datos perdidos o dañados.

1.2.6 Monitor del Sistema

El monitor de sistema permite llevar un registro de los recursos que acceden al CPU, disco duro, memoria y red. Este se encuentra dentro de las Herramientas del sistema.

En la consola del monitor se agregan los recursos del sistema de los que se necesita conocer su rendimiento y el estado de sus actividades.

Esta utilería permite localizar los problemas más frecuentes que son los que se presentan en la memoria y en el CPU.

1.3 Taller de Mantenimiento Correctivo

1.3.1 Introducción

A pesar de un correcto funcionamiento del software, no se está exento de las posibles fallas a nivel hardware que pudieran ocurrir debido a la falta de mantenimiento o bien, cuando algún componente se daña por sobrecargas de voltaje, mal manejo del mismo o simplemente porque su vida útil llego a su fin.

Hay que contar con las técnicas para determinar que pieza es la que esta fallando y realizar la reparación correspondiente si es que es posible o proceder con el reemplazo.

Realizar una evaluación del estado del hardware de forma acertada es de gran importancia para evitar molestos y costosos contratiempos debido a descomposturas en el hardware provocadas por situaciones ajenas a nuestro control como descargas eléctricas o fin de la vida útil del componente.

1.3.2 Consideraciones Previas

Al revisar un equipo de cómputo que ha sufrido una descompostura hay que tener en cuenta:

- Cuando y en que momento ocurrió el problema.
- Físicamente, en que lugar se encuentra.
- Tiempo de uso.
- Programas que tiene instalados.
- Averiguar si este problema tiene algún antecedente o es recurrente.
- Características.
- Frecuencia de uso del equipo.

1.3.3 Determinación del Origen de la falla

Para precisar más el origen del problema se comenzara por descartar razones de fallas partiendo de la más general a la particular:

- Defectos materiales.
- Problemas en el cableado de los componentes.
- Instalación incorrecta de expansiones (componentes agregados).
- Problemas en la configuración de los componentes en el BIOS.
- Conflictos de software.

Una vez que se tenga una idea más clara de la causa de la falla se puede tratar de reproducir para dimensionar su gravedad. Hay que evaluar si es posible realizar una reparación o si realmente sería más barato y práctico reemplazar la pieza dañada.

1.3.4 Criterios de Reemplazo

Realmente los criterios de reemplazo son pocos, la tendencia de los componentes a ser cada vez mas compactos deja pocas alternativas de reparación ante un alto costo por mano de obra, tiempo y esfuerzo. La modularidad con que una computadora esta ensamblada permite sustituir casi con toda libertad cualquier componente.

En todo caso, los factores determinantes para elegir una marca sobre otra son las particularidades operativas como la velocidad y tipo de bus para los discos duros, frecuencia de operación cuando se habla de módulos de memoria RAM o procesadores. Cuando una tarjeta integrada se daña o simplemente se quiere sustituirla por una mejor, hay que apegarse a las capacidades que el CHIPSET de la tarjeta madre soporta.

1.3.5 Causas de falla más comunes

Componente	Causa de falla
Fuente de poder	• Cables trozados o mal conectados causantes de un falso contacto. • Ventilador desbalanceado o fallo en su motor. • Fusibles quemados a causa de una descarga eléctrica. • Dispositivos electrónicos quemados (diodos, capacitores, resistencias, etc.)
Tarjeta madre, de sonido, video, etc.	• Uniones de soldadura mal realizados que pueden desconectarse o causar cortos circuito. • Componentes que se averían con el calor. • Líneas conductoras que se raspan impidiendo el paso de energía. • La falta de limpieza permite la acumulación de polvo provocando que la temperatura del dispositivo aumente. • Los conectores del borde de la tarjeta se corroen provocando falsos contactos. • Bahías de conexión dañadas en el caso de la tarjeta madre.
Módulos de memoria	• Conectores de la tarjeta dañados. • Fallo en los dispositivos electrónicos que la componen.
Disco Duro	• Desbalanceo de los discos a causa de un movimiento brusco o golpe. • Falla en el motor que lo hace girar. • Problemas en el suministro de energía.
Teclado y Mouse	• Cables trozados o pines doblados. • Botones atascados por falta de limpieza.

1.4 Introducción a las redes locales

1.4.1 Introducción

La computadora es una potente herramienta en el procesamiento y manejo de información, pero en la actualidad, una computadora aislada del mundo no es

suficiente para dar solución a las complejas tareas de los sistemas de hoy en día. La necesidad del intercambio de información entre distintos puntos de forma rápida, eficiente y sin procesos engorrosos, compartir los recursos entre equipos para reducir costos, dieron pie al nacimiento de las redes de computadoras.

1.4.2 Objetivo

Mostrar las características básicas de una red de computadoras, su funcionamiento y la manera en que esta puede ser implementada conforme a las condiciones físicas del lugar y sus componentes lógicos y físicos.

1.4.3 Contenido

Las redes en general, consisten en compartir recursos, y uno de sus objetivos es hacer que todos los programas, datos y equipo estén disponibles para cualquier equipo de la red que así lo solicite sin importar la localización física del recurso y del usuario. En otras palabras, el hecho de que el usuario se encuentre a miles de kilómetros de distancia de los datos, no debe evitar que este los pueda utilizar como si fueran originados localmente.

Se puede decir que una red la constituyen dos o más computadoras que comparten determinados recursos de hardware (impresoras, dispositivos de almacenamiento, etc.) o software (aplicaciones, archivos, datos).

1.4.4 Clasificación de las Redes

La clasificación de las redes se hace por su alcance geográfico, tecnología y forma de transmisión.

Por su alcance:

- Redes LAN (Local Area Network). Conectan varias computadoras dentro de la misma institución.
- Redes MAN (Metropolitan Area Network). Redes de tamaño superior a una LAN, soliendo abarcar el tamaño de una ciudad. Son típicas de empresas y organizaciones que poseen distintas oficinas repartidas en una misma área metropolitana.
- Redes WAN (Wide Area Network). Tienen un tamaño superior a una MAN, y consisten en una colección de redes LAN conectadas por una subred.

Por su la tecnología de transmisión:

- Broadcast. Aquellas redes en las que la transmisión de datos se realiza por un sólo canal de comunicación, compartido entonces por todas las máquinas de la red. Cualquier paquete de datos enviado por cualquier máquina es recibido por todas las de la red.
- Point-To-Point. Aquellas en las que existen muchas conexiones entre parejas individuales de máquinas. Para poder transmitir los paquetes desde una máquina a otra a veces es necesario que éstos pasen por máquinas intermedias, siendo obligados en tales casos un trazado de rutas mediante dispositivos routers.

Por su tipo de transferencia de datos que soportan:

- Transmisión simple. Son aquellas redes en las que los datos sólo pueden viajar en un sentido.
- Half-Duplex. Aquellas en las que los datos pueden viajar en ambos sentidos pero sólo en uno de ellos en un momento dado, es decir, sólo puede haber transferencia en un sentido a la vez.
- Full-Duplex. Aquellas en las que los datos pueden viajar en ambos sentidos a la vez.

1.4.5 Topologías de Red

Los diferentes componentes que van a formar una red se pueden interconectar o unir de diferentes formas, siendo la forma elegida un factor fundamental que va a determinar el rendimiento y la funcionalidad de la misma.

La disposición de los diferentes componentes de una red se conoce con el nombre de **topología de red**. La topología idónea para una red concreta va a depender de diferentes factores, como el número de máquinas a interconectar, el tipo de acceso al medio físico, etc.

Se pueden distinguir dos aspectos diferentes a la hora de considerar una topología, el primero de ellos es la disposición real de las máquinas, dispositivos de red y cableado (los medios) en la red y la segunda es la forma en que las máquinas se comunican a través del medio físico.

1.4.5.1 Topología de Bus

La topología de bus tiene todos sus nodos conectados directamente a un enlace y no tiene ninguna otra conexión como se muestra en la figura 1. Físicamente cada computadora está conectada a un cable común, por lo que se pueden comunicar directamente, aunque la ruptura del cable hace que los equipos queden desconectados.

Figura 1 – Topología de Bus

Es común que se produzcan problemas de tráfico y colisiones, que se pueden atenuar segmentando la red en varias partes.

1.4.5.2 Topología de Anillo

Una topología de anillo se compone de un sólo anillo cerrado formado por nodos y enlaces, en el que cada nodo está conectado solamente con los dos nodos adyacentes como se muestra en la figura 2.

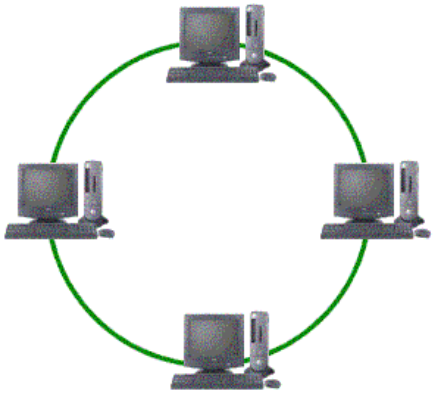


Figura 2 – Topología de Anillo

Para que la información pueda circular, cada estación debe transferir la información a la estación adyacente.

1.4.5.3 Topología de Estrella

La topología en estrella tiene un nodo central desde el que se conectan todos los enlaces hacia los demás nodos como se muestra en la figura 3. Por el nodo central, generalmente ocupado por un concentrador, pasa toda la información que circula por la red.

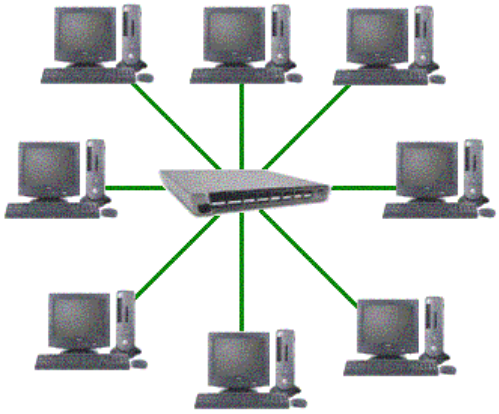


Figura 3 – Topología de Estrella

La ventaja principal es que permite que todos los nodos se comuniquen entre sí de manera conveniente. La desventaja principal es que si el nodo central falla, toda la red se desconecta.

1.4.5.4 Topología en Malla Completa

En una topología de malla completa, cada nodo se enlaza directamente con los demás nodos como se muestra en la figura 4. Las ventajas son que, como cada todo se conecta físicamente a los demás, creando una conexión redundante, si algún enlace deja de funcionar la información puede circular a través de cualquier cantidad de enlaces hasta llegar a destino. Además, esta topología permite que la información circule por varias rutas a través de la red.

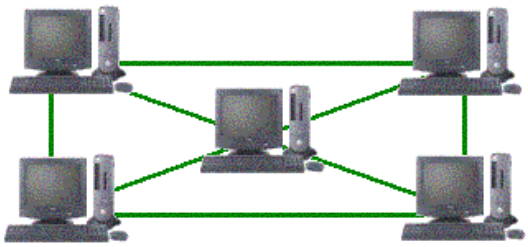


Figura 4 – Topología en Malla Completa

La desventaja física principal es que sólo funciona con una pequeña cantidad de nodos, ya que de lo contrario la cantidad de medios necesarios para los enlaces, y la cantidad de conexiones con ellos se torna sumamente compleja.

1.5 Actualización del equipo de Cómputo

1.5.1 Introducción

Cuando se navega en Internet o se intercambia información por medio de discos extraíbles es muy común encontrar un virus o un programa malicioso que puede dañar información o perjudicar el desempeño del equipo de cómputo.

A pesar de contar con los medios para detectar y detener estos programas, se puede decir que son soluciones momentáneas, el constante desarrollo de los sistemas de seguridad trae de la mano el desarrollo de virus cada vez más sofisticados y destructivos. Además de la actualización del antivirus de igual manera es recomendable estar al pendiente de las actualizaciones de seguridad del propio sistema operativo para cerrar huecos de seguridad.

La mayor parte de las actualizaciones de contra virus están enfocadas al sistema operativo Windows. Cada fabricante de software antiviral o firewalls liberan actualizaciones de lo que se conoce como definiciones de virus, estas son las respuestas que pueden dar contra el ataque de estos programas.

Todas estas distribuciones permiten establecer una configuración sobre la frecuencia con que se descargan las últimas actualizaciones. Otra buena costumbre es revisar los boletines de seguridad que contienen los detalles de vulnerabilidades en el sistema operativo que permiten intrusiones o comportamientos anormales, estos son publicados por Microsoft en el caso de Windows. Desgraciadamente, algunas de las versiones de los sistemas operativos han sido descontinuadas en lo que respecta a las actualizaciones como es el caso de Windows 95, 2000 y ME.

De igual manera, Microsoft ofrece una herramienta de actualización llamada WUS por sus siglas en inglés (Windows Software Update Services). Esta utilería permite obtener las últimas actualizaciones, elegir al grupo de equipos a los que se aplicaran las actualizaciones en caso de estar trabajando en una red, manejar bitácoras, etc.

La actualización de los controladores de los diferentes dispositivos de la computadora también serán de gran ayuda para mejorar su desempeño, sólo basta entrar al sitio Web del fabricante e ingresar el modelo del componente y elegir el sistema operativo en uso.

1.6 Conclusiones

El rendimiento y la vida útil de nuestro equipo de cómputo está en función de los componentes que elegidos para ensamblarla y su marca. La prevención de las perdidas de información o costosas composturas son problemas que pueden ser evitados antes de sufrir las consecuencias si se siguen las sencillas normas de limpieza en el aspecto físico y la instalación de programas que ayuden a estar al tanto del funcionamiento del sistema operativo.

Una instalación adecuada refleja resultados óptimos cuando estas se complementan con las últimas actualizaciones para mejorar su desempeño o disminuir las probabilidades de falla.

En caso de una falla, algunas causas pueden ser excluidas para determinar el origen de este y continuar con una reparación de ser posible. Hay que tener en cuenta que por lo general es más práctico y barato la sustitución del dispositivo que intentar una reparación.

2. Informe General del Diplomado de Desarrollo e Implementación de Software libre en Linux

2.1 Introducción al Sistema Operativo Linux

2.1.1 Breve Historia

La historia de LINUX inicia con el desarrollo de un pequeño programa llamado Minix, un sistema operativo escrito por el científico Andrew Tannebaum, este sistema ganó popularidad siendo desarrollado para poder trabajar en diferentes plataformas de equipo de cómputo lo que inspiro al desarrollo de Linux.

En la década de los 60´s cuando todo el software se mantenía en una arquitectura cerrada dictada, es decir, que ciertos elementos de los programas no eran entregados al usuario final como el código fuente para hacer modificaciones o la distribución del programa era casi a la medida del hardware en el que se pretendía instalar la aplicación. Los Laboratorios Bell de AT&T en conjunto con GE trabajaron en el Instituto de Tecnología de Massachussets en un proyecto llamado MULTICS, un antiguo sistema de tiempo compartido el cual tenia como finalidad proporcionar un soporte para diferentes arquitecturas de hardware compartiendo los recursos de hardware y software y así poder eliminar el concepto de arquitectura cerrada.

Kenneth Thompson y Dennis Ritchie desarrollan un lenguaje de alto nivel con gran interacción con hardware que más tarde emplearían para el desarrollo de UNIX, esta versión de sistema operativo permitía la portabilidad entre distintas arquitecturas de hardware.

En 1990 un estudiante Finlandés, Linus Torvalds, inicio un proyecto personal basado en Minix, un sistema operativo tipo UNIX, su intención era desarrollar un sistema compatible con plataforma PC el cual fuera más confiable. Para 1991 libera la primera versión de Linux sin ser una distribución oficial, la 0.01

En el mismo año pone a disposición del mundo a través de Internet la primera versión oficial de Linux, la 0.02

2.1.2 Características de Linux

- **Multiusuario**. Soporta la actividad de más de un usuario al mismo tiempo
- **Multitarea.** Puede realizar procesos simultáneos.
- **Modular.** El sistema entero se compone de programas más pequeños que al interactuar entre si, logran realizar una tarea.
- **Portable entre sistemas.** Los programas desarrollados en una computadora pueden ser ejecutados en otra.
- **Multiplataforma.** Puede ser instalado en diferentes arquitecturas de hardware.
- **Programable.** Las distribuciones incluyen el código fuente lo que permite programar módulos según nuestras necesidades.

2.1.3 Distribuciones más comunes

- Slackware
- Red Hat

- Debian
- Mandrake
- Suse
- FreeBSD
- Turbolinux
- Knoppix

2.1.4 Estructura del Sistema Operativo Linux

Básicamente, Linux está compuesto básicamente de cuatro capas, la capa más interna esta conformada por el hardware, que es el conjunto de piezas físicas del equipo de cómputo.

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

La tercera capa está conformada por el grupo de los Shells o interpretes de comandos, los cuales funcionan como la interfaz entre el usuario y el kernel, proporcionan las herramientas para que el usuario se pueda comunicar con el núcleo del sistema.

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

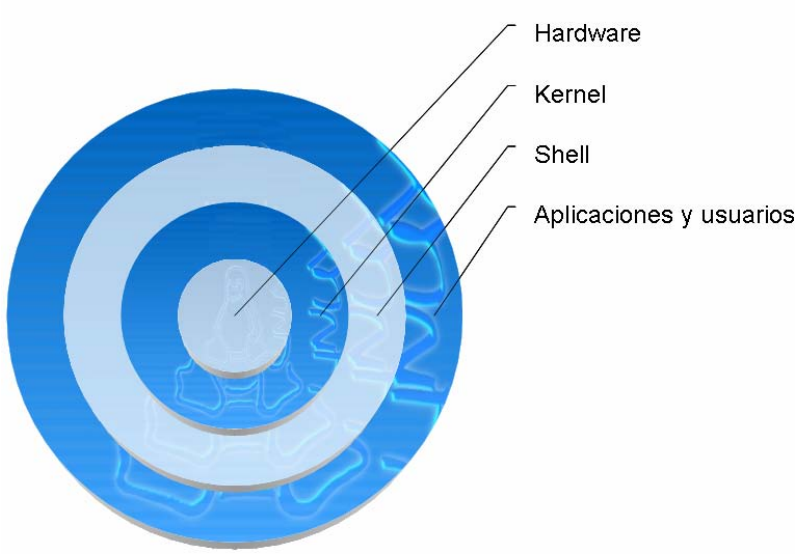


Figura 1 - Capas del Sistema Operativo Linux

2.1.5 Sistemas de Archivos

Linux está compuesto de un sistema de archivos jerárquicos en donde los directorios, ligas (accesos directos), unidades, etc. Son manejados como archivos.

Existe sólo una raíz en el sistema la cual representa la parte más alta de la estructura de directorios y es conocida como ***root*** o raíz, es representada por el símbolo de ***/*** a partir de este punto se desprenden diferentes ramas de directorios.

Entre los directorios más importantes se encuentran:

/	Raíz del sistema
/var	Variables del sistema
/dev	Dispositivos de hardware
/home	Directorios de usuarios
/bin	Comandos del sistema
/sbin	Comandos del sistema que sólo el administrador del equipo puede utilizar
/mnt	Dispositivos montados

2.1.6 Cuentas de Usuario

Una cuenta de usuario define las acciones que ese usuario puede llevar a cabo dentro del sistema operativo.

Básicamente, una cuenta de usuario tiene dos elementos: login y contraseña. El login es el identificador del usuario dentro del servidor, dicho login es único, es decir, no puede haber dos cuentas en el servidor con el mismo login ya que éste es el nombre con que se conoce al usuario. La contraseña es la clave de acceso al sistema, sólo debe ser conocida por el dueño de la cuenta ya que permite el acceso al servidor, a todos sus servicios disponibles y a su información.

La manera de expresar una cuenta de usuario en Linux es la siguiente.

usuario@fobos.unam.mx

Donde *usuario* es el nombre del usuario asignado, *fobos* es el nombre del servidor donde se encuentra esa cuenta que pertenece a la UNAM.

2.1.7 Salida del Sistema

Cuando se emplea una sesión de secure shell[1] para conectarse a un servidor, es necesario indicarle al sistema que es necesario terminar la sesión de trabajo, para ello existen varios comandos, entre ellos se encuentran los comandos *exit* y *logout* o la combinación de teclas *ctrl+d*.

2.1.8 Comandos Básicos

Cuando se trabaja en la shell de Linux, es común asignarle al sistema alguna actividad que procese cierta información de alguna forma en particular, la manera en como se hace es a través de los comandos.

[1] Su funcionamiento es similar al de una sesión en telnet. Secure Shell permite ejecutar comandos, trabajar con archivos, copiarlos, etc. de una manera segura ya que posee fuertes métodos de autenticación además de que la comunicación entre dichos equipos de hace de forma cifrada.

Un comando es una instrucción que el sistema operativo reconoce e interpreta realizando la tarea que tiene asociada a dicho comando.

La estructura de los comandos es muy rígida y generalmente siguen la misma sintaxis, es decir, siempre se escribe primero el comando, en seguida las opciones si se requieren y al final los argumentos necesarios.

Las *opciones* modifican el funcionamiento del comando, generalmente se utilizan para agregar información a la salida o aumentar su potencial, estas siempre van antecedidas por un guión. Las opciones no son requeridas así que el comando puede ser ejecutado sin ellas.

Los *argumentos* son requeridos para el funcionamiento de los comandos, si estos son omitidos la shell devolverá un mensaje de error.

Entre los más indispensables se encuentran:

ls	Lista el contenido de los directorios
pwd	Imprime en pantalla el directorio de trabajo actual
mkdir	Crea directorios
rmdir	Elimina directorios vacíos
rm	Elimina archivos. Con la opción –r elimina directorios que aún contienen archivos
cd	Permite el cambio entre directorios
cp	Copia archivos
find	Encuentra archivos que cumplan con determinado criterio de búsqueda
mv	Mueve archivos a otras direcciones
man	Muestra los archivos de ayuda sobre cualquier comando

2.1.9 Archivos de texto

Para visualizar el contenido de archivos se cuenta con múltiples comandos, estos se diferencian por la manera en que permiten desplegar la información e interactuar con ellos:

cat	Permite ver el contenido de uno o más archivos, este comando lee imprime el contenido del archivo en pantalla sin hacer ninguna pausa.
more	Lee el archivo y muestra su contenido de forma paginada.
less	Funciona de la misma manera que more, sólo que less permite la navegación en el archivo de atrás hacia delante y viceversa.
head	Muestra las 10 primeras líneas de un archivo. Con la opción –n se puede especificar el número de líneas que se desea mostrar.
tail	Muestra las 10 últimas líneas de un archivo. Con la opción –n se puede especificar el número de líneas que se desea mostrar.

2.1.10 Editores de Texto

De igual manera, Linux cuenta con editores de texto para editar su contenido:

2.1.10.1 Vi

El editor *vi* es un editor de texto de pantalla completa que maneja en memoria el texto entero de un archivo. Es el editor clásico de UNIX y Linux; está en todas sus versiones. Puede usarse en cualquier tipo de terminal con un mínimo de teclas.

2.1.10.2 Emacs

Emacs es un editor de texto gráfico altamente adaptable a las necesidades de los usuarios, es muy útil en el desarrollo de códigos. Este editor no viene incluido en la instalación por defecto del sistema operativo.

2.1.10.3 Pico

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

2.1.11 Redireccionamientos

En Linux como en otros sistemas operativos la salida estándar del resultado de las operaciones de los comandos es el monitor, de igual manera, la entrada estándar para estos comandos es el teclado.

Es posible modificar este comportamiento por medio de los *redireccionamientos* de los flujos estándar. Existen los redireccionamientos para la entrada y para la salida.

El carácter mayor que (**>**) permite redireccionar la salida de un programa hacia un archivo, por ejemplo, para enviar la salida del comando *ls* a un archivo llamado *listado.txt* se haría de la siguiente forma

```
ls > listado.txt
```

De este modo, el resultado del comando no se imprime en pantalla sino en el archivo. Si el archivo especificado no existe se crea y si existía previamente entonces se sobrescribe. A este tipo de redireccionamiento se le conoce como destructivo.

Para realizar esta misma operación conservando el contenido del archivo y agregando el resultado del comando al final, la forma de hacerlo seria la siguiente:

```
ls >> listado.txt
```

El carácter menor que (**<**) permite redireccionar un archivo hacia un programa, por ejemplo, para enviar el contenido del archivo *listado.txt* por correo al usuario con el login *usuario* se haría de la siguiente manera:

```
mail usuario < listado.txt
```

2.1.11.1 Redireccionamiento de un programa a otro

Intrínsecamente, los comandos de la shell son potentes herramientas que pueden ayudar al manejo de la información. La shell también da la opción de conjuntar todos estos comandos en una tarea en particular como filtrar la información de las bitácoras del sistema por fechas o de un listado sólo traer los registros que cumplan con ciertos criterios ordenados alfabéticamente.

Para esto se emplea el carácter *pipe* (|) o tubería que envía la salida estándar de un comando a la entrada estándar de otro ejemplo:

ls | more

Por lo general se usan más de dos comandos entrelazados por pipes para la depuración de la información, al comando que se encuentra en medio de otros dos se le conoce como ***filtro***.

2.1.11.2 Redireccionamiento Condicional

Como se menciono anteriormente, hay una salida estándar, esta salida contendrá el resultado de un comando que se ejecuto de forma exitosa. De igual manera, existe una salida estándar para el manejo de errores que se provoquen durante la ejecución de algún comando. A la salida estándar le corresponde el número *1* mientras que al error estándar le corresponde el número *2*.

2.1.12 Atributos de los archivos

Dentro del sistema operativo se encuentran archivos de diferentes tipos, los cuales son distinguidos por el primer bit de sus propiedades. Este bit indica si se trata de un directorio o un archivo y que tipo de archivo es.

Entre los archivos más comunes se tienen:

d	Directorio
-	Archivo ordinario. Puede ser un archivo de texto, un archivo binario, etc.
l	Liga suave
c	Archivo de tipo carácter (Archivos de acceso de hardware mediante un bit como el mouse).

2.1.13 Permisos

Los permisos especifican “quien puede hacer que cosa”, los permisos aplican tanto para archivos como para directorios. Existen tres tipos de permisos:

r	Permiso de lectura. Permite las operaciones de copiado y lectura del archivo o directorio
w	Permiso de escritura. Permite la modificación o eliminación del contenido del archivo o directorio.

x	Permiso de Ejecución. Permite que un archivo como un programa realice su proceso.

La asignación de permisos se aplica a tres categorías diferentes: al propietario del archivo o directorio, al grupo al que pertenece dicho usuario y finalmente, al resto de los usuarios del sistema.

Por tal motivo, los atributos se componen de diez bits: el primer bit indica el tipo de elemento que esta listado (archivo o directorio), los siguientes 9 bits representan los tipos de permisos asignados a cada elemento listado.

2.1.13.1 Manejo de Permisos

El comando ***chmod*** permite modificar los permisos a los archivos y directorios. La asignación de permisos puede hacerse a través del método simbólico o mediante el método octal.

chmod opera sobre la siguiente lista de indiadores cuando se trabaja de forma simbolica:

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

Para modificar los permisos se pone el indicador de los permisos a afectar ya sea para el usuario, el grupo o los otros, seguido del signo por el signo más (+) si se quiere agregar el permiso o de menos (–) si se quiere quitar separando con una coma (,) los bloques de permisos, al final se indica el nombre del elemento a afectar.

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

x	1
w	2
r	4

En base a lo anterior se obtendría la siguiente tabla

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

2.1.14 Procesos

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

Todos los procesos tienen un dueño, este dueño es el usuario quien inicio el proceso desde el interprete de comandos, de tal forma que ningún otro usuario puede afectar las tareas que ese usuario esta realizando.

Cuando se inicia una sesión dentro del servidor en realidad se ejecuta una copia del interprete de comandos que se tiene por defecto, a esta copia del interprete se le asigna un número el cual esta asociado al proceso en ejecución (identificador de proceso o PID), por cada sesión que sea iniciada se ejecutara una nueva copia del shell asignándole un nuevo número de identificación.

Es importante recalcar que todo proceso tiene una dependencia del shell principal del sistema, en el caso de un usuario en particular, todos los procesos que se generen dependerán de la copia del shell que se este ejecutando en ese momento.

Una vez que se esta ejecutando un programa, este puede ser enviado a segundo plano presionando la combinación de teclas ctrl+z.

El comando ***jobs*** permite ver los programas que se están ejecutando en segundo plano, nótese que a cada proceso en segundo plano se le asigna un número que sirve para identificarlo.

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

Suponiendo que quisiera buscar el archivo *notas.txt* desde la raíz del sistema y cuando lo encuentre, envíe su ubicación a un archivo llamado *encontrado.txt* y los errores generados a un archivo llamado *errores.txt*. Quizá este proceso tarde un poco así que se ejecutara en segundo plano. El comando quedaría de la siguiente manera:

```
find / -name notas.txt 1>encontrado.txt 2>errors.txt &
```

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

Para traer un proceso de vuelta a primer plano se ingresa la instrucción ***fg*** **#** donde el # representa el número que el comando *jobs* devuelve.

El comando *kill* permite entre otras opciones terminar procesos. Como usuarios únicamente es posible terminar los procesos que perteneces al usuario que ejecuto el comando.

2.1.15 Ligas

Existen dos tipos de ligas, una de ellas es la de tipo suave y la otra es de tipo dura.

2.1.15.1 Liga suave

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

Sintaxis de la liga suave es:

ln –s nombre_archivo nombre_liga

El comando ln permite generar las ligas, en este caso con la opción –s se indica al sistema que se esta generando una liga de tipo suave.

2.1.15.2 Liga dura

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

Sintaxis de la liga dura es:

ln nombre_archivo nombre_liga

En este caso se esta generando una liga dura, al aplicar el comando ln sin opciones el sistema interpreta el comando como la creación de una liga dura.

2.2 Instalación y Administración de Linux

Para atender las cuestiones de administración Linux incorpora herramientas para el mantenimiento de cuentas de usuario, organización del sistema de archivos, instalación y actualización de software, creación de respaldos, etc.

2.2.1 Actividades del Administrador

Antes de inicial la instalación del sistema operativo se debe pensar en que tipo de tareas se empleara, dado que hay configuraciones predefinidas de instalación como servidor o estación de trabajo, cada uno incluye aplicaciones distintas, por ejemplo, no tendría mucho sentido instalar herramientas de desarrollo en un servidor de correo junto con una interfaz gráfica o en el caso contrario, una estación de trabajo no sería de mucha ayuda sin los paquetes de desarrollo.

Una de las tareas más importantes del administrador del sistema es tener una serie de respaldos dependiendo de la misión del servidor, respaldos atrasados no son muy útiles en situaciones de contingencia. Esta serie de respaldos pueden ser manejados como procesos calendarizados. Es aconsejable que estos respaldos se almacenen en particiones distintas o de ser posible, en equipos distintos.

El registro de los eventos en el sistema tiene un papel fundamental en la determinación de orígenes de fallas en el sistema, comportamiento de los usuarios

para detectar de forma oportuna intentos de actividades no autorizadas o en el análisis forense de sistemas en que su seguridad ha sido vulnerada.

2.2.2 Archivos de Arranque del sistema

Después de que el sistema operativo arranca y monta el sistema de archivos de root, el primer programa que ejecuta el sistema es ***init***. Este programa es el encargado de lanzar los scripts de inicialización del sistema y de modificar el sistema operativo de su estado inicial de arranque al estado estándar multiusuario. También define los intérpretes de órdenes ***login*** de todos los dispositivos tty del sistema y especifica otras características del arranque y apagado.

Una vez completada la secuencia de arranque, *init* permanece ejecutándose en segundo plano, "monitoreando" la ejecución del sistema. Todas las tareas que realiza se definen en el archivo ***inittab*** que se encuentra en el directorio */etc*.

El programa init pasa a través de una serie de ***niveles de ejecución***, que corresponden a varios estados del sistema. Al nivel de ejecución 1 se entra inmediatamente después de iniciar el sistema, los niveles de ejecución 2 y 3 son los modos de operación del sistema normal y multiusuario respectivamente, el nivel de ejecución 4 lanza el sistema X Window a través del X display manager ***xdm*** y el nivel de ejecución 6 reinicia el sistema. Los niveles de ejecución asociados a cada orden, son el segundo término de cada línea del archivo /etc/inittab.

2.2.3 Mantenimiento de cuentas de Usuario

La correcta clasificación de usuarios permite llevar un control sobre las cuentas que pueden realizar determinadas actividades dentro del sistema. Los usuarios son colocados en grupos y a estos grupos se les asignan privilegios, de esta forma es más sencillo asignar permisos a un conjunto de usuarios con características similares a llevar el control de cada uno de ellos por separado. Un usuario puede pertenecer a más de un grupo.

Los comandos para la administración de cunetas de usuario son:

useradd Crea un nuevo usuario o actualiza a un usuario existente

La sintaxis de useradd es la siguiente:

useradd [opciones]

Entre las principales opciones de *useradd* están:

-g	Especifica el grupo inicial. Este grupo debe existir previamente
-d	Especifica el directorio inicial del usuario conocido como *home*.
-G	Especifica una lista de grupos adicionales de los que el usuario también es miembro.

groupadd Crea un nuevo grupo

La sintaxis de *groupadd* es la siguiente:

groupadd [opciones] nombre_del_grupo

Entre las principales opciones de *groupadd* están:

-g Especifica el identificador del grupo que se va a crear, este valor numérico deberá ser único.

userdel Elimina a un usuario junto con todos los archivos relacionados con él.

La sintaxis de *userdel* es la siguiente:

Userdel [-r] nombre_del_usuario

Donde la única opción de este comando especifica que los archivos del usuario serán eliminados junto con su directorio inicial y su bandeja de mensajes de correo.

Se recomienda que el usuario cambie su contraseña de forma periódica para procurar una mayor seguridad en el sistema. Todos los usuarios pueden hacerlo a través del comando ***passwd***.

2.2.4 Sistemas de Archivos

Como ya se mencionó, la información contenida en Linux es almacenada y organizada en directorios. Antes de poder comenzar a escribir estos archivos se debe dar formato al medio de almacenamiento. Este formato especificara el tamaño de la unidad de almacenamiento.

Entre los sistemas de archivos más comunes para Linux se encuentran:

2.2.4.1 EXT2 (Second Extended File System)

Fue el sistema de archivos estándar de Linux por varios años, tiene una unidad similar al cluster, llamada bloque, y que es, por lo general de 1Kilobyte, que puede ser especificado por el usuario e independiente del tamaño de la partición, lo cual asegura un buen aprovechamiento del espacio libre con archivos pequeños.

Ext2 usa 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.

La principal desventaja de EXT2 es que no posee una bitácora, por lo que muchos de sus usuarios están emigrando a ReiserFS y su sucesor ext3.

2.2.4.2 EXT3 (Third Extended File System)

Es un sistema de archivos con registro por diario. 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. Un sistema de archivos ext3 puede ser montado y usado como un sistema de archivos ext2.

2.2.4.3 ReiserFS

Es un sistema de archivos de propósito general, diseñado e implementado por un equipo de la empresa Namesys, liderado por Hans Reiser Actualmente es soportado por Linux y existen planes de futuro para incluirlo en otros sistemas operativos como sistema de archivos base como es el caso de Slackware, SuSe,

Entre las características más sobresalientes se encuentra el Journaling que previene la corrupción del sistema de archivos por cierres inesperados del sistema operativo, reparticionamiento del sistema de archivos cuando este se encuentra montado o desmontado.

Comparado con ext2 y ext3en el uso de archivos menores de 4k, ReiserFS es normalmente más rápido. Esto proporciona una elevada ganancia manejo de cache para servicios HTTP, agentes de correo y otras aplicaciones en las que el tiempo de acceso a archivos pequeños debe ser lo más rápida posible.

2.2.5 Instalación del Sistema operativo

En esta sección se realizara la instalación de la distribución de Slackware 10 de Linux, esta distribución posee cualidades que están presentes en la mayoría de las demás de distribuciones de Linux.

Slackware es una distribución GNU de Linux que es desarrollada y mantenida por Patrick Volkerding.

Linux es distribuido de forma gratuita y puede ser descargado del sitio http://www.linuxiso.org.

2.2.5.1 Iniciando la Instalación

Una vez que se ha descargado y quemado la imagen del disco en un CD, hay que reiniciar la computadora con el disco de Slackware. Una ventana de preinstalación aparecerá para preguntar por el kernel que se utilizara para la instalación. En este caso se elegirá el kernel ***bare.i*** que es el empleado por defecto.

A continuación se seleccionara el idioma de nuestro teclado, después de eso, se ingresa al sistema como ***root***, el sistema no pedirá ningún password.

La instalación de Slackware requiere al menos una partición de Linux y una partición de intercambio tipo ***swap***. Para realizar las particiones necesarias para la instalación, Linux ofrece dos herramientas: ***fdisk*** y ***cfdisk***. La diferencia entre estas herramientas es que cfdisk es más sencillo en su manejo gracias a una interfaz de menú.

Para particionar un disco simplemente se escribe el nombre del comando seguido por el nombre del dispositivo, por ejemplo, para formatear el disco duro que esta conectado como maestro en el primer IDE de la siguiente forma:

fdisk /dev/hda

En el supuesto de que este disco duro tuviera una partición hay que hacer referencia a ella como */dev/hda1*.

Las particiones son creadas en el espacio disponible del disco duro, esta debe ser una partición primaria. El tipo de partición Linux es especificado por defecto. Para la creación de la partición swap involucra los mismos pasos para la partición nativa de Linux pero el tipo de partición especificado para la partición deberá ser ***Linux Swap***. Se recomienda que el tamaño de la partición swap sea del doble de nuestra memoria RAM.

Si las particiones están listas, se guarda la información en la tabla de particiones y se sale de la herramienta fdisk.

Una vez que todo esta en orden, se ejecuta el comando ***setup*** para comenzar la instalación.

El programa Setup contendrá el menú mostrado en la figura 2

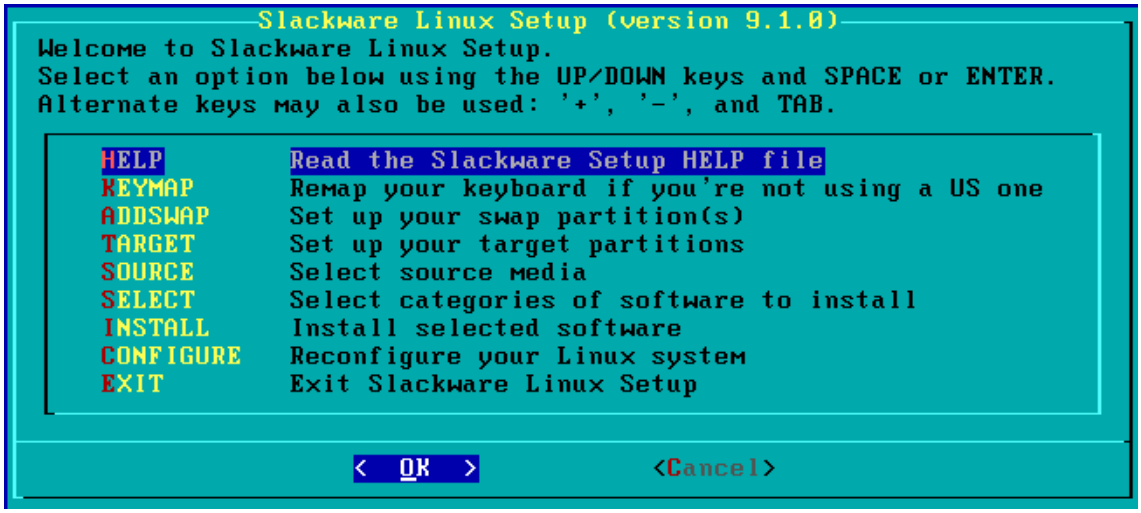


Figura 2 – Menú de Instalación de Slackware

La opción ***Addswap*** agrega la partición elegida como swap. La utilería preguntara si es necesario dar formato a la partición a lo que se contestara que si.

Después de formatear la partición swap, el menú correspondiente a ***Target*** será lanzado. En este menú se elegirá la partición o particiones que se montaran para la instalación de Slackware como se muestra en la figura 3.

Se elije la opción de formateo rápido y el sistema de archivos por defecto es ReiserFS y se hace clic en Aceptar.

La primer partición automáticamente se carga como root (/). Para otras particiones el punto de montaje puede ser seleccionado. Tener particiones separadas para /, /var,

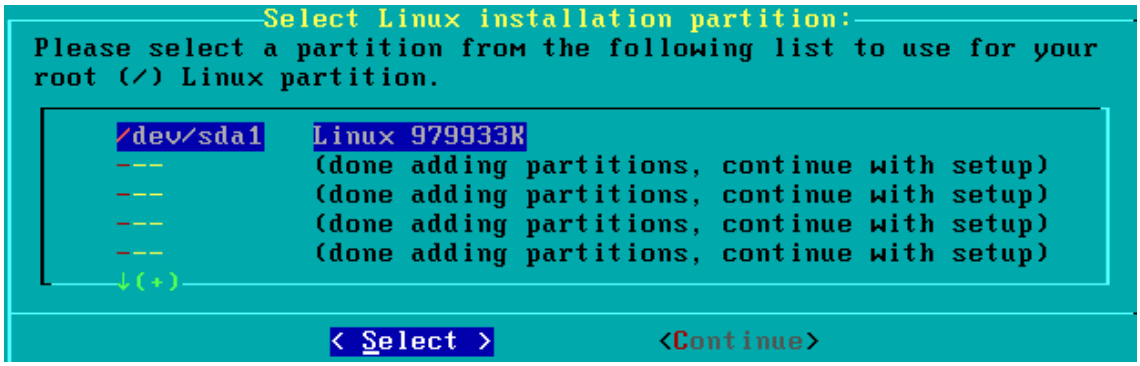


Figura 3 – Selección de la partición o particiones a montar
para la instalación de Slackware

/usr y /home provee una mejor protección a los archivos en caso de presentarse una catástrofe en el sistema operativo, así si es necesario realizar alguna reparación esta se hará sobre la partición correspondiente y no sobre todo el sistema de archivos.

Una vez hecho lo anterior, el siguiente paso es seleccionar el medio desde el que se hará la instalación. La opción utilizada para este caso es ***Install from a Slackware CD or DVD***.

Después de elegir el medio desde el cual se instalara el sistema, se escogerán los paquetes que se instalaran junto con el resto del sistema operativo.

La siguiente pantalla mostrara tres tipos de instalaciones: completa, menú y experto. La opción *completa* es la más sencilla, esta instalará todos los paquetes que incluye el CD, la desventaja es que esto ocupara más espacio del disco duro. La opción de *menú* permite seleccionar que paquetes se instalaran. La opción *experto* funciona de de la misma forma que menú, a diferencia que en modo experto es posible no instalar paquetes importantes.

Después de completar la instalación, la herramienta de *setup* mostrara una ventada donde se seleccionara el medio desde que se instalara el Kernel. Para este caso se elije la opción ***cdrom*** como se muestra en la figura 4.

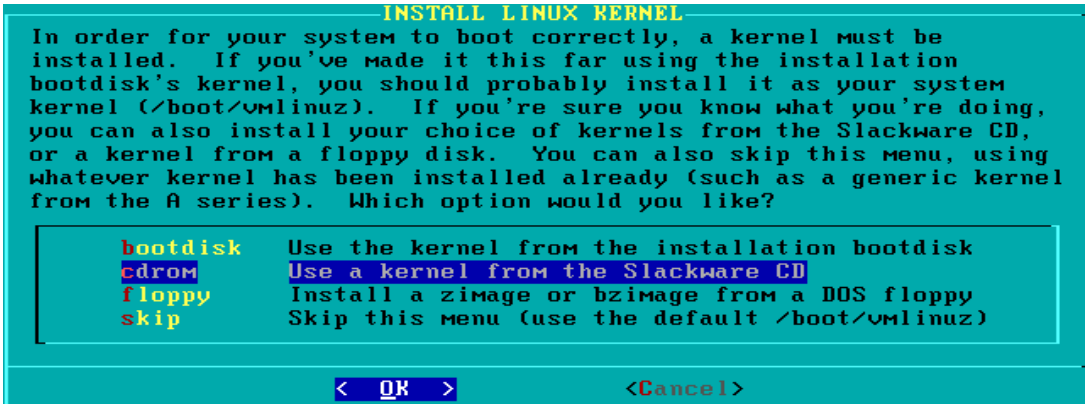


Figura 4 – Instalación del Kernel

En este punto, la herramienta de instalación ofrecerá la opción de hacer un disco de inicio. Es recomendable contar con un disco de inicio en caso de que la configuración del gestor de arranque (lilo) falle.

Las siguientes pantallas mostraran los pasos necesarios para la instalación del gestor de arranque ***lilo*** (**Li**nux **Loador**) el cual permitirá seleccionar el sistema operativo con el que la computadora arrancara en caso de tener más de uno.

Del menú mostrado en la figura 5 se elije la opción de instalación ***simple***.

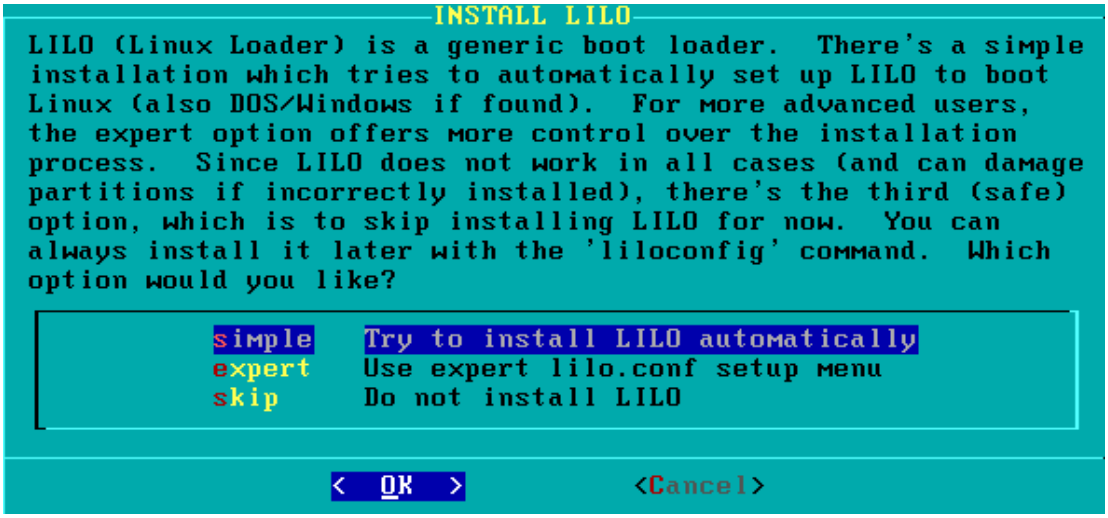


Figura 5 – Menú de Instalación de Lilo

Después de seleccionar la opción de instalación simple, la utilería de instalación de lilo permitirá habilitar un buffer para el frame como se muestra en a figura 6. Este buffer permitirá utilizar distintas resoluciones para lilo con dimensiones diferentes a las que se usan por defecto que son de 80 x 25 caracteres.

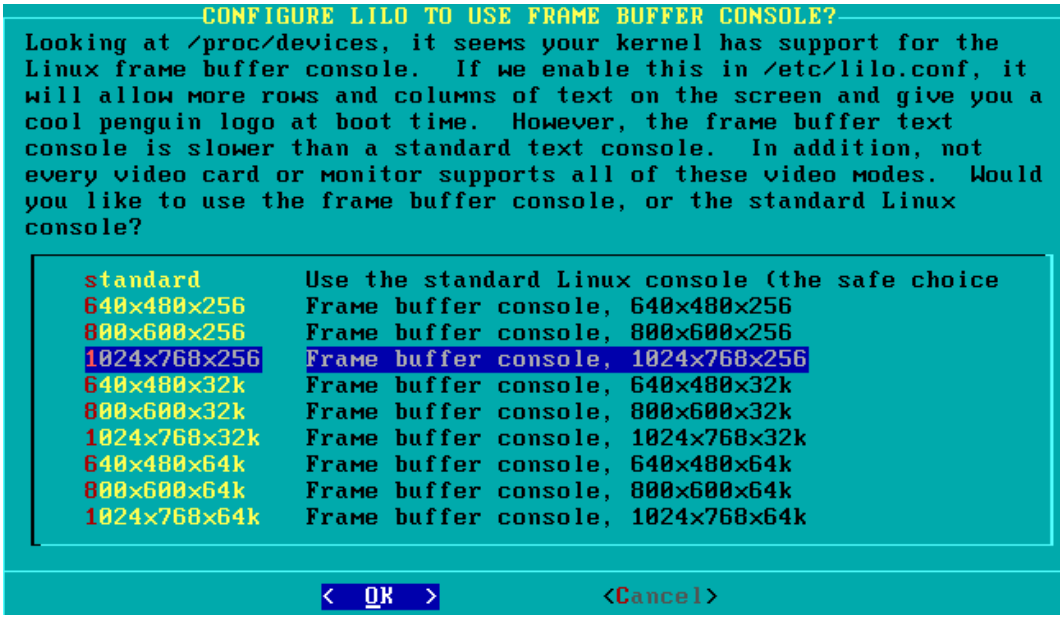


Figura 6 – Selección del Frame buffer

Después de configurar el buffer para el frame vendrá la configuración de parámetros que se desean pasar al Kernel a través de lilo. Por lo regular no es necesario especificar ningún parámetro así que se continuara con la instalación sin escribir nada para seguir con la instalación de lilo como se muestra en la figura 7.

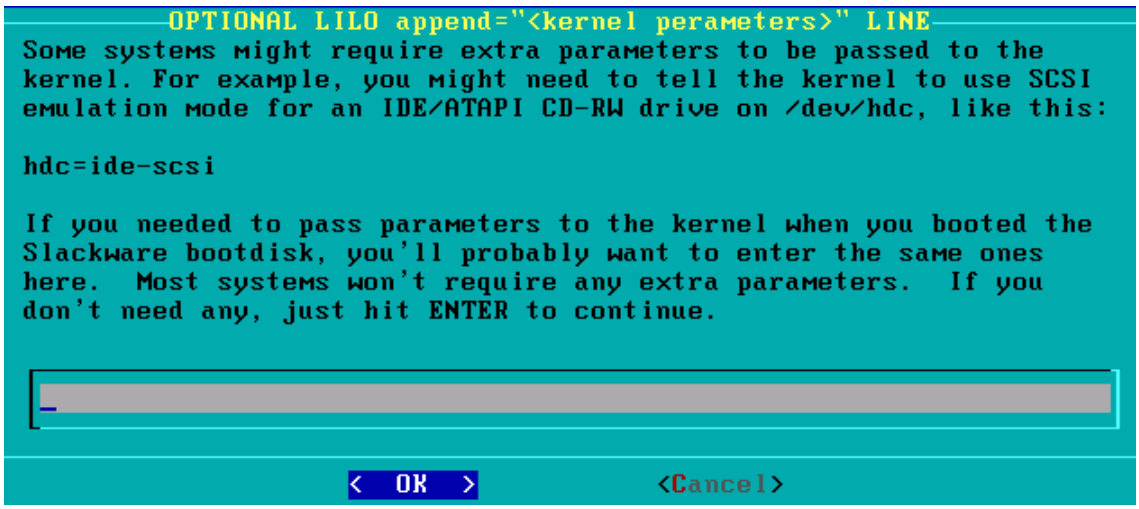


Figura 7 – Paso de parámetros al Kernel por medio de Lilo

El último paso de la configuración es seleccionar la forma en que lilo será instalado conforme a las opciones mostradas en la figura 8. El ***MBR*** es el registro de arranque principal de todas las computadoras, esta opción instalara lilo en el MBR, esta opción es empleada si se tiene a Slackware como único sistema operativo o si se quiere que lilo arranque a más sistemas operativos.

La opción ***Root*** instalara a lilo en el sector de arranque de la partición de Slackware (/), esta opción es utilizada si se tiene otro gestor de arranque diferente a lilo.

Figura 8 – Opciones de Instalación de lilo.

Para la configuración del mouse, Slackware provee una lista de controladores genéricos. Al no mostrar marcas comerciales o sólo las más importantes habrá que elegir el que más se apegue a las características del hardware que se están empleando.

La siguiente parte de la instalación hará la configuración de la red, estos pasos son opcionales. En este caso se elegirá la opción de configuración de la red.

En la pantalla de la figura 9 Slackware pedirá que se ingrese sólo el nombre del *host* o servidor, el dominio será agregado en el siguiente paso.

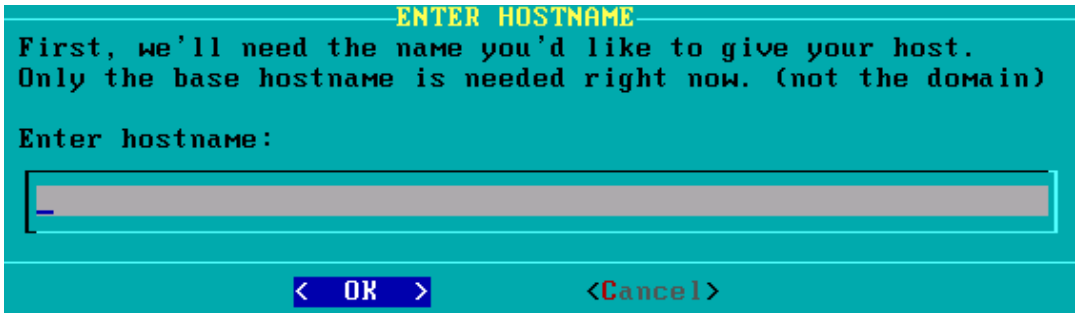


Figura 9 – Configuración del nombre del Host

Después de especificar el nombre del servidor se ingresa el dominio al que el equipo pertenecerá.

El resto de la configuración de la red dependerá de la manera en que se especificara la IP. Algunas redes asignan un IP dinámica por medio de un servidor DHCP. Si este es el caso, en este punto se elije la opción DHCP del menú mostrado en la figura 10. Los pasos subsecuentes a la configuración de la red con DHCP son opcionales, generalmente, los campos que se muestran en las ventanas se dejan en blanco.

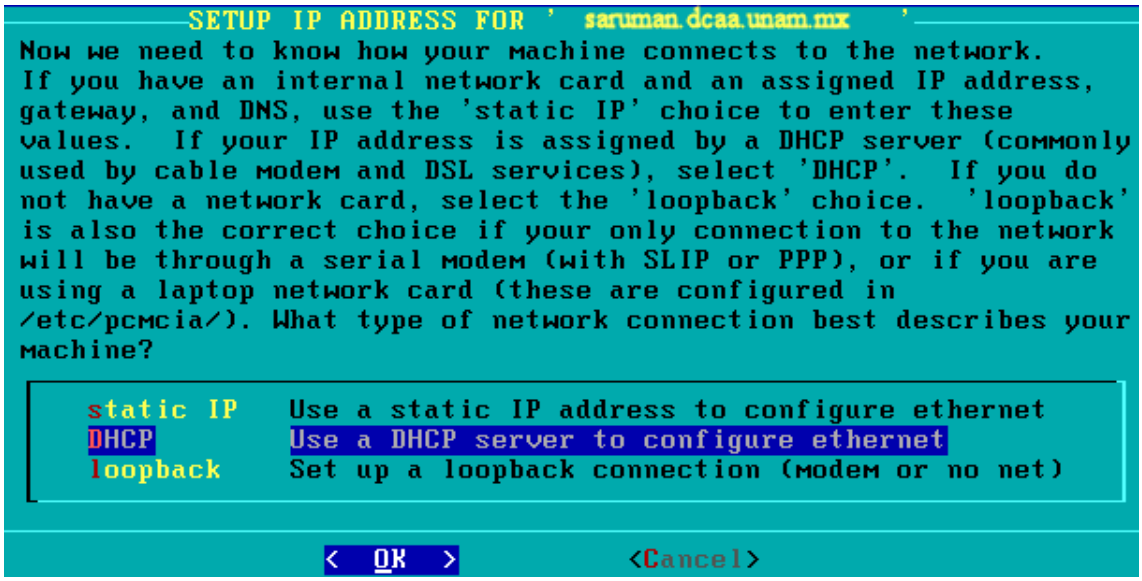


Figura 10 – Configuración de la dirección IP

Para especificar la dirección IP manualmente se elije la opción ***static IP***.

La primera parte de la configuración es la asignación de la dirección IP a la primer interfaz de red identificada como ***eth0*** como se muestra en la figura 11.

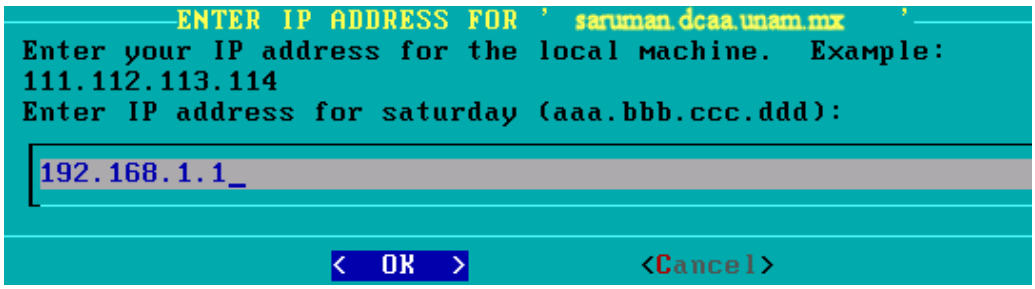


Figura 11 – Asignación manual de la dirección IP

A continuación se ingresa la mascara de red conforme a la dirección IP. Por lo general la mascara de red es *255.255.255.0.* En la siguiente pantalla corresponde a la dirección de la puerta de enlace (gateway). Si la red no posee una puerta de enlace, se puede oprimir la tecla *enter* para continuar sin especificar uno.

El siguiente dialogo preguntará si se desea configurar un servidor DNS o no. Se elije la opción de configurar y se especifica la dirección IP.

Finalmente, la herramienta de configuración de red mostrara un resumen como el mostrado en la figura 12 de los datos ingresados para la red dando la oportunidad de hacer correcciones en caso de ser necesario.

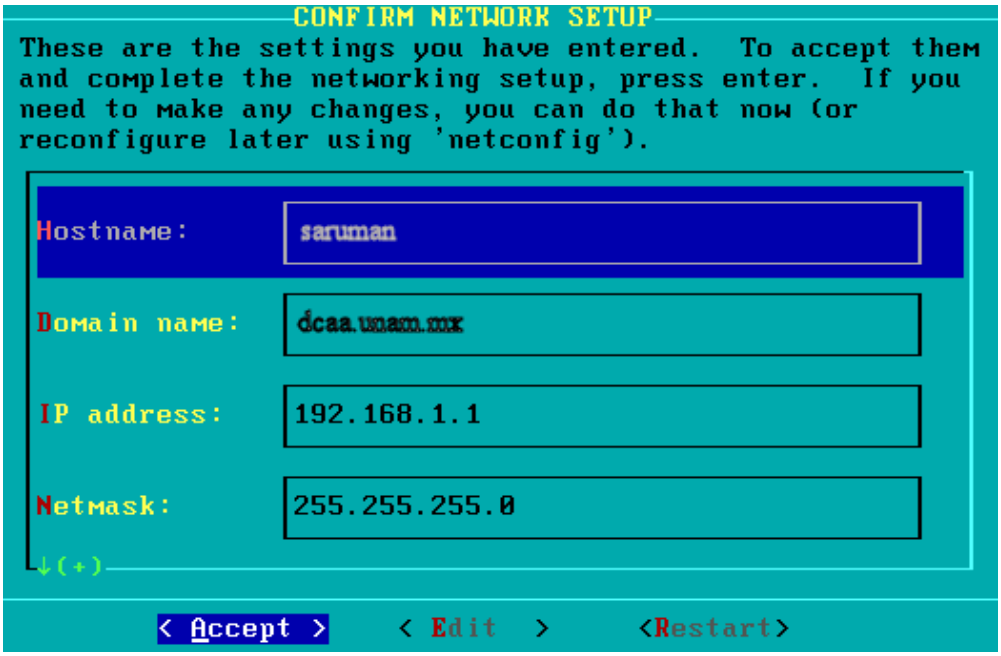


Figura 12 – Resumen de la configuración de red

Al término de la configuración de la red aparecerá una lista de los servicios que serán iniciados de forma automática cuando el sistema operativo arranque.

En el siguiente punto se escogerá la zona horaria que corresponda a México para ajustar el reloj del sistema.

Después, se seleccionara el administrador de ventanas por defecto que será usado para controlar y manejar el entorno gráfico del sistema. La tarea más básica del administrador de ventanas es la de proporcionar funcionalidad como la de las barras de título. Opciones como el manejador ***KDE*** ofrecen un entrono gráfico completo.

Finalmente, lo único que resta por hacer es crear una contraseña para el usuario *root*, de lo contrario, el sistema operativo quedara altamente desprotegido.

El mensaje mostrado en la figura 13 indicara que la instalación se realizo de forma exitosa, es necesario retirar el disco de instalación de Slackware y reiniciar el equipo.

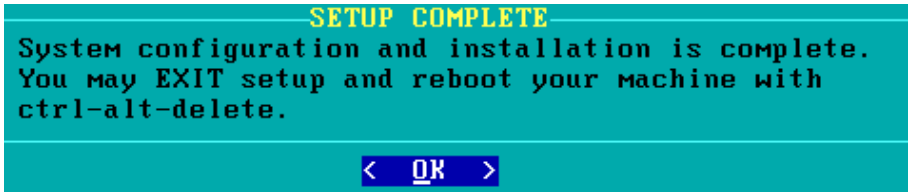


Figura 13 – Instalación terminada

2.2.6 Inicialización y montaje de dispositivos

Una de las configuraciones post-instalación que se recomienda es la de modificar el archivo ***fstab***. Este archivo dicta las especificaciones que el sistema operativo seguirá para el montaje de dispositivos como discos duros, unidades de CD-ROM, memorias USB, unidades de floppy, etc.

Fstab se encuentra dentro del directorio */etc*, y tiene la siguiente estructura:

# Dispositivo	Punto de montaje	FS	Opciones	Dump	Revisión
/dev/hda1	/	ext2	defaults	1	1
/dev/hda2	/home	ext2	defaults	1	2
/dev/hda3	/usr	ext2	defaults	1	2
/dev/hda4	swap	swap	defaults	0	0
/dev/fd0	/mnt/floppy	auto	noauto		
/dev/cdrom	/mnt/cdrom	iso9660	noauto		
# Partición de Windows					
/dev/hdb1	/mnt/windows	vfat	defaults	1	0

Donde:

- **Dispositivo** representa el bloque del dispositivo que se quiere montar.
- **Punto de montaje** representa el directorio donde se montara la unidad especificada por la columna que corresponde al dispositivo. Este directorio debe existir previamente.
- **FS** es el sistema de archivos del volumen que se desea montar. Este de puede especificar de forma explicita en el caso de particiones Linux o como auto para volúmenes con sistemas de archivos como FAT16, FAT32, dispositivos de CD-ROM o unidades de Floppy.

- **Opciones** especifica que parámetros deben ser usados al montar el sistema de archivos. Entre las opciones más comunes están:

  noauto — Los sistemas de archivos que están listados en el archivo fstab normalmente son montados de forma automática cuando el sistema operativo arranca, con la opción ***noauto*** el sistema de archivos sólo de montara después de ejecutar el comando ***mount***. Cuando se tiene configurado algún sistema de archivos de esta manera, sólo basta escribir el comando mount con la ruta del punto de montaje como única opción, tomando como ejemplo la descripción del archivo fstab mostrada anteriormente para montar la unidad de CD-ROM se escribe la instrucción
  mount /mnt/cdrom

  user — Permitirá a los usuarios normales montar este sistema de archivos. El montaje de sistemas de archivos normalmente sólo puede ser hecho por el usuario root.

  noexec — Con esta opción se impide a los usuarios ejecutar programas desde este sistema de archivos.

  defaults — Emplea la opciones por defecto para el montaje del sistema de archivo tales como que este volumen sólo puede ser instalado por root, se monta en modalidad lectura-escritura, permite la ejecución de programas desde él, entre otros.

- **Dump** recibe un valor de 1 o mayor a 1 que especifica después de cuantos días se debe realizar un respaldo del sistema de archivos. Esta opción sólo funciona cuando la utilería de respaldo *dump* está instalada y correctamente configurada para manejar esta configuración.
- **Revisión** determina si un sistema de archivos debe ser examinado por la utilería fsck para verificar su integridad cada que el sistema operativo inicia. Cuando se da el valor de 0 el sistema de archivos no es revisado. El valor de 1 comúnmente es asignado a la partición raíz y el valor de 2 se asigna a las particiones Linux para que estas sean verificadas en el arranque del sistema.

2.2.7 Administración del área Swap

Si las actividades del sistema operativo exigen el uso de una partición de intercambio más grande y no es posible volver a hacer la instalación completa del sistema, entonces se crea una nueva partición y se agrega a la cantidad total de almacenaje de la partición swap de forma dinámica.

2.2.7.1 mkswap

Para crear una nueva área de swap se emplea el comando ***mkswap***. La sintaxis del comando mkswap es la siguiente:

mkswap [opciones] dispositivo [tamaño]

Entre las principales opciones de *mkswap* están:

-c — Examina el dispositivo en busca de bloques dañados antes de crear la nueva área de swap.

-f — Forza la creación de la nueva área de swap.

2.2.7.2 swapon

Una vez que se tiene el área que será destinada para la partición swap, se le notifica al sistema operativo para que haga uso de ella. El comando ***swapon*** permite hacer esto. La sintaxis de swapon es la siguiente:

swapon dispositivo

2.2.7.3 swapoff

Si se necesitara detener el uso del área de swap se emplea el comando ***swapoff***. La sintaxis de swapoff es la siguiente:

swapoff dispositivo

Por ejemplo, si se agregara el dispositivo /dev/hda2 como área swap:

```
# mkswap /dev/hda2
# swapon /dev/hda2
```

Comúnmente se agregan las líneas correspondientes para el montaje de la partición swap al archivo *fstab* para que este sea montado de forma automática al iniciar el sistema.

2.2.8 Monitoreo del Sistema

Al administrar el sistema hay que estar al pendiente de los recursos que algunas de las aplicaciones están consumiendo o que procesos se esta corriendo y por quien fueron ejecutados. Linux cuenta con sencillas herramientas para realizar estas tareas.

2.2.8.1 top

El programa *top* provee una vista dinámica en tiempo real de un sistema que se encuentra en ejecución. Puede mostrar el resumen de la información del sistema junto con la lista de tareas que el kernel está manejando. Los tipos de resúmenes de información del sistema, orden de la presentación de la información de las tareas y el consumo de recursos ocupados son vistas que pueden ser configurables por el usuario.

Los campos manejados por top son:

PID	Identificador único de proceso.
PPID	Identificador único del proceso padre.
USER	Nombre del usuario propietario del proceso.
SIZE	Tamaño virtual del proceso.
STAT	Estado del proceso.
%CPU	Porcentaje de uso del CPU para esta tarea desde la última actualización de la pantalla.
%MEM	Porcentaje de uso de la memoria física del sistema para esta tarea.
TIME	Tiempo total de uso del CPU que la tarea ha usado desde que empezó.
COMMAND	Nombre del comando o programa en ejecución-

2.2.8.2 ps

El comando *ps* da una "imagen estática" de los procesos que actualmente se encuentran en ejecución, si se necesitará un monitoreo repetitivo del estatus de un proceso se recurriría al programa top.

La sintaxis del comando ps es la siguiente:

ps [opciones]

Entre las principales opciones de *ps* están:

-f	Hace un listado detallado de los procesos
-e	Selecciona todos los procesos.
-a	

- UID. Identificador del dueño del proceso.
- PID. Identificador del proceso en ejecución.
- PPID. Identificador del proceso padre del proceso en ejecución.

ps también muestra el identificador del proceso de acuerdo con lo siguiente:

z	Zombie, cuando un proceso corre en forma independiente, incluso del shell o del proceso que lo invocó.
r	En ejecución, se encuentra en la CPU.
t	Detenido
s	Sleeping Durmiendo en espera de recibir una solicitud para iniciar su trabajo.

2.2.9 Respaldos

2.2.9.1 tar

El comando ***tar*** permite "empacar" una estructura de directorios, su función principal es almacenar el contenido de un directorio en forma recursiva dentro de un archivo.

La sintaxis del comando *tar* es:

tar [opciones] nombre_del_respaldo.tar archivos_a_empacar

Entre las principales opciones de *tar* están:

c	Crea un nuevo archivo tar
v	Muestra todos los archivos que están siendo afectados
f	Indica al sistema que trabajara sobre un archivo
x	Extrae el contenido de un archivo tar
z	Permite comprimir el paquete.

Por ejemplo, para respaldar el archivo *tareas.txt* en un paquete llamado *respaldo_tareas.tar* el comando quedaría de la siguiente forma:

tar cvf respaldo_tareas.tar tareas.txt

Ahora, para extraer el contenido del paquete generado:

tar xvf respaldo_tareas.tar

En este caso el uso de los guiones en las opciones de tar son optativas.

Como regla general cualquier opción es combinada con las opciones ***vf*** ya que es recomendable ver qué esta realizando el sistema, así como para indicarle que se está enviando la salida del comando a un archivo, de lo contrario tomará por defecto la unidad de cinta.

2.2.9.2 gzip

Gzip reduce el tamaño ocupado por los archivos especificados. Gzip sustituye el archivo original por un archivo con extensión ***.gz*** manteniendo las mismas referencias de su propietario, fechas accesos y de modificaciones.

Gzip también puede descomprimir archivos creados con zip ya que ambos emplean el algoritmo Lempel-Ziv.

La sintaxis del comando *gzip* es:

gzip [opciones] nombre_del_archivo

Entre las principales opciones de *gzip* están:

-d	Descomprime un archivo.
-f	Forza la compresión o descompresión de un archivo.
-l	Lista los detalles de los archivos comprimidos.
-r	Realiza la compresión de forma recursiva, esta opción es empleada
-v	Muestra el nombre y la tasa de compresión de cada archivo.
-S	Permite asignar cualquier extensión al nombre del archivo

2.2.9.3 dump

Examina archivos o sistemas de archivos y determina cual debe ser respaldado. Estos archivos son copiados al medio de almacenamiento especificado.

La sintaxis del comando *dump* es:

dump [opciones] sistema_de_archivos

Entre las principales opciones de *dump* están:

-0-9	Niveles de respaldo. El nivel 0 especifica que se debe hacer un respaldo completo lo que garantiza que el sistema de archivos entero sea copiado. Los niveles posteriores indican que los respaldos serán incrementales, es decir, que sólo copiara los archivos nuevos o modificados desde el último respaldo.
-f	Especifica la ruta donde el respaldo será escrito.
-W	Especifica los archivos o sistemas de archivos que deben ser respaldados.

2.2.10 Interfaces Gráficas en Linux

Dentro de Slackware el archivo encargado de manejar y brindar un entorno gráfico a través del sistema ***X Window*** es ***xorg.conf*** que se encuentra ***/etc/X11***.

A su vez, el sistema X Window es implementado a través de muchos programas que se encuentran en el sistema operativo desde la instalación. Los dos componentes fundamentales de X Window son el *servidor* y el *administrador de ventanas.* El servidor provee las funciones de bajo nivel para la interacción con el hardware de video. El administrador de ventanas se encuentra en un nivel superior al servidor y provee la interfaz de usuario. La ventaja de esto es que se puede cambiar entre diferentes interfaces de usuario con sólo cambiar el administrador de ventanas.

2.2.10.1 Configuración del X Server

La configuración de este servicio puede ser una tarea complicada dado que existen múltiples tipos de tarjetas de video para la arquitectura de una PC y la mayoría de ellas utiliza diferentes interfaces de programación. Afortunadamente, la mayoría de las tarjetas actuales soportan el estándar de video VESA. Si la tarjeta de video que se encuentra instalada maneja este estándar entonces es posible ejecutar el comando ***startx*** para iniciar el entorno gráfico.

Si la tarjeta de video no llegara a funcionar como debe o se pretende aprovechar cualidades especificas de la tarjeta como aceleración 3D habrá que hacer las modificaciones correspondientes al archivo ***xorg.conf*** que se encuentra en el directorio */etc*.

2.2.10.2 Herramientas de configuración

Xorg.conf contiene un gran número de detalles acerca del hardware de video, mouse y monitor. Es posible configurarlo con la ayuda de programas especializados en su manejo como:

xorgsetup

Este es un menú muy sencillo. Indica al X Server que haga una rápida inspección a la tarjeta de video y genere un archivo de configuración inicial basado en la información que obtenga. El archivo xorg.conf que obtenga es un buen punto de inicio para la mayoría de las arquitecturas de PC's y deberá trabajar sin modificaciones.

xorgconfig

Xorgconfig es un programa basado en texto que permite realizar una configuración más avanzada. Permite elegir el tipo de mouse instalado e incluso emular el tercer botón con el que algunos modelos cuentan, asignación de teclas especiales, configuración de los rangos de frecuencia de barrido del monitor, elección del chipset de la tarjeta gráfica (o de la misma familia), memoria RAM de la tarjeta y resoluciones de pantalla entre otras opciones.

Lo que resta por hacer para completar la configuración del entorno gráfico es elegir al administrador de ventanas predeterminado. Para ver la lista de administradores de ventanas que incluye Slackware se ejecuta el programa ***xwmconfig***.

2.2.10.3 KDE (K Desktop Environment)

KDE es un entorno de escritorio gráfico e infraestructura de desarrollo para sistemas Unix y en particular para Linux. KDE llena la necesidad de un escritorio amigable para estaciones de trabajo Linux, similar a los escritorios de los sistemas operativos MacOS y Windows.

KDE está escrito casi exclusivamente en C++, un lenguaje derivado del lenguaje de programación C con algunas funcionalidades añadidas, en especial en cuanto a la programación orientada a objetos.

KDE cuenta además con su propio sistema de entrada/salida llamado KIO, el cual puede acceder a un archivo local, un recurso de red a través de protocolos como HTTP, FTP, NFS, SMB, etc. con absoluta transparencia. Esta arquitectura permite a los desarrolladores agregar nuevos protocolos sin requerir modificaciones en la base del sistema.

Adicionalmente posee su propio motor HTML llamado KHTML, el cual está siendo reutilizado y ampliado por Apple para la creación de su navegador Safari.

2.2.10.4 GNOME (GNU Network Object Model Environment)

GNOME es un entorno de escritorio para sistemas operativos de tipo Unix y Linux bajo la tecnología X Window, se encuentra disponible actualmente en más de 35 idiomas. Forma parte oficial del proyecto GNU.

Tiene la intención de construir un completo y amigable escritorio basado enteramente en software libre. Permite usar varios espacios de trabajo, cada uno como un escritorio independiente de los demás. Básicamente GONE ofrece dos cosas: un entorno gráfico para usuarios finales y una plataforma de desarrollo.

A diferencia de KDE, GNOME permite a los desarrolladores emplear lenguajes de programación como C, C++, Python, Perl, Java, incluso C#, para la creación de aplicaciones que son fácilmente integrables al resto del proyecto.

2.3 Introducción al Diseño de páginas Web con HTML

HTML viene del acrónimo en inglés de **H**yper**t**ext **M**arkup **L**anguage (lenguaje de formato de documentos de hipertexto), es un lenguaje de etiquetas diseñado para estructurar textos y presentarlos en forma de hipertexto, que es el formato estándar de las páginas Web.

Un hipertexto es un documento digital o no, que se puede leer de manera no secuencial. Un hipertexto tiene los siguientes elementos: secciones, enlaces o hipervínculos y anclajes. Las secciones o nodos son los componentes del hipertexto. Los enlaces son las uniones entre nodos que facilitan la lectura secuencial o no secuencial del documento. Los anclajes son los puntos de activación de los enlaces.

Quizá las dos características más importantes de un documento HTML es que es sencillo de crear y que este es casi independiente de la plataforma y el navegador que lo interpreta.

2.3.1 DTD (Documente Type Definition)

Los DTD's son generalmente empleados para determinar la estructura de un documento de hipertexto. Un DTD describirá un conjunto de etiquetas que pueden ser empleadas dentro del documento.

2.3.2 Componentes de un documento HTML

Como se menciono anteriormente, HTML consiste en un grupo de etiquetas predefinidas que son empleadas a lo largo de todo el documento con el texto. Estas etiquetas le dicen al navegador si el texto es un párrafo, un encabezado, una imagen, etc. Al mismo tiempo, las etiquetas permiten que la información tenga un formato en específico.

2.3.2.1 Etiquetas

Hay dos partes básicas en cualquier página Web: el ***encabezado*** y el ***cuerpo***. Ambas partes emplean etiquetas para realizar sus respectivas tareas. Generalmente las etiquetas son escritas en minúsculas, pero los navegadores no hacen diferencia entre minúsculas y mayúsculas.

El encabezado contendrá todas las etiquetas que especificarán características de la página propiamente como: título, palabras de búsqueda empleadas por un buscador, última actualización, autor de la página, etc, estos datos serán invisibles en el contenido que el usuario vera. El cuerpo de la página tendrá todo el contenido que el usuario vera: imágenes, texto, enlaces, etc.

Hay dos tipos de etiquetas: las ***etiquetas de inicio*** y las ***etiquetas de cierre***. Ambas son escritas entre paréntesis angulares, por ejemplo <h1>. Las etiquetas de cierre son idénticas a las de inicio, excepto que estas están precedidas por una diagonal, por ejemplo </h1>. Estas etiquetas indican cuando empieza un efecto especial y cuando termina.

El siguiente código mostrara una página Web muy sencilla donde se verán las dos secciones básicas, el encabezado y el cuerpo de la página.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
<title>P&aacute;gina de Prueba</title>
<meta name="Author" content="Gabriel Gonzalez garcia">
<meta name="Description" content="Primera pagina">
</head>

<body>
        <h1>Cuerpo de la p&aacute;gina</h1>
        La inforaci&oacute;n va aqu&iacute;.
</body>
</html>
```

El código anterior resulta en la página mostrada en la figura 14:

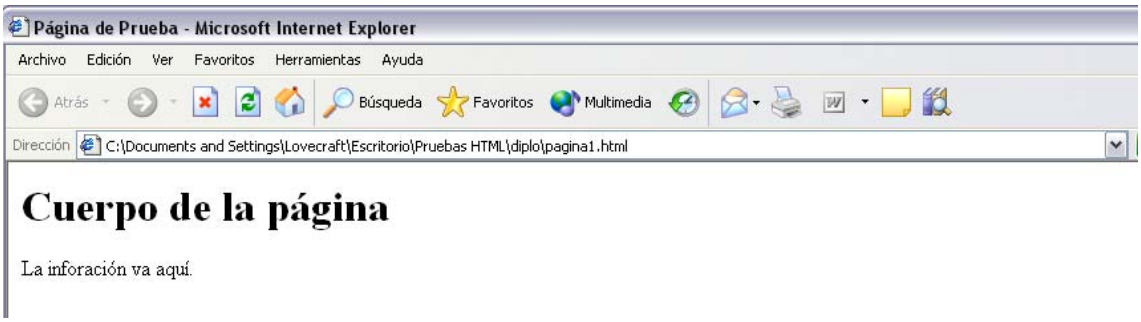


Figura 14 – Página generada

La mayoría de los efectos tienen una etiqueta de inicio y una de cierre pero no es el caso para todas.

Casi todas las etiquetas pueden ser anidadas, es decir, tener etiquetas dentro de otras para obtener efectos más complejos. El orden en que las etiquetas son escritas es importante. Cuando se anidan etiquetas se debe cerrar primero la etiqueta que se encuentra “más adentro”:

esta palabra está en letras cursivas

2.3.2.2 Atributos

Los atributos permiten extender las capacidades de las etiquetas. Los atributos son empleados en algunas etiquetas para controlar tipos de fuente, bordes, colores, alineaciones, etc.

Por ejemplo, la etiqueta <h1> que imprime un título en una línea separada como se mostró en la figura 15, tiene un atributo de alineación. Si se escribiera la etiqueta <h1> con su atributo ***align*** de la siguiente manera:

<h1 align=’center’>Título Centrado</h1>

Se obtendría la imagen mostrada en la figura 15:

Figura 15 – Atributo *align* de la etiqueta <h1>

2.3.3 Etiquetas básicas de HTML

<html> Señala el inicio y el fin de un documento HTML y encapsula todo aquello que debe ser interpretado como tal.

<head> Señala el encabezado del documento el cual es usado para contener información genérica del documento tal como título, estilo del documento, etc. Dentro de la etiqueta <head> es muy común encontrar las siguientes etiquetas:

<title>	Despliega la cadena contenida entre la etiqueta <title> de inicio y fin en la barra de título del navegador.
<link>	Define las relaciones del documento Web con otros documentos. El atributo más común de <link> es ***href*** que designa la dirección del recurso ligado al documento.
<style>	Define las características del estilo que se aplicará al documento Web.
<script>	Define las funciones implementadas a través de leguajes de programación que pueden incrustarse dentro del documento HTML como es el caso de Java Script para brindar contenidos dinámicos.
<meta>	Es usada para agregar información adicional sobre el documento HTML. Esta información es extraída e interpretada por los navegadores para identificar y realizar funciones específicas sobre el documento. Entre los atributos más comunes de <meta> están: • ***name***: Especifica el nombre de la propiedades de la página como autor, fecha de generación, compañía, etc. • ***content***: Establece un valor para el nombre de la propiedad declarado por ***name***.

<body> Esta etiqueta marca el inicio del cuerpo del documento HTML Existen varios atributos que pueden ser usados con <body>

- ***background***: Especifica que una imagen será usada como fondo de la página.
- ***bgcolor***: Establece un color en hexadecimal de fondo para la página.
- ***alink***: Establece un color en hexadecimal para la liga de hipertexto activa. La liga está activa
- ***link***: Establece un color en hexadecimal para la liga de hipertexto que no ha sido visitada.
- ***vlink***: Establece un color en hexadecimal para las ligas de hipertexto visitadas
- ***text***: Establece un color en hexadecimal para el texto de la página

Dentro de la etiqueta <body> es muy común encontrar las siguientes etiquetas:

<hn>	Esta etiqueta permite especificar títulos, donde la *n* es un número que representa el tamaño de la letra, este va de 1 a 6, donde el 1 es el mayor
<p>	Esta etiqueta permite separar los textos en párrafos con sus respectivas alineaciones: centrado, izquierda, derecha, justificado. El atributo de <p> es ***align***.
y	Estas etiquetas permiten definir listas. La etiqueta

 especificar que el tipo de lista no estará ordenada, es decir, que los elementos contenidos en dicha lista no tienen que tener una numeración o algo que denote la continuidad de los elementos. Por el contrario, la etiqueta <ol> denota que los elementos contenidos estarán enumerados o tendrán alguna relación de continuidad.

Para comenzar a agregar los elementos a las lista, independientemente del tipo se utilizara la etiqueta <li>.

<b> Texto en negritas

<i> Texto en cursivas

<u> Texto subrayado

<font> Esta etiqueta permite modificar atributos de la fuente que se utilizara dentro de la página HTML. Los atributos de <font> son:

- ***color***: Color en hexadecimal de la fuente.
- ***face***: Tipo de fuente.
- ***size***: Tamaño de la fuente

<a> Esta etiqueta permite crear ligas o hipervínculos hacia otros documentos HTML, archivos o direcciones remotas. Lo atributos más comunes de <a> son:

- ***href***: Especifica el destino de la liga.
- ***target***: Señala un parte especifica del documento HTML para mostrar el contenido de la liga, este argumento es muy útil cuando se trabaja con frames. ***Target*** puede tener cuatro valores:
  1. **_blank.** Carga el contenido de una liga en una nueva ventana.
  2. _**self**. Carga el contenido de la liga en la misma página o frame.
  3. _**parent**. Es usado cuando se tienen frames anidados. Si el frame actual es hijo entonces se actualizara el frame padre.
  4. _**top**. Actualizara todo el documento HTML.

<img> Esta etiqueta permite insertar una imagen en el cuerpo de la página HTML, la imagen puede estar físicamente en nuestro disco duro o en alguna dirección remota. Cuenta con sus atributos *src* que especifica la ruta de la imagen, *width* y *height* que especifican el tamaño.

2.3.4 Creación de Tablas

Con la aparición las tablas se revolucionó el diseño de las páginas Web. Las tablas son una herramienta perfecta para organizar datos de manera más legible, pero su

utilidad no se queda ahí, ya que escondiendo los bordes pueden ser usadas para definir la estructura de las páginas.

Para crear tablas será necesario usar la etiqueta ***<table>***. Esta etiqueta consta de instrucción de inicio, <table> y la instrucción de fin, </table>. Entre ambas se escriben otras etiquetas, que definirán la estructura de la tabla. Estas etiquetas son:

- ***<tr>***: La etiqueta *Table Row* permitirá insertar filas en la tabla. La tabla tendrá tantas filas como apariciones de esta etiqueta haya entre <table> y </table>. La instrucción de inicio es <tr> que marca el comienzo de la línea. La instrucción de fin es optativa, si no se usa se considera que una línea ha acabado cuando comienza otra o cuando acaba la tabla. Entre el comienzo y el fin de la línea hay insertar las celdas de la tabla.
- ***<td>***: La etiqueta *Table Data* permite introducir todos los datos que estarán en las celdas definidas de esta forma. Cualquier elemento de HTML puede ser empleado.

Los principales atributos de <table> son:

- **align**: Alinea horizontalmente la tabla con respecto a su entorno.
- **background**: Permite colocar un fondo para la tabla a partir de un enlace a una imagen.
- **bgcolor**: Establece un color en hexadecimal de fondo para la tabla.
- **border:** Define el número de píxeles del borde principal.
- **bordercolor**: Establece un color en hexadecimal de fondo para el borde de la tabla.
- **height**: Define la altura de la tabla en píxeles o porcentaje.
- **width**: Define la altura de la tabla en píxeles o porcentaje.

2.3.5 Formularios

Una de las mayores ventajas de los sitios Web sobre lo medios tradicionales de difusión de la información es la retroalimentación casi instantánea por parte de los usuarios.

Los formularios ofrecen una forma de recopilar información de los usuarios y procesar dicha información en base a la lógica de dicho formulario. Por ejemplo, las formas son utilizadas muy frecuentemente para registrar a un usuario antes de que puede abrir una cuenta de correo o iniciar la descarga de algún programa. Desde el punto de vista de la programación, todos los documentos HTML se dividen en dos partes: la parte del lado del cliente y la parte del lado del servidor. La parte de lado del cliente es aquella que acepta la entrada de información por parte de los usuarios y la envía al servidor, es decir, la misma forma HTML Una vez que la información es enviada el navegador simplemente asume que el servidor sabe que hacer con ella.

Para crear formularios será necesario usar la etiqueta ***<form>***. Esta etiqueta consta de instrucción de inicio, <form> y la instrucción de fin </form>. Entre ambas se escriben otras etiquetas que definirán la estructura del formulario. Estas etiquetas son:

- ***<input>*** Define el tipo de elementos que el formulario tendrá para obtener información del usuario. Los principales son:

text	Campo de texto editable
password	Similar al tipo text, sólo que esta propiedad no mostrara los

	caracteres escritos, en vez de eso los mostrara como asteriscos.
radio	Botones de radio
button	Botón clásico
submit	Botón que enviara los valores de los elementos input al servdor
hidden	Campo de información oculto. Se emplea para enviar información al servidor que no necesita ser vista o manipulada por el usuario.

- ***<select>*** Define un control de tipo caja de combo donde el usuario podrá escoger sólo una de las opciones mostradas entre la etiqueta de inicio y fin. Dentro de <select> se define la etiqueta ***<option>*** que representa las opciones que la caja de combo mostrara.

Estas etiquetas a su vez tienen los siguientes atributos:

- ***name***: Nombre del control input que contendrá el dato ingresado por el usuario. Este atributo es muy importante ya que por medio de este nombre el servidor hará referencia al valor de este control.
- ***value***: Valor del control input. El valor de este atributo se asigna después de que el usuario hace clic en el botón enviar.

Los principales atributos de <form> son:

- ***action***: Este atributo es usado para especificar la URL que recibirá los datos pasados por la forma HTML. No es necesario que esta URL este en la misma máquina donde se encuentra el formulario.
- ***method***: Este atributo define el método empleado para enviar la información al servidor. Estos métodos son ***POST*** y ***GET***. POST es el método más recomendado para el envío de información, POST envía la información como un flujo de datos, de este modo, es posible enviar una gran longitud de información. GET es un método más antiguo. Con este método la información es enviada al servidor a través de la URL lo cual puede significar un riesgo ya que toda la información sensible incluyendo contraseñas puede ser vista sin ningún problema como se muestra en la figura 16.

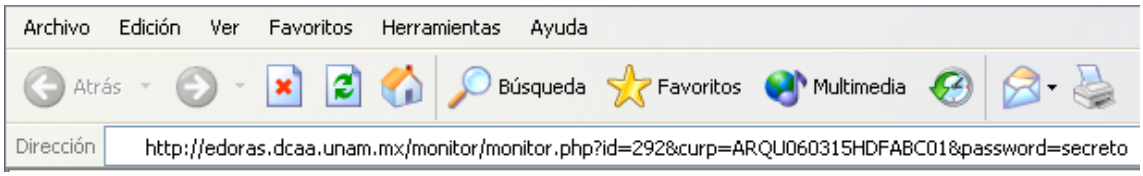


Figura 16 – Paso de valores a través del método GET

En la figura anterior, la forma HTML envía los valores de los campos, estos valores son referenciados por el nombre que se le dio con el atributo *name* de cada *input*

El siguiente código creara una forma HTML con los elementos básicos, una caja de texto, un campo de contraseña y un botón para enviar la información al servidor.

```
<html>
<head>
<title>Formulario B&aacute;sico</title>

</head>

<body>
        <form action='procesar.php' method='POST'>
                Usuario <input type='text' name='usuario'><br>
                Password <input type='password' name='password'><br>
                <input type='submit'>
        </form>
</body>
</html>
```

En el código anterior, el formulario enviara la información a un script de PHP llamado *procesar.php* para su procesamiento y el método que utilizara para hacerlo será POST.

El procesamiento de la información enviada por un documento HTML se mostrara en el modulo correspondiente a *Programación con PHP*.

El formulario estará compuesto de un campo donde se escribirá un nombre de usuario y un campo especial donde se escribirá la contraseña de acceso y finalmente un botón para enviar la información contenida en la forma HTML al servidor.

El código anterior resulta en el formulario mostrado en la figura 17

Figura 17 – Formulario HTML

2.3.6 Frames

Una de las características más modernas de HTML son los frames que fueron añadidos a los navegadores más populares en sus últimas versiones. Los frames o marcos son una manera de dividir la página en distintos espacios independientes unos de otros, de modo que en cada espacio se coloca una página distinta; cada página se codifica en un archivo .html diferente.

Cada frame o marco contiene las propiedades específicas indicadas en el código HTML a presentar en ese espacio. Así mismo, y dado que cada marco es

independiente, tendrán sus propias barras de desplazamiento, horizontales y verticales.

Al principio se crearon como etiquetas propietarias del navegador Netscape y rápidamente la potencia del recurso hizo que el uso de los frames se extendiera por casi todos los sitios Web. Poco tardaría Internet Explorer en incluirlos como parte de funcionalidad. Finalmente, como respuesta a la popularidad entre los desarrolladores de los frames, el estándar HTML 4.0 incluyó estas etiquetas dentro de las permitidas.

Las etiquetas utilizadas para crear una página HTML con frames son:

<frameset ***x***> Esta etiqueta le indica al navegador como se va a dividir la pantalla. X representa el atributo ***cols*** o ***rows***; estos atributos especifican si las divisiones se harán vertical u horizontalmente.

<frame> Por regla general se emplea el atributo ***src*** de <frame> para indicar el nombre de la página HTML que se colocara en ese frame.

<noframes> Esta etiqueta mostrara un mensaje al usuario en caso de que la versión de su navegador no sea capaz de manejar frames

Para crear una página que tuviera dos frames: uno para un menú y otro para mostrar el contenido, habría que escribir cuatro códigos, uno para cada una de las páginas, es decir, un código para la página de menú, principal, la siguiente página a mostrar y la página que contendrá a los frames.

menu.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD html 4.0 Transitional//EN">
<html>
<head>
<title>Men&uacute;</title>

</head>

<body>
        <a href='principal.html' target='principal'>P&aacute;ginaPrincipal</a><br> <br>
        <a href='otrapagina.html' target='principal'>Ver la siguiente p&aacute;gina</a>
</body>
</html>
```

principal.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD html 4.0 Transitional//EN">
<html>
<head>
<title>P&aacute;gina Principal</title>

</head>

<body>
        <br><br><br>
        <h2 align='center'>Bienvenido, esta es la p&aacute;gina principal</h2>
</body>
</html>
```

otrapagina.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD html 4.0 Transitional//EN">
<html>
<head>
<title>Otra P&aacute;gina</title>

</head>

<body>
        Este es el contenido de la siguiente p&aacute;gina.
</body>
</html>
```

frames.html (pagina que creara los frames)

```
<html>
<head>
        <title>Página de frames</title>
</head>

<frameset cols="30%,70%">
         <frame src="menu.html">
         <frame src="principal.html" name="principal">
</frameset>

<noframes>
         Su navegador no soporta frames.
</noframes>
</html>
```

El código de frames.html sería el código HTML que define la página que contiene los frames. De esta forma, se indica al navegador que hay que dividir la página en dos frames (en columnas), uno que ocupara el 30% de la parte izquierda, y otro que ocupara el 70% de la parte derecha de la página.

Si se observa el código, se apreciara que no contiene la etiqueta <*body*>. Esto es debido a que el contenido de la página son únicamente los frames. En lugar de la etiqueta <*body*> se utiliza la etiqueta <*frameset*> que contiene la definición de los frames.

Como se menciono anteriormente se utiliza la etiqueta <*noframes*> para mostrar el contenido en navegadores que no soporten frames.

Las etiquetas <*frame src=…*> indican el contenido de los frames. En el caso del frame izquierdo, contendrá la página *menu.html* y en el caso del frame derecho esta tendrá la página *principal.html*. Cabe destacar que en la definición del frame derecho se incluyo el parámetro *name="principal"*, el cual permite asignar un nombre a ese frame. De esta manera, cuando se hace clic en cualquiera de las ligas del frame menú su contenido será mostrado en principal

La unión de los códigos anteriores generaría la página mostrada en la figura 18.

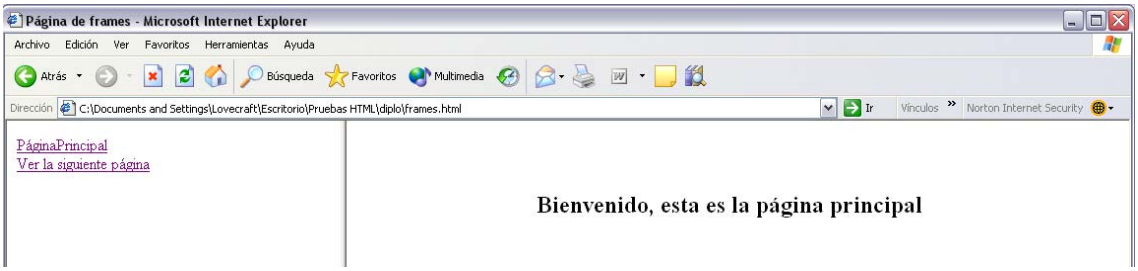


Figura 18 – Página inicial con frames

2.4 Administración de Servidores WWW con Linux

2.4.1 Introducción a los servidores WWW

Un servidor WWW o servidor Web es un programa que implementa el *protocolo HTTP*. Este protocolo está diseñado para transferir los hipertextos, páginas Web o páginas HTML: textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de sonidos.

Sin embargo, el hecho de que HTTP y HTML estén íntimamente ligados no debe dar lugar a confundir ambos términos. HTML es un formato de archivo y HTTP es un protocolo.

Cabe destacar el hecho de que la palabra *servidor* identifica tanto al programa como a la máquina en la que dicho programa se ejecuta. Existe, por tanto, cierta ambigüedad en el término.

Un servidor Web se encarga de mantenerse a la espera de *peticiones HTTP* llevada a cabo por un *cliente HTTP* que es el *navegador*. El navegador realiza una petición al servidor y éste le responde con el contenido que el cliente solicita. El servidor responde al cliente enviando el código HTML de la página; el cliente, una vez recibido el código, lo interpreta y lo muestra en pantalla. El cliente es el encargado de interpretar el código HTML, es decir, mostrar las fuentes, los colores y la disposición de los textos y objetos de la página; el servidor tan sólo se limita a transferir el código de la página sin llevar a cabo ninguna interpretación de la misma, después de ésta transferencia, el servidor cierra la conexión.

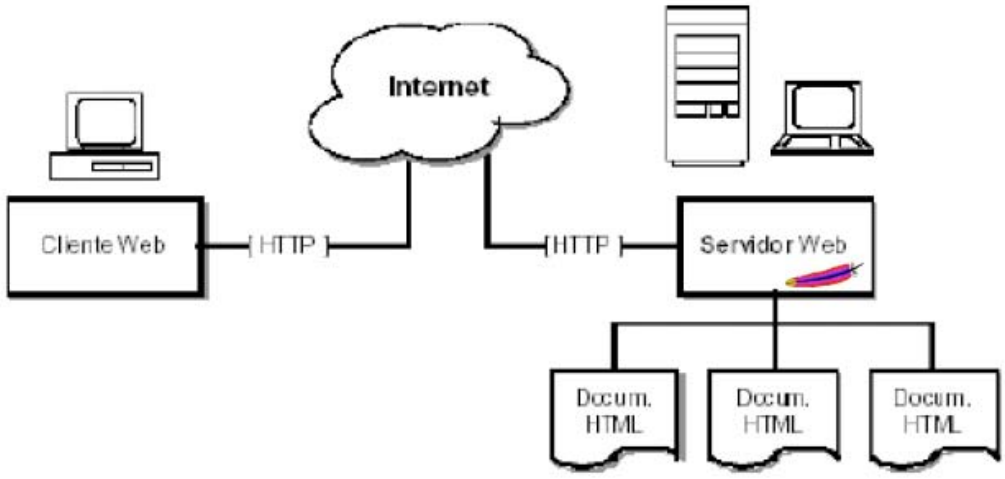


Figura 19 – Esquema del Servidor WWW

Sobre el servicio Web *clásico* se puede disponer de aplicaciones Web. Éstas son fragmentos de código que se ejecutan cuando se realizan ciertas peticiones o respuestas HTTP. Hay que distinguir entre:

- Aplicaciones en el lado del cliente: el cliente Web es el encargado de ejecutarlas en la máquina del usuario. Son las aplicaciones tipo Java o JavaScript el servidor proporciona el código de las aplicaciones al cliente y éste, mediante el navegador, las ejecuta. Es necesario, por tanto, que el cliente disponga de un navegador con capacidad para ejecutar aplicaciones, también llamadas *scripts*. Normalmente, los navegadores permiten ejecutar aplicaciones escritas en lenguaje *javascript y java*, aunque pueden añadirse mas lenguajes mediante el uso de *plug-in's.*
- Aplicaciones en el lado del servidor: el servidor Web ejecuta la aplicación; ésta, una vez ejecutada, genera cierto código HTML; el servidor toma este código recién creado y lo envía al cliente por medio del protocolo HTTP.

Algunos servidores Web son:

- Apache
- IIS
- Sun Java Web Server
- Zeus Web Server

2.4.2 Instalación del Servidor WWW

2.4.2.1 Servidor HTTP Apache

El servidor HTTP Apache es un servidor HTTP de código abierto para plataformas Unix/Linux y Windows y otras, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual. Cuando comenzó su desarrollo en 1995 se basó inicialmente en código del popular NCSA HTTPd, un Servidor web desarrollado originalmente en el National Center for Supercomputing Applications y que suspendió su desarrollo en 1998.

El servidor Apache se desarrolla dentro del proyecto HTTP Server (httpd) de la Apache Software Foundation, presenta entre otras características mensajes de error altamente configurables, bases de datos de autenticación y negociado de contenido, pero fue criticado por la falta de una interfaz gráfica que ayude en su configuración. En la actualidad, Apache es el servidor HTTP más usado, siendo el servidor HTTP del 70% de los sitios Web en el mundo.

2.4.2.2 Módulos

La arquitectura del servidor Apache es muy modular. El servidor consta de un sección *core* (núcleo) y mucha de la funcionalidad que podría considerarse básica para un servidor Web es provista por módulos. Algunos de estos son:

- mod_ssl - Comunicaciones Seguras vía TLS.
- mod_rewrite - Reescritura de direcciones servidas (generalmente utilizado para transformar páginas dinámicas como PHP en páginas estáticas HTML para así engañar a los navegantes o a los motores de búsqueda en cuanto a como fueron desarrolladas estas páginas).
- mod_deflate - Compresión transparente con el algoritmo *deflate* del contenido enviado al cliente.
- mod_auth_ldap - Permite autentificar usuarios contra un servidor LDAP.
- mod_proxy_ajp - Conector para enlazar con el servidor Jakarta Tomcat de páginas dinámicas en Java.

El servidor de base puede ser extendido con la inclusión de módulos externos entre los cuales se encuentran:

- mod_perl - Páginas dinámicas en Perl.
- mod_php - Páginas dinámicas en PHP.
- mod_python - Páginas dinámicas en Python.
- mod_aspdotnet - Páginas dinámicas en .NET de Microsoft.

2.4.2.3 Criterios de Selección

En esta sección se enfocara a la instalación y configuración del servidor Web Apache. Este servidor fue elegido de entre una amplia variedad por sus características:

- Robusto. Da soporte de un gran número de transacciones.
- Configurable para diferentes entornos de trabajo.
- Alto nivel de seguridad.
- Disponibilidad para una gran variedad de plataformas.
- Soporte para servicio de Proxy.
- Soporte para granjas de servidores.
- Soporte para Scripting languages integrados como módulos, ejemplo PHP, mod_perl, etc.
- Incluye el código fuente del servidor.
- Soporte para accesos restringidos.
- Soporte para SSL.
- Es libre y gratuito.

2.4.2.4 Instalación

Apache puede ser descargado de la dirección http://www.apache.org, los siguientes pasos pueden seguirse para la instalación en cualquier Unix o Linux:

```
$ tar zxvf httpd-2.0.54.tar.gz
$ cd httpd-2.0.54
$ ./configure
$ make
$ make install
```

Al ejecutar el script de configuración *configure* se puede emplear la opción *--prefix* para decirle la ruta en la que se hará la instalación. La sintaxis de configure con prefix es:

./configure –prefix=/algun/directorio

2.4.3 Configuración del Servidor

2.4.3.1 Directivas de Apache

El comportamiento de Apache se determina a través de directivas. Las directivas son parámetros de configuración que le dicen al servidor de Web como debe comportarse. Las directivas se encuentran en el archivo de configuración ***httpd.conf*** ubicado en */ruta/instalación/apache/conf*.

El administrador controla que directivas estarán disponibles de acuerdo a los módulos que integre en Apache. Si se emplea una directiva será necesario disponer del módulo que la soporta.

Las directivas de Apache se dividen en tres grupos:

- Ambiente Global (Global Environment)
- Servidor Principal (Main Server)
- Sitios Virtuales (Virtual Servers)

2.4.3.1.1 Sección Global Environment

La sección Global Environment administra las directivas generales de operación del Servidor de Web, es decir, que puertos escuchara, definirá el usuario que podrá realizar la ejecución de Apache y las configuraciones iniciales del servidor.

Entre las directivas más importantes de la sección global están:

Listen	Define cual es el puerto en el que Apache escuchara, por ejemplo, cuando un usuario ingresa en su navegador la dirección https://algun.dominio:**8443**, el número indica el puerto y en la configuración de apache deberá existir una directiva ***Listen 8443***. Es posible indicarle a Apache que escuche en más de un puerto. El puerto por defecto es el 80

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

ServerAdmin gabrielg@ejemplo.com

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

ServerName — Define el nombre para el servidor Apache el cual será utilizado al construir los URL's cuando el cliente solicite otros recursos Web del servidor. Ejemplo:

ServerName saruman.dcaa.unam.mx

DocumentRoot — Define la ruta absoluta donde se almacenarán los archivos HTML o scripts PHP que se desean publicar. Ejemplo:

DocumentRoot "/usr/local/apache/htdocs"

Cabe mencionar que esta directiva es usada para la sección del servidor principal y para los servidores virtuales.

ServerRoot — Define la ruta absoluta donde se instala Apache o donde los directorios *conf, logs, bin, modules,* podrán ser encontrados. Ejemplo:

ServerRoot "/usr/local/apache"

MinSpareServers — Define el número mínimo de instancias de Apache que son mantenidos en reserva.

Apache monitorea el número de procesos en reserva, si esté es menor a *MinSpareServers* levanta los procesos necesarios. El objetivo de esta directiva es mantener no menos del mínimo. Ejemplo:

MinSpareServers 7

El valor por defecto es 5.

MaxSpareServers — Define el número máximo de instancias de Apache que son mantenidos en reserva.

Al igual que *MinSpareServers*, Apache monitorea el número de procesos en reserva, si este es mayor al número que indica *MaxSpareServers* se eliminan siempre y cuando no estén realizando alguna tarea. Ejemplo:

MaxSpareServers 20

StartServers — Define el número de servidores que se inician cuando Apache arranca. Ejemplo:

StartServers 5

El valor por defecto es 5.

MaxClients

Define el número máximo de procesos se Apache que podrán ser iniciados con el objetivo de atender las solicitudes de los clientes. Ejemplo:

MaxClients 80

Apache no debe superar este número, es el máximo de procesos que puede mantener en operación.

El valor por defecto es 20

ErrorDocument

Si al recibir la petición de un documento ocurre algún problema con Apache, es posible configurarlo para que haga una de las siguientes opciones:

- Enviar un mensaje de error (opción por defecto).
- Enviar un mensaje personalizado.
- Redirigir a una URL local quien manejara el problema o el error.
- Redirigir a una URL externa quien manejara el problema o el error.

Ejemplo:

ErrorDocument 404 "Lo sentimos, el recurso solicitado no existe"

DirectoryIndex

Define el archivo que se mostrara por defecto si se solicita una URL que corresponde a un directorio. Ejemplo:

DirectoryIndex portada.html

PidFile

Define la ruta del archivo PidFile donde se almacenara el ID del proceso httpd padre. Ejemplo:

PidFile logs/apache.pid

Directory

Permite aplicar otras directivas de forma específica a todos los recursos dentro de la ruta especificada. A su vez, Directory está conformada por otras directivas como:

options	Controla las características del servidor que estarán disponibles para un directorio en particular. Ejemplo.
	options [+|-] opciones
all	Todas la opciones se activan excepto MultiViews
ExecCGI	Permite la ejecución de scripts CGI
Includes	Permite el uso de archivos incluidos del lado del servidor
Indexes	Si una URL mapea a un directorio solicitado y no hay un DirectoryIndex definido, entonces el servidor devolverá un listado de dicho directorio.

Ejemplo:

<Directory /usr/local/apache/htdocs/Ejemplo/privado>
Options -Indexes
</Directory>

2.4.3.1.2 Sección Main Server

Las directivas definidas en esta sección se encargan de responder a todas las peticiones hechas al servidor principal y que no son manejadas por ninguna directiva *VirtualHost*. Los valores definidos por la sección principal del servidor también proveen los valores por defecto que los contenedores *VirtualHost* emplean.

Muchas de las directivas mostradas anteriormente pueden aparecer dentro de los contenedores *VirtualHost*, en este caso, a los valores tomados por defecto se les asignara el valor que se esta definiendo dentro del VirtualHost.

Entre las directivas más importantes de la sección principal del servidor están:

User (usuario)	Define el identificador de usuario con el cual se ejecutara Apache. El usuario por defecto es User #-1
Group (grupo)	Define el identificador del grupo del usuario con el cual se ejecutara Apache.

2.4.3.1.3 Sección Sitios Virtuales

La sección ***VirtualHost*** administra las directivas donde los mismos procesos de Apache soportan diversas direcciones IP o nombres de dominio.

Las directivas empleadas para el manejo de sitios virtuales serán mostradas y explicadas en la sección *Manejo de Sitios Virtuales.*

2.4.4 Ejecución del Servidor

2.4.4.1 apachectl

Apachectl es una interfaz de usuario al servidor HTTP de Apache. Esta diseñado para ayudar al administrador a controlar el funcionamiento del demonio **httpd** Apache. Este demonio se ejecuta es segundo plano para atender las peticiones al servidor.
Apachectl requiere uno de los siguientes argumentos para su funcionamiento:

start	Inicia el servicio de Apache
stop	Detiene el servicio de Apache
restart	Reinicia el servicio de Apache, si éste no esta corriendo entonces es arrancado. Esta opción automáticamente revisa el archivo de configuración httpd.conf a través de *configtest* antes de reiniciar el servicio para evitar que el servidor caiga.
graceful	Igual que *restart.* La única diferencia entre el reinicio normal y este

	es que si existen conexiones abiertas estas no serán cerradas al reiniciar el servidor.
configtest	Realiza una prueba de sintaxis al archivo de configuración *httpd.conf* y en caso de existir algún error lo reporta.

Este script fija determinadas variables de entorno que son necesarias para que httpd funcione correctamente en el sistema operativo y después invoca al binario httpd. El script apachectl pasa a httpd cualquier argumento que se le pase a través de la línea de comandos, de forma que cualquier opción de httpd puede ser usada también con apachectl. Se puede editar directamente el script apachectl y cambiar la variable **HTTPD**, esta variable está al principio y en ella se especifica la ubicación exacta en la que el binario de httpd está ubicado.

2.4.4.2 Inicio del Servidor

Si el puerto especificado en el archivo de configuración es el puerto por defecto, 80 (o cualquier otro por debajo de 1024), es necesario tener privilegios de administrador para poder arrancar Apache en ese puerto privilegiado. Una vez que el servidor ha arrancado y completado algunas actividades preeliminares como la apertura de sus archivos de log, lanzará varios procesos *hijo* que hacen el trabajo de escuchar y responder las peticiones de los clientes. El proceso principal *httpd* continua corriendo como usuario root, pero los procesos hijo se ejecutan con un usuario con menos privilegios.

La forma recomendada para invocar al programa httpd es usando el script de control *apachectl* con la opción *start*. Lo primero que hace httpd cuando es invocado es localizar y leer el archivo de configuración httpd.conf. El lugar en el que httpd.conf está se determina al compilar, pero también es posible especificar la ubicación en la que se encuentra al iniciar el servidor Apache usando la opción –f, por ejemplo:

apachectl –f ruta/conf/httpd.conf start

Si todo va bien durante el arranque, la sesión de consola se suspenderá un momento y volverá a estar activa casi inmediatamente. Se puede usar el navegador para conectarse al servidor y ver la página de prueba que hay en el directorio DocumentRoot, para verificar que el servidor Apache se ha iniciado se puede ejecutar el comando:

ps – fea | *grep httpd*

2.4.4.2.1 Errores durante el Arranque

Si Apache encuentra un error irrecuperable durante el arranque, se escribirá un mensaje describiendo el problema en la consola o en el archivo ***ErrorLog*** antes de abortar la ejecución. Uno de los mensajes de error más comunes es "*Unable to bind…"*, cuando se escribe este mensaje, normalmente se debe a alguna de las siguientes razones:

- Se está intentando iniciar el servidor Apache en un puerto privilegiado (puerto 0 al 1024) sin ser el usuario root.
- Se está intentando iniciar el servidor Apache cuando éste ya se encuentra corriendo o cuando algún otro servidor Web está ocupando el mismo puerto.

2.4.4.2.2 Iniciar Apache al arrancar el Sistema Operativo

Si se requiere que el servidor Apache continúe su ejecución después de reiniciar el sistema, es necesario añadir una llamada a apachectl en los archivos de arranque (normalmente rc.d o un archivo en ese directorio del tipo rc*N*.d). Esto iniciará Apache como root.

El script apachectl está diseñado para actuar como un script estándar de tipo SysV init; puede tomar los argumentos Stara, restart y stop y traducirlos en las señales apropiadas para httpd. De esta manera, casi siempre se puede enlazar apachectl con el directorio init adecuado. Para iniciar Apache al arrancar el sistema se realiza lo siguiente:

$ cp /ruta/bin/apachectl /etc/init.d
$ cd /etc/rc*N*.d
$ln –s ../init.d/apachectl S99apache

Donde *N* es el nivel de preferencia en el que el servidor es levantado después de iniciar.

2.4.4.3 Detener Apache

Enviar las señales ***TERM*** o stop al proceso padre hace que se intenten eliminar todos los procesos hijo inmediatamente. Esto puede tardar algunos minutos. Una vez que se hayan terminado todos los procesos hijo, terminara el proceso padre. Cualquier petición en este proceso terminara inmediatamente y ninguna petición posterior será atendida.

Esto se logra de la siguiente manera:

apachectl stop

2.4.4.4 Reiniciar Apache

El envío de las señales ***HUP*** o restart al proceso padre hace que los procesos hijo terminen como si se les hubiese mandado la señal TERM para eliminar al proceso padre. La diferencia está en que estas señales vuelven a leer los archivos de configuración y vuelven a abrir los archivos log. Se genera un nuevo conjunto de procesos hijo y se continúa sirviendo peticiones.

Esto se logra de la siguiente manera:

apachectl restart

2.4.5 Incorporación de Módulos

Apache es un servidor modular. Esto implica que sólo las funcionalidades más básicas están incluidas en la instalación por defecto. Las características extras son habilitadas a través de módulos que pueden ser interpretados desde Apache. Por defecto, un juego de módulos base son incluidos en el servidor durante la compilación. Si el servidor es compilado para usar módulos cargados dinámicamente, entonces los módulos adicionales pueden ser compilados por separado y añadidos en cualquier

momento utilizando la directiva ***LoadModule***. Las directivas de configuración pueden ser incluidas en una condición utilizando un bloque ***<ifModule>***.

Para listar cada uno de los módulos actualmente compilados e instalados en el servidor se utiliza la opción –l

apachectl –l

2.4.5.1 Instalación de los módulos de PHP y MySQL de forma dinámica

1. Descargar los archivos fuente de PHP y MySQL de los sitios www.php.net y www.mysql.com respectivamente.
2. Extraer los archivos fuente de PHP y MySQL
   Para MySQL:

   ```
   tar-zxvfmysql-5.xxx.tar.gz
   cdmysql-5xxx
   ./configure --prefix=/usr/local/mysql5
   make
   makeinstall
   cp support-files/my-medium.cnf/etc/my.cnf
   groupaddmysql
   useradd-g mysqlmysql
   cd/usr/local/mysql5
   /usr/local/mysql5/bin/mysql_install_db--user=mysql
   chown-R root.
   chown-R mysqlvar
   chgrp-R mysql .
   ```

   Para PHP:

   ```
   tar-zxvfphp-5.xxx.tar.gz
   cd php-5.xxx/
   ./configure --prefix=/usr/local/php5
   --with-apxs2=/usr/local/apache2/bin/apxs
   --with-mysql=/usr/local/mysql5
   make
   make install
   cp php.ini-dist/usr/local/php5/lib/php.ini
   ```

3. Agregar las siguientes líneas al archivo httpd.conf de Apache

   ```
   AddTypeapplication/x-httpd-php.php.phtml
   AddTypeapplication/x-httpd-php-source.phps
   ```

4. Reiniciar Apache

   ```
   apachectl restart
   ```

Hasta este punto se ha terminado la instalación. Para revisar la configuración se crea un archivo llamado *info.php* En la carpeta htdocs ubicada en /usr/local/apache2/ con lo siguiente:

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

En cualquier navegador se visita la dirección 127.0.0.1/info.php para visualizar la información que info.php tiene como se muestra en la figura 20

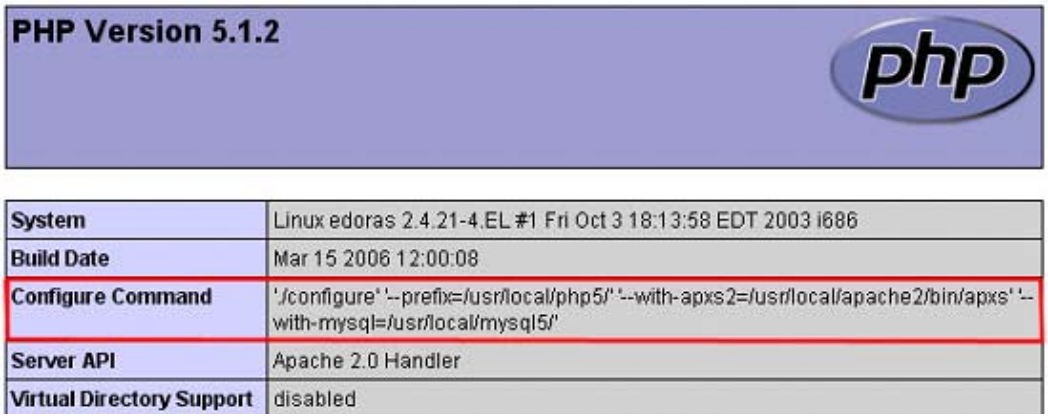


Figura 20 – info.php

El contenido del campo *Comando de Configuración* (Configure Command) debe coincidir con las rutas donde se instalaron cada uno de los módulos.

2.4.6 Accesos Restringidos

Apache ofrece la funcionalidad al estilo ACL[2] con lo cual es factible asignar un usuario y una contraseña por usuario quienes tendrán acceso sobre recursos específicos del servidor Web.

Los archivos especiales son usualmente llamados *.htaccess*, pero cualquier nombre puede ser especificado en la directiva ***AuthUserFile***. Las directivas colocadas en los archivos .htaccess siguen la misma sintaxis que los archivos de configuración principales. Desde que los archivos .htaccess son leídos en cada petición, los cambios realizados en estos archivos toman efecto inmediato.

Las directivas empleadas son:

AuthUserFile — Indica cual es el archivo que contiene la lista de usuarios y contraseñas para realizar la autenticación. Ejemplo:

AuthUserFile /usr/apache/conf/claves/.htpasswd

Es importante que este archivo no sea accesible por los clientes del servidor, ya que tendrían acceso a la lista de usuarios y contraseñas y tienen la posibilidad de atacarla por métodos de diccionario o incluso fuerza bruta.

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

[2] Lista de Control de Acceso por su nombre en inglés *Access Control List*

AuthGroupFile /dev/null

AuthName — Define el nombre de autorización del recurso protegido, éste aparecerá como un mensaje en la caja de diálogo que solicita el usuario y la contraseña para visualizar el contenido del recurso protegido. Ejemplo:

AuthName “Bienvenidos al Sitio Privado"

require valid-user — Indica que se requiere de un usuario y una contraseña valida para obtener un recurso determinado.

Un ejemplo muy útil sobre el uso de accesos restringidos es permitir a ciertos usuarios visitar el contenido de un sitio privado, como se muestra a continuación:

1. Crear un archivo con la lista de usuarios y contraseñas que el servidor utilizara, este archivo debe colocarse en un directorio que no pueda ser visto desde el Web, por ejemplo en un subdirectorio creado por nosotros en el directorio conf de Apache. Para crear este archivo se emplea la utilería *ntpasswd* que se encuentra en el directorio ***bin*** de Apache.

   htpasswd -cb /usr/local/apache2/conf/claves/.htpasswd gabriel secreto

2. En el archivo httpd.conf se agregan las siguientes directivas que restringirán el acceso al directorio elegido.

```
<Directory /usr/local/apache/htdocs/privado >
   AuthUserFile /usr/local/apache/conf/claves/.htpasswd
   AuthGroupFile /dev/null
   AuthName "Ingrese su usuario y contraseña"
   AuthType Basic
    require valid-user
</Directory>
```

3. Al intentar ver el recurso protegido el navegador pedirá el usuario y contraseña que fueron especificados en el archivo .htpasswd a través de htpasswd.

2.4.7 Registro de Accesos

2.4.7.1 Precauciones de seguridad

Cualquiera que pueda escribir en el directorio donde Apache está escribiendo un archivo de bitácora puede casi seguramente obtener acceso al UID con el cual el servidor es iniciado el cual normalmente es root. No es recomendable dar a los usuarios permisos de escritura en el directorio donde las bitácoras están siendo almacenadas.

Además, los archivos de bitácoras pueden contener información proporcionada directamente por el cliente, por tal razón es posible que clientes maliciosos pueden insertar caracteres de control en estos archivos.

2.4.7.2 Bitácoras de Acceso

La bitácora de accesos a servidor graba todas los solicitudes procesadas por el servidor, La localización y contenido de este log de acceso es controlada por la directiva "***CustomLog"***. La directiva "***LongFormat"*** se usa para simplificar la selección del contenido de las bitácoras.

Por su puesto, almacenar la información en las bitácoras de accesos es sólo el inicio de la administración de estos archivos. Lo siguiente es analizar la información para producir estadísticas. Diferentes versiones de Apache han usado otros módulos y directivas para controlar la bitácora de acceso, incluyendo *mod_log_referer, mod_log_agent* y la directiva *transferLog.* La directiva ***CustomLog*** absorbe la funcionalidad de todas estas antiguas directivas.

El formato de la bitácora de accesos es altamente configurable. El formato es especificado usando una cadena de formato que se asemeja en gran parte al estilo del lenguaje de programación C generado por la instrucción para el formato de cadenas *printf*.

2.4.7.3 Bitácoras de Error

La bitácora para los errores del servidor es el archivo de log más importante, es aquí donde Apache enviará información de diagnostico y grabara cualquier error que es encontrado durante el proceso de una solicitud. Este es el primer lugar donde se debe buscar cuando ocurre un problema al iniciar el servidor o al realizar alguna de las operaciones del mismo, además, puede contener detalles de que fue lo que sucedió y provee de pistas para solucionar el problema.

La bitácora de error usualmente es escrita en un archivo llamado *error_log*. El formato de esta bitácora es descriptivo y de forma libre, por ejemplo, la siguiente línea es un mensaje típico:

```
[Wed Apr 10 14:35:56 2006] [error] [client 127.0.0.1] client denied by
server configuration: /usr/local/apache2/htdocs/privado
```

El primer elemento es la fecha y la hora del mensaje. La segunda entrada lista la severidad del error reportado. Por último se muestra la ruta en el sistema de archivos donde se encuentra el elemento solicitado.

2.4.7.4 Directivas de las Bitácoras

ErrorLog	Define el nombre del archivo donde el servidor Web Apache registrara los errores que se presenten durante su operación. Si el nombre del archivo para la bitácora de errores comienza con una diagonal (/) se considera que la ruta tiene origen en la raíz del sistema de archivos de lo contrario se toma como origen la ruta determinada por ServerRoot. Ejemplo: ***ErrorLog logs/errores_log***
LogLevel	Define el nivel de mensajes de error que serán registrados en la

bitácora de errores determinada por ErrorLog. Cuando un nivel es especificado, se reportan los mensajes de todos los niveles de mayor importancia incluyendo al nivel especificado.

Los niveles son:

Nivel	Descripción
emerg	Emergencias – El sistema es inutilizable
alert	Una acción debe ser tomada inmediatamente
crit	Condiciones criticas
error	Condiciones de error
warn	Condiciones de aviso
notice	Condiciones normales pero significantes
info	Informativa
debug	Mensajes de depuración

Ejemplo:

LogLevel debug

CustomLog

Define el nombre del archivo donde Apache registrara los accesos de los clientes al servidor Web.

Si el nombre del archivo para la bitácora de accesos comienza con una diagonal (/) se considera que la ruta tiene origen en la raíz del sistema de archivos de lo contrario se toma como origen la ruta determinada por ServerRoot. Ejemplo:

CustomLog logs/accesos_log

LongFormat

Define un formato para los archivos de bitácoras en base a directivas de porcentaje. Cada una de ellas del dice al servidor que almacene una pieza particular de información. También es posible colocar caracteres literales en el formato de cadena para ser impresos directamente en la bitácora.

Las directivas de LongFormat son:

Directiva	Descripción
%h	Registra la IP del cliente
%l	Si el demonio identd corre en el cliente, reporta la información que este devuelva.
%u	Si se requiere de un usuario y una contraseña para acceder entonces se registra.
%t	Fecha y hora en que se hace una solicitud en el formato [día/mes/año:hora:minuto:segundo]
\"%r\"	Recurso solicitado por el cliente entrecomillado.
%>s	Código de tres dígitos donde se muestra el valor del status devuelto al cliente.
%b	Número de bytes devueltos al cliente exceptuando encabezados.

Después de especificar la cadena de formato puede ser asociado con un alias. Ejemplo:

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

Existe una serie de cadenas de formato ya definidos en la configuración del httpd.conf:

- **Common**. Orientado a obtener la lista de direcciones y fechas en los clientes que se conectan al servidor así como el status que este devolvió.
- **Referer**. Orientado a llevar un registro de las rutas que el cliente visito dentro del servidor.
- **Agent**. Orientado a registrar el nombre de los navegadores del cliente.
- **Combined**. Combina las características de los tres anteriores.

2.4.8 Sitios Virtuales

Mediante los sitios virtuales se manejan varios dominios apuntando a una misma máquina. Dentro de cada sitio se definen las directivas específicas para el dominio que esté representando.

Es posible definir un sitio virtual a través de una IP o bien, a través de un nombre de dominio. El sitio basado en IP's usa la dirección IP de la conexión para determinar qué sitio es el que se tiene que mostrar, por lo tanto, será necesario tener diferentes direcciones IP para cada sitio virtual. Los sitios virtuales basados en nombres atienen al nombre del sitio que especifica el cliente en las cabeceras de HTTP. Usando esta técnica, una sola dirección IP puede ser compartida por muchos sitios Web diferentes.

El sitio virtual basado en nombres normalmente es más sencillo ya que sólo es necesario configurar un servidor DNS para que localice la dirección IP correcta y dar de alta las directivas necesarias en el archivo de configuración httpd.conf. Una gran ventaja al hacer esto es que se reduce la demanda de direcciones IP.

2.4.8.1 Directivas de los Sitios Virtuales

VirtualHost — En un grupo de directivas las cuales se aplicaran sólo a un servidor en particular. Casi cualquier directiva mostrada anteriormente puede ser aplicada a un sitio virtual. Cuando el servidor recibe una petición de un documento en un sitio virtual usa este grupo de directivas. Ejemplo

```
<VirtualHost 10.1.2.3>
  ...
</VirtualHost>
```

NameVirtualHost — Designa una dirección IP (y posiblemente un puerto) en el servidor que estará atendiendo las solicitudes a los sitios virtuales. En el caso en que todas la direcciones IP del servidor serán ocupadas puede pasarse el carácter asterisco (*) a esta directiva. Ejemplo:

NameVirtualHost *:80

2.4.8.2 Creación de un Sitio Virtual

1. Definir la estructura que se utilizara para implementar el sitio virtual, suponiendo que todos los contenidos serán colocados en la ruta /usr/local/apache2/sitiovirtual. Para fines de este ejemplo se crearan dos directorios que funcionaran como repositorios para el contenido de dichos sitios.

```
$make /usr/local/apache2/sitiovirtual
$make /usr/local/apache2/sitiovirtual/sitio1
$make /usr/local/apache2/sitiovirtual/sitio2
```

2. Ahora será necesario definir la estructura de los sitios en el archivo httpd.conf

```
NameVirtualHost *:80

<VirtualHost 192.168.24.2:80>
        ServerAdmin administrador@192.168.24.2
        DocumentRoot /usr/local/apache2/sitiovirtual/sitio1
        ServerName www.dominio1.com
</VirtualHost>

<VirtualHost 192.168.24.2:80>
        ServerAdmin administrador@192.168.24.2
        DocumentRoot /usr/local/apache2/sitiovirtual/sitio2
        ServerName www.dominio2.com
</VirtualHost>
```

3. Ahora sólo es necesario agregar los nombres de los sitios en una tabla de DNS. Para probar el ejercicio se agregan las siguientes líneas al archivo *hosts* ubicado en el directorio /etc

```
192.168.24.2        www.dominio1.com
192.168.24.2        www.dominio2.com
```

2.5 Programación con PHP

2.5.1 Historia

PHP es un lenguaje creado por una gran comunidad de personas. El sistema fue desarrollado originalmente en el año 1994 por Rasmus Lerdorf como un CGI escrito en C que permitía la interpretación de un número limitado de comandos. El sistema fue denominado Personal Home Page Tools y adquirió relativo éxito gracias a que otras personas pidieron a Rasmus que les permitiese utilizar sus programas en sus propias páginas. Dada la aceptación del primer PHP y de manera adicional, su creador diseñó un sistema para procesar formularios al que le atribuyó el nombre de FI (de sus siglas en inglés *Form Interpreter*) y el conjunto de estas dos herramientas, sería la primera versión compacta del lenguaje: PHP/FI.

En el año de 1997, dos programadores israelíes de Technion: Zeev Suraski y Andi Gutmans reescribieron el analizador gramatical y crearon la base de PHP 3, cambiando el nombre del lenguaje a la forma actual. Experimentaciones públicas de PHP 3 comenzaron inmediatamente y fue lanzado oficialmente en junio del 1998.
Para 1999, Suraski y Gutmans reescribieron el código de PHP, produciendo lo que hoy se conoce como *Motor Zend*. En mayo de 2000 PHP 4 fue lanzado bajo el poder del

motor Zend Engine 1.0. El 13 de julio de 2004, PHP 5 fue lanzado, utilizando el motor *Motor Zend II*. La versión más reciente de PHP es la 5.1, que incluye el novedoso PDO (Objetos de Información de PHP o PHP Data Objects) y mejoras utilizando las ventajas que provee el nuevo Motor Zend II.

2.5.2 ¿Qué es PHP?

Es un acrónimo de Preprocesador de Hipertexto, es decir es un lenguaje interpretado de alto nivel embebido en páginas HTML y ejecutado en el servidor.

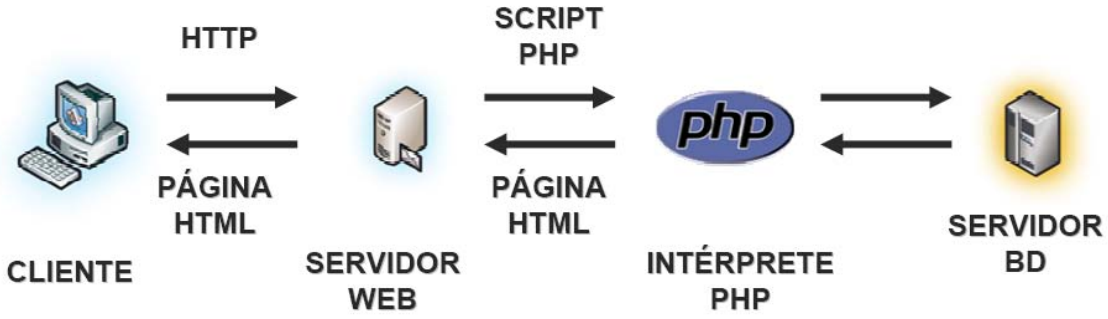


Figura 21 – Funcionamiento cliente - servidor

La mayoría de su sintaxis es similar a C, Java y Perl. La meta de este lenguaje es permitir escribir a los creadores de páginas Web, páginas dinámicas de una manera rápida y fácil, procesar formularios o mandar o recibir cookies.
Sin duda alguna la capacidad más importante es el soporte para una gran cantidad de base de datos.

2.5.3 Sintaxis de PHP

Existen cuatro maneras en las cuales se puede incorporar el código PHP en una página HTML.

<?
... código de php ...
?> [3]

<?php
... código de php ...
?>

<SCRIPT LANGUAGE="php">
... código de php ...
</SCRIPT>

<%
... código de php ...
%> [4]

[3] Para utilizar estos delimitadores es necesario tener habilitada la directiva short_open_tag=On
[4] Para utilizar estos delimitadores es necesario tener habilitada la directiva aspa_tags=On

2.5.4 Variables

Como la mayoría de los lenguajes de programación, PHP almacena todo tipo de datos en localidades de memoria a las que se les conoce como variables. Estos datos almacenados de forma temporal pueden ser recuperados para su uso por medio del nombre de dicha variable.

PHP es un lenguaje débilmente tipado, es decir, las variables no tienen asociada la naturaleza del tipo de información que almacenan, por tal motivo, una misma variable podrá almacenar distintos tipos de información.

La sintaxis para la declaración de una variable es:

$nombre_de_la_variable

Como se observa, todos los nombres de variable deben iniciar con el símbolo de pesos ($). Los nombres pueden contener letras, números y guiones bajos. PHP es sensible a mayúsculas y minúsculas.

2.5.4.1 Tipos de Datos

PHP tiene tres tipos de datos básicos: enteros, números con punto flotante y cadenas y dos datos de tipo compuesto: arreglos y objetos. Cada variable tiene un tipo específico, aunque como se mencionó anteriormente, este tipo puede cambiar en cuanto cambia el valor de la variable.

2.5.4.1.1 Enteros (Integer)

Los enteros son usados para representar números no decimales, estos números varían entre -2 billones y 2 billones, se pueden representar en formato decimal, octal o hexadecimal. Los enteros pueden ser especificados usando cualquiera de las siguientes sintaxis:

```
$var = 123;     #Un entero positivo
$var = -123;    #Un entero negativo
$var = 0123;    #Un número octal
```

2.5.4.1.2 Números con punto flotante (Float)

También conocidos como números de punto flotante o números reales, son usados para representar números que contienen un valor decimal o un exponente.

Los números reales pueden ser especificados usando cualquiera de las siguientes sintaxis:

```
$var = 1.234;
$var = 123.456e3;
$var = -123.45e-3;
$var = 0.00123;
```

2.5.4.1.3 Cadenas (String)

Las cadenas son usadas para representar valores no numéricos, estas pueden ser especificadas usando comillas dobles (") o comillas sencillas (').

Cuando se utilizan comillas dobles, las variables dentro de la cadena serán evaluadas, en el caso de las comillas sencillas dichas variables serán omitidas. Si se requiere imprimir caracteres especiales se utiliza la diagonal invertida (\) para escaparlos.

\n	Salto de línea
\r	Retorno de carro
\\	Diagonal invertida
\$	Signo de pesos
\”	Comilla doble
\’	Comilla simple

2.5.4.1.4 Arreglos (Arrays)

Un arreglo es una estructura que permite almacenar un conjunto de datos y referirse a ellos por el mismo nombre. PHP soporta arreglos de datos de distinto tipo. Se tienen dos tipos de arreglos: los simples y los asociativos.

Los arreglos simples pueden ser creados a través del constructor *array*():

$arreglo = array(“lunes”, “martes”, 24, 67.3);

Otra forma de generar el mismo arreglo es declararlo de forma explicita:

```
$arreglo[0] = “lunes”;
$arreglo[1] = “martes”;
$arreglo[2] = 24;
$arreglo[3] = 67.3;
```

El inicio de un arreglo simple siempre será la posición 0.

Una manera sencilla (pero no optima) de ver el contenido del arreglo anterior es accediendo directamente a sus elementos. Para imprimir en pantalla el contenido de las variables se emplea la función *echo*.

```
echo $arreglo[0];
echo $arreglo[1];
echo $arreglo[2];
echo $arreglo[3];
```

La forma más común de acceder al contenido de un arreglo simple es por medio de una estructura de repetición llamada ciclo *for* que será explicada más adelante.

2.5.4.1.5 Arreglos Asociativos

Son arreglos especiales, funcionan de la misma manera que un arreglo simple a diferencia de que en los arreglos asociativos los índices son valores de tipo cadena de modo que cada elemento del arreglo está definido por el par clave-valor.

De igual manera, los arreglos asociativos pueden ser creados a través del constructor *array*(), la sintaxis es:

$arreglo = array('dia' => "lunes", 'costo' => 12.65);

En forma explicita:

```
$arreglo['dia'] = "lunes";
$arreglo['costo'] = "12.65";
```

Donde día y costo corresponden a la clave y lunes y el número 12.65 corresponden al valor del elemento.

La forma más común de acceder al contenido de un arreglo asociativo es por medio de una estructura de repetición llamada ciclo *foreach* que será explicada más adelante.

2.5.5 Constantes

Una constante es una variable que mantiene su valor durante toda la ejecución del programa. La única manera de asignarle un valor a una constante es cuando ésta es creada.

Las constantes son útiles cuando se requiere manejar un valor que no cambiara en ninguna parte del programa como el valor de cambio de una moneda, tasa de I.V.A. o constantes matemáticas como PI.

PHP define varias constantes y proporciona un mecanismo para definir más en tiempo de ejecución.

Las constantes más comunes que se tienen en PHP son TRUE y FALSE que son empleadas en expresiones lógicas. La manera en se crean constantes personalizadas es con el uso de la función ***define()***.

Para definir constantes se utiliza la siguiente sintaxis:

define("NOMBRE_CONSTANTE", Valor_de_la_constante);

Para una constante no es necesario anteponer el símbolo de pesos al nombre de ésta como se haría comúnmente al declarar una variable normal, de igual modo, no será necesario anteponer el mismo signo para recuperar su valor. Por convención el nombre de las constantes se escribe en mayúsculas.

2.5.6 Comentarios

Los comentarios son líneas de texto dentro del código que serán ignoradas por el intérprete. Estas anotaciones son útiles para describir las funciones de algunas partes del código o tener una referencia de algún programa para fechas posteriores, esto ayuda a que el programador no pierda mucho tiempo en tratar de entender la estructura del código.

2.5.6.1 Sintaxis de los comentarios

PHP tiene dos tipos de comentarios: los comentarios de bloque y los comentarios de una sola línea.

/

Esto es un comentario de bloque
Todo lo que este contenido dentro de estas líneas será ignorado por el intérprete

****/***

// Esto es un comentario de sólo una línea

\# Este es otro tipo de comentario de una sola línea

2.5.7 Operadores

2.5.7.1 Aritméticos

Ejemplo	Operador	Resultado
$a + $b	+	Suma de $a y $b.
$a - $b	-	Diferencia entre $a y $b.
$a * $b	*	Producto de $a y $b.
$a / $b	/	Cociente de $a y $b.
$a % $b	%	Resto de $a dividido entre $b.

2.5.7.2 Asignación

Ejemplo	Operador	Resultado
$a = $b	=	El valor de $b será asignado a $a
$a += $b	+=	$a = $a + $b
$a -= $b	-=	$a = $a - $b
$a *= $b	*=	$a = $a * $b
$a /= $b	/=	$a = $a / $b

2.5.7.3 Cadenas

Ejemplo	Operador	Resultado
$a . $b	.	El valor de $a será concatenado al valor de $b
$a .= $b	.=	$a = $a . $b

2.5.7.4 Incremento y Decremento

Ejemplo	Operador	Resultado
$a++	++	Se emplea el valor de $a en alguna expresión y después este valor es incrementado en 1.
++$a	++	El valor de $a es incrementado en 1 y después es empleado en alguna expresión.
$a--	--	Se emplea el valor de $a en alguna expresión y después este valor es decrementado en 1.
--$a	--	El valor de $a es decrementado en 1 y después es empleado en alguna expresión.

2.5.7.5 Comparación

Ejemplo	Operador	Resultado
$a == $b	==	TRUE si $a es igual a $b.
$a === $b	===	TRUE si $a es igual a $b, y son del mismo tipo.
$a != $b	!=	TRUE si $a no es igual a $b.
$a <> $b	<>	TRUE si $a no es igual a $b.
$a !== $b	¡===	TRUE si $a no es igual a $b, o si no son del mismo tipo.
$a < $b	<	TRUE si $a es menor que $b.
$a > $b	>	TRUE si $a es mayor que $b.
$a <= $b	<=	TRUE si $a es menor o igual que $b.
$a >= $b	>=	TRUE si $a es mayor o igual que $b.

2.5.7.6 Lógicos

Ejemplo	Operador	Resultado
! $a	!	TRUE si $a no es TRUE.
$a && $b	&&	TRUE si tanto $a como $b son TRUE.
$a || $b	||	TRUE si cualquiera de $a o $b es TRUE.

2.5.8 Funciones

Una función es un bloque de código que puede ser definido una vez y ser ejecutado desde diferentes partes de un programa. Típicamente una función toma uno o más argumentos, ejecuta un conjunto de operaciones sobre ellos y a veces devolverá algún valor.

Cuando una función es ejecutada, la aplicación que la invoca le pasa el control, cuando la función termina su bloque de sentencias, ésta devuelve el control al programa que la mando ejecutar.

PHP incluye un gran número de funciones que ahorran tiempo de programación, entre las más comunes se encuentran:

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 vacía, 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.
strcmp($cad1,$cad2)	Compara dos cadenas y devuelve un valor menor a 0 si la segunda cadena es mayor que la primera; mayor que 0 si la primera es mayor que la segunda y 0 si son iguales, distingue entre mayúsculas y minúsculas.
strtoupper($cadena)	Convierte una cadena a mayúsculas.
Strtolower($cadena)	Convierte una cadena a minúsculas.

De igual manera, PHP permite definir funciones personalizadas según se requiera. El criterio que hay que tener para crear una función es que ésta sólo deberá dedicarse a realizar una tarea específica.

Una función se define con la siguiente sintaxis:

```
function nombreFuncion(/* Lista de parámetros */) {

        /* Bloque de código */

        return $resultado       // En caso de que la función devuelva algún valor
}
```

2.5.8.1 Paso de Parámetros por Valor

La función recibe una copia del valor de la variable, ejecuta su bloque de sentencias y cuando termina de ejecutarse, ésta no habrá modificado el valor de la variable que recibió como argumento.

Ejemplo:

```
function exponente($a){
        return $a * $a;
}
```

2.5.8.2 Paso de Parámetros 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. Para especificar la referencia a una variable se antepone un símbolo de ampersand (&) al nombre.

Ejemplo:

```
function exponente(&$a){
        return $a * $a;
}
```

2.5.8.3 Paso de Parámetros 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. Los parámetros por defecto se ponen después de los requeridos.

Ejemplo:

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

2.5.9 Estructuras de Control

Las estructuras de control o sentencias de control modifican el flujo de ejecución de un programa haciendo que la ejecución no tenga que ser secuencial, sino que permite bifurcar el flujo del programa (estructuras condicionales) o que cierta porción de código se ejecute un determinado número de veces (estructuras cíclicas).

2.5.9.1 Estructuras Condicionales

Son estructuras que permiten elegir que bloques de código se ejecutaran en base a una condición determinada.

En PHP existen dos tipos de estructuras condicionales:

2.5.9.1.1 if

La sentencia if ejecuta un conjunto de sentencias englobadas dentro de ella si se cumple una condición *booleana*, es decir, if evaluara si el resultado de una operación arroja como resultado un *falso* o un *verdadero*.

La sintaxis de if es:

```
if ( Expresión ){
   /* bloque de sentencias a ejecutar */
}
```

El bloque de sentencias dentro de if sólo se ejecutara si el resultado de alguna comparación es verdadero, en el caso contrario, este bloque se omitirá. Por ejemplo:

```
<?

/* Si el valor de $a es mayor al de $b entonces imprime un mensaje */
if ( $a > $b ){
   echo “$a es mayor a $b”;
}

?>
```

Una variante de if es el if/else. Esta variante funciona de la misma manera que if, sólo que esta incluye un bloque de sentencias que se ejecutara si la condición evaluada fuera falsa. Por ejemplo, si se agregara la parte else al código anterior para que imprimiera un mensaje cuando el valor de $b sea mayor al de $a se tendría lo siguiente:

```
<?

/* Si el valor de $a es mayor al de $b entonces imprime un mensaje */
if ( $a > $b ){
        echo “$a es mayor a $b”;
} else {
        echo $b es mayor a $a”;
}

?>
```

La parte correspondiente a else no necesita llevar una expresión a comparar. Si el resultado de la comparación ejecutada por if es falsa entonces se ejecutara el bloque de sentencias comprendido por else.

PHP también ofrece la estructura i/elseif/else. *Elseif*, como su nombre sugiere, es una combinación de *if* y *else*. Como *else*, extiende una sentencia *if* para ejecutar una sentencia diferente en caso de que la expresión *if* original se evalúa como falsa. No obstante, a diferencia de *else*, ejecutará esa expresión alternativa solamente si la expresión condicional *elseif* se evalúa como verdadera. Por ejemplo, el siguiente código evaluara si *a* es mayor que *b*, *a* es igual a *b* o *a* es menor que *b*:

```
<?

 if ($a > $b) {
        echo "$a es mayor que $b";
 } elseif ($a == $b) {
        echo "$a es igual que $b";
 } else {
        echo "$a es mayor que $b";
 }

?>
```

2.5.9.1.2 switch

La sentencia *switch* es similar a una serie de sentencias IF en la misma expresión. En muchas ocasiones, se quiere comparar la misma variable (o expresión) con muchos

valores diferentes, y ejecutar una parte de código distinta dependiendo de a qué valor es igual.

La sintaxis de switch es.

```
switch($variable){
        case valor1:
                /* bloque de sentencias a ejecutar */
                break;
        case valor2:
                /* bloque de sentencias a ejecutar */
                break;
        case valorN:
                /* bloque de sentencias a ejecutar */
                break;
        default:
                sentencias;
}
```

La sentencia *switch* ejecuta línea por línea (realmente, sentencia a sentencia). Al comienzo, no se ejecuta código. Sólo cuando se encuentra una sentencia *case* con un valor que coincide con el valor de la expresión *switch* PHP comienza a ejecutar las sentencias. PHP continúa ejecutando las sentencias hasta el final del bloque *switch*, o la primera vez que vea una sentencia *break*. Si no se escribe una sentencia *break* al final de una lista de sentencias case, PHP seguirá ejecutando las sentencias del siguiente case.

Un caso especial es el *default*. Este "case" coincide con todo lo que no coincidan los otros case, es decir, todo lo que no este especificado entrara al bloque deault.

2.5.9.2 Estructuras de Repetición

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 cuatro tipos:

2.5.9.2.1 for

Realiza un conjunto de instrucciones un determinado número de veces.

La sintaxis de for es:

```
for( inicialización;condición;incremento ) {
        /* Bloque de sentencias */
}
```

La primera expresión (*inicialización*) se evalúa (ejecuta) incondicionalmente una vez al principio del ciclo, ésta indica el valor que tiene la variable que posteriormente se evaluara en una condición. Al comienzo de cada iteración, se evalúa condición. Si se evalúa como verdadera, el ciclo continúa y las sentencias anidadas se ejecutan. Si se

evalúa como falsa, la ejecución del ciclo finaliza. Al final de cada iteración, se evalúa (ejecuta) *incremento*. Ejemplo:

```
<?

        echo "Listado del 1 al 10\n";
        for($i=1;$i<=10;$i++){
                echo "$i\n";
        }

?>
```

La forma más optima de trabajar con arreglos es con el uso de las estructuras de repetición, en el caso de un arreglo simple, se emplea un for para recorrer todos los elementos de un arreglo, de este modo se garantiza que se comience desde el primer elemento del arreglo hasta el último sin importar cuantos sean. Ejemplo:

```
<?
        $producto[0] = 1;
        $producto[1] = "chiles";
        $producto[2] = "herdez";
        $producto[3] = 7.89;

        echo "Tamaño del arreglo : " . count($producto) . "\n";
        echo "Tamaño del arreglo : " . sizeof($producto) . "\n";

        echo "\n";
        for($cont=0;$cont<count($producto);$cont++)
                echo $producto[$cont] . "\n";

        for($cont=0;$cont<sizeof($producto);$cont++)
                echo $producto[$cont] . "\n";

?>
```

La inicialización de la variable *icontador* comienza en cero (primer elemento del arreglo). En este código, se auxilia de las funciones ***count*** o ***sizeof***, ambas devuelven el tamaño del arreglo que en ese caso es de 4 (del 0 al 3) y que ayudaran a establecer la condición que determinará si el ciclo se repetirá una vez más o no.

La tercera parte del for, indicara que la variable cont será incrementada de uno en uno.

2.5.9.2.2 foreach

Foreach es un modo fácil de iterar sobre matrices. *Foreach* funciona solamente con arreglos de este tipo y devolverá un error si se intenta utilizar con otro tipo de datos o variables no inicializadas.

Se utiliza para recorrer las estructuras de tipo arreglo, obteniendo en cada iteración uno de sus elementos componentes.

La sintaxis de foreach es:

```
foreach(nombre_arregloas $clave=> $valor){
        /* Bloque de sentencias */
}
```

La forma más optima de trabajar con arreglos es con el uso de las estructuras de repetición, en el caso de un arreglo asociativo, se emplea un foreach para recorrer todos los elementos de un arreglo, de este modo se garantiza que se comience desde el primer elemento del arreglo hasta el último sin importar cuantos sean. Ejemplo:

```
<?
        $producto['ID'] = 1;
        $producto['NOMBRE'] = "chiles";
        $producto['MARCA'] = "herdez";
        $producto['EXISTENCIA'] = 15;
        $producto['INFORMACION ADICIONAL'] = "Producto Agotado";
        $producto['COSTO'] = 7.89;

        foreach($productoas $key=> $value)
                echo $key= $value . "\n";
?>
```

En el foreach se aprecia que se pasa el nombre del arreglo y se especifica el par de variables que recibirán la clave y el valor de cada uno de los elementos del arreglo asociativo. El código anterior escribiría el valor del *string* que funge como índice y el valor de este.

2.5.9.2.3 while

La sentencia while le dice a PHP que ejecute las sentencias anidadas repetidamente, mientras la expresión while se evalúe como verdadera. El valor de la expresión es comprobado cada vez al principio del ciclo, así que incluso si este valor cambia durante la ejecución de las sentencias anidadas, la ejecución no parará hasta el fin de la iteración (cada vez que PHP ejecuta las sentencias en el ciclo es una iteración). A veces, si la expresión while se evalúa como falsa desde el principio, si este fuera el caso las sentencias no se ejecutarán ni siquiera una vez.

La sintaxis de while es:

```
while(condición){
        /* Bloque de sentencias */
}


<?
        $i = 1;

        /* mientras $i sea menor a 10, imprime la serie de números */
        while ($i <= 10) {
                echo “$i \n”;
                $i++;
        }
?>
```

2.5.9.2.4 do… while

Los ciclos do..while son muy similares a los while, excepto que las condiciones se comprueban al final de cada iteración en vez de al principio. La principal diferencia frente a los ciclos regulares while es que se garantiza la ejecución de la primera iteración de un ciclo do..while (la condición se comprueba sólo al final de la iteración), mientras que puede no ser necesariamente ejecutada con un ciclo while regular (la condición se comprueba al principio de cada iteración, si esta se evalúa como falsa desde el principio la ejecución del ciclo finalizará inmediatamente). Ejemplo:

```
<?
        $i = 0;
         do {
                echo $i;
        } while ($i>0);
?>
```

El ciclo de arriba se ejecutaría exactamente una sola vez, después de la primera iteración, cuando la condición se comprueba, se evalúa como falsa ($i no es más grande que 0) y la ejecución del ciclo finaliza.

2.5.9.3 break y continue

Estas sentencias se emplean para controlar el flujo del programa una vez que se entra en alguna de las estructuras de selección o repetición.

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 pasar a la siguiente iteración sin romper la estructura de repetición.

2.5.10 Inclusión de Archivos

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

2.5.10.1 require e include

include() permite incluir el mismo archivo en varias ocasiones dentro del mismo programa, produce un aviso si no logra encontrar el archivo que se especifico pero continuara con la ejecución de la siguiente línea en el código.

Por otra parte, con require() sólo podrá incluirse el archivo una vez. require() produce un error fatal si no encuentra el archivo especificado y detendrá la ejecución del script.

Ejemplo:

```
<html>
<head>
<title>Inclusión de Archivos</title>
</head>
<body>

<?
        require "tesina/funcPassw.php";
        include " cabecerasitio.php";
        echo "<h1 align='center'>Página principal</h1>";

        echo "<br><br><center>Generando una contraseña : " . genPassw() . "</center>";
?>
</body>
</html>
```

En el código anterior, la función *require* generalmente es empleada cuando se requiere que la inclusión de un archivo sea forzosa, su argumento puede ir entre paréntesis y ser una ruta relativa o absoluta. Después de hacer la inclusión del archivo se puede acceder a las funciones y/o variables que se encuentran dentro de él, en este caso, al incluir el script ***funcPassw.php*** que se encuentra en el directorio ***tesina/*** es posible emplear la función *genPassw()* que se encuentra definida ahí.

Include funciona de una forma muy similar, la diferencia está en que si include no encuentra el archivo ***cabecerasitio.php*** en la ruta que se le especifico, este mandara un aviso con el mensaje correspondiente pero no interrumpirá la ejecución del programa.

2.5.11 CGI (Common Gateway Interface)

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

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

La esencia del CGI es que cuando el cliente solicita dicho programa, en vez de ser descargarlo, éste será ejecutado por el servidor y devolverá sólo los resultados al cliente.

Los CGI’s tienen diversos usos, entre los más comunes están:

- 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.

2.5.11.1 Lenguajes Interpretados

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. Para hacer uso de estos programas no hay necesidad de generar archivos binarios.

Entre los leguajes interpretados más comunes se encuentran: ASP, PERL, PHP.

Ventajas: Se puede editar y probar de forma rápida ya que no requiere el proceso de volver a compilar.

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.

2.5.11.2 Lenguajes 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 el proceso será abortado y éste nunca podrá ser ejecutado.

Para codificar el código fuente a un programa ejecutable o código binario se requiere de un compilador. Un compilador traduce un programa una vez. 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.

Entre los lenguajes compilados más comunes se encuentran: 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.

2.5.12 Diseñar una Aplicación CGI

2.5.12.1 Entrada de datos a partir de formularios

La manera en como se envían los datos del usuario al script de PHP para su procesamiento es a través de formularios HTML. Es importante especificar un nombre a cada elemento que contendrá datos como una caja de texto, un combo, campo de contraseña, etc.

Ejemplo:

```
<html>
<head>
	<title>Formulario B&aacute;sico</title>
</head>

<body>
	<form action=procesar.php' method=’POST’>
		Usuario <input type='text' name='usuario'><br>
		Password <input type='password' name='password'><br>
		<input type='submit'>
	</form>
</body>
</html>
```

2.5.12.2 Script que recibe los Datos

Una vez que se ha creado el formulario que recopilara la información que el usuario ingreso ahora será necesario crear el script de PHP que recibirá y procesara dichas variables.

El script al que se le enviara la información se llamara ***procesar.php*** que es el que se especifico en el action de la etiqueta form:

```
<?
	if ( $_POST) {
		echo "Hola <b>" . $_POST['usuario'] . "</b><br>";
		echo "Tu contraseña es: <b>" . $_POST['password'] . "</b><br>";
	}else {
		echo "No se recibió información.";
	}
?>
```

Antes de evaluar variables o realizar algún otro proceso, es conveniente realizar una validación para tener la certeza de que el script realmente esta recibiendo alguna variable

```
if ( $_POST) {
.
.
} else {
		.
		.
}
```

Al enviar datos del formulario a script de PHP por el método POST, automáticamente se declarara un arreglo asociativo llamado ***$_POST***, este arreglo contendrá los pares clave y valor que corresponden a todas las variables del formulario HTML.
La instrucción if evalúa si la variable $_POST esta declarada, si es así, ejecutara el bloque de sentencias para el procesamiento de los valores que tiene $_POST, en caso contrario, el script entrara al bloque else y mostrara un mensaje.

Para obtener el valor de las variables habrá que hacer referencia a ellas en el arreglo $_POST donde el subíndice será el nombre del elemento en el formulario HTML que se especifico con el atributo *name*.

```
echo "Hola <b>" . $_POST['usuario'] . "</b><br>";
echo "Tu contraseña es: <b>" . $_POST['password'] . "</b><br>";
```

Es conveniente usar etiquetas HTML para dar formato a los mensajes impresos por el script, finalmente, estas líneas serán interpretadas por el navegador.

Si por alguna razón se necesita enviar la información al servidor por el método GET, el servidor crearía un arreglo asociativo llamado ***$_GET*** en vez de $_POST. La manera en como se validara este arreglo y como se obtendrán los valores que contiene es la misma que se utilizo para trabajar con $_POST.

2.5.13 Cookies

Una *cookie* es un fragmento de información que se almacena en el disco duro del visitante de una página Web a través de su navegador, a petición del servidor de la página. Esta información puede ser luego recuperada por el servidor en visitas posteriores. Al ser el protocolo HTTP incapaz de mantener información por sí mismo, para que se pueda conservar información entre una página vista y otra (como nombre de usuario, preferencias de colores, etc.), ésta debe ser almacenada, ya sea en la URL de la página, en el propio servidor, o en una cookie en la computadora del visitante.

De esta forma, los usos más frecuentes de las cookies son:

- Llevar el control de usuarios: cuando un usuario introduce su nombre de usuario y contraseña, se almacena una cookie para que no tenga que estar introduciéndolas para cada página del servidor. Sin embargo una cookie no identifica a una persona, sino a una combinación de computadora y navegador.
- Ofrecer opciones de diseño (colores, fondos, etc.) o de contenidos personalizados al visitante.
- Conseguir información sobre los hábitos de navegación del usuario. Esto puede causar problemas de privacidad y es una de las razones por la que las cookies tienen sus restricciones.

PHP permite definir cookies a través del arreglo asociativo $_COOKIE[ ] que funciona casi de la misma manera en que los arreglos asociativos $_GET[ ] y $_POST[ ] lo hacen. La información que manejan las cookies de PHP es almacenada en pequeños archivos de texto no mayores a 4 kb de memoria.

2.5.13.1 Creación de las Cookies

Para la creación de cookies en PHP se emplea la función setcookie(). Esta función define una cookie para ser enviada junto con el resto de las cabeceras HTTP. Como otras cabeceras, las cookies deben ser enviadas antes de cualquier salida desde el script (esta es una restricción de protocolo). Esto requiere que se coloquen las llamadas a esta función antes de cualquier salida, incluyendo las etiquetas <html> y <head> así como cualquier espacio en blanco. Si existe salida antes de llamar esta

función, setcookie() fallará y devolverá FALSE. Si setcookie() se ejecuta con éxito, devolverá TRUE. Esto no indica si el usuario aceptó la cookie.

La sintaxis de setcookie es:

setcookie(string nombre [,string valor] [,int caducidad] [,string ruta] [,string dominio] [,int seguro]);

donde:

Parámetro	Descripción
nombre	El nombre de la cookie.
valor	El valor de la cookie. Este valor es almacenado en el equipo del cliente.
caducidad	La hora en la que expira la cookie. Comunmente, este valor es definido con la función time() más el número de segundos antes de que usted quiera que expire.
ruta	La ruta en el servidor en la que estará disponible la cookie.
dominio	El dominio en el que la cookie está disponible.
seguro	Indica que la cookie debería ser transmitida únicamente sobre una conexión HTTPS segura. Cuando su valor es TRUE, la cookie será definida únicamente si existe una conexión segura. El valor predeterminado es FALSE.

2.5.13.2 Eliminación de Cookies

Para eliminar una cookie también se usara la función setcookie() pasándole como único argumento el nombre de la cookie, de esta forma, se indica al interprete de PHP que elimine la cookie con dicho nombre. Ejemplo:

setcookie(“accesos”);

2.5.14 Sesiones

Las sesiones consisten en una forma de conservar ciertos datos a lo largo de los subsiguientes accesos, lo cual permite construir aplicaciones más personalizadas.

Una sesión es el 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. A cada usuario que accede al sitio Web, PHP le asigna un identificador único llamado *session_id*. Este soporte permite registrar un número arbitrario de variables que se conservarán a lo largo de las siguientes peticiones.

PHP comprobara cuando se le indique de forma explicita si se le ha enviado un id de sesión especifico a la petición del usuario, si estos corresponden entonces se recuperan los valores almacenados previamente en la sesión.

Cuando el interprete de PHP inicia una sesión, éste crea un arreglo asociativo llamado ***$_SESSION[ ]***. Este arreglo almacenara todas las variables de sesión tal y como lo haría $_POST, $_GET y $_COOKIE.

Una buena práctica es eliminar las variables de sesión y posteriormente la sesión misma cuando el script termine de utilizarlas.

2.5.14.1 Funciones de Manejo de Sesión

session_start()	Crea una sesión (o la continúa basándose en el session_id pasado por GET, POST o mediante una cookie).
session_id([string id])	Devuelve el session_id de la sesión actual. Si se especifica un *id*, reemplazará el session_id actual.
session_unset()	Elimina y libera el espacio ocupado por todas las variables de la sesión actual registradas.
session_destroy()	Destruye todos los datos asociados con la sesión actual. No destruye ninguna de las variables globales asociadas a la sesión ni la cookie. Esta función devuelve TRUE si se ha destruido la sesión correctamente y FALSE si ha habido algún problema al intentarlo.

Ejemplo:

```
<?
        session_start();

        $_SESSION[‘mensaje’] = ‘este mensaje persistira’;

        if ( $_SESSION ) {
                echo “Lo que hay en la variable de sesion “;
                echo $_SESSION[‘mensaje’];
        }
?>
```

2.6 Interacción de WWW con bases de Datos

2.6.1 Introducción

Una base de datos es un gran conjunto no redundante de datos estructurados y organizados independientemente de su uso y su implementación, accesibles en tiempo real y compatible con usuarios concurrentes con necesidad de información diferente.

Para trasladar un problema de la vida real que requiera el uso de una base de datos se recurre a un modelo. Un modelo es básicamente una "descripción" de algo conocido como contenedor de datos (algo en donde se guarda la información), así como de los métodos para almacenar y recuperar información de esos contenedores. Los modelos de datos no son cosas físicas: son abstracciones que permiten la implementación de un sistema eficiente de base de datos; por lo general se refieren a algoritmos, y conceptos matemáticos.

Algunos modelos con frecuencia utilizados en las bases de datos:

- **Bases de datos jerárquicas**. Éstas son bases de datos que, como su nombre indica, almacenan su información en una estructura jerárquica. En este modelo los datos se organizan en una forma similar a un árbol (visto al revés), en donde un nodo padre de información puede tener varios hijos. El nodo que no tiene padres es llamado raíz, y a los nodos que no tienen hijos se los conoce como hojas.
- **Bases de datos de red**. Éste es un modelo ligeramente distinto del jerárquico; su diferencia fundamental es la modificación del concepto de nodo: se permite que un mismo nodo tenga varios padres (posibilidad no permitida en el modelo jerárquico). Fue una gran mejora con respecto al modelo jerárquico, ya que ofrecía una solución eficiente al problema de redundancia de datos; pero, aun así, la dificultad que significa administrar la información en una base de datos de red ha significado que sea un modelo utilizado en su mayoría por programadores más que por usuarios finales.
- **Bases de Datos Relacionales**. Éste es el modelo más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente. Tras ser postulados sus fundamentos en 1970 por Edgar Frank Codd. Estas relaciones podrían considerarse en forma lógica como conjuntos de datos llamados "tuplas". Esto es pensando en cada relación como si fuese una tabla que está compuesta por registros (las filas de una tabla), que representarían las tuplas, y campos (las columnas de una tabla).

  En este modelo, el lugar y la forma en que se almacenen los datos no tienen relevancia (a diferencia de otros modelos como el jerárquico y el de red). Esto tiene la considerable ventaja de que es más fácil de entender y de utilizar para un usuario esporádico de la base de datos. La información puede ser recuperada o almacenada mediante "consultas" que ofrecen una amplia flexibilidad y poder para administrar la información.
- **Bases de Datos orientadas a Objetos.** Este modelo, bastante reciente, trata de almacenar en la base de datos los objetos completos (estado y comportamiento). Este tipo de bases de datos incluyen los conceptos del paradigma de objetos: encapsulación, herencia y polimorfismo.

2.6.2 Sistemas Gestores de Bases de Datos

Las bases de datos se valen de otros programas que son los intermediarios entre la base de datos y el usuario, estos son conocidos como Sistemas Gestores de Bases de Datos. Los Sistemas Gestores de Bases de Datos (SGBD o DBMG) son un tipo de software muy específico, dedicado a servir de interfaz entre las bases de datos y las aplicaciones que la utilizan. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta.

Entre los propósitos más generales de un SGDB están:

- ***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 por lo menos evitarla al máximo.
- ***Consistencia***. En aquellos casos en los que no se ha logrado esta redundancia nula, será necesario 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. Los SGBD deben garantizar que esta información se encuentra asegurada frente a usuarios malintencionados, que intenten leer o modificar información privilegiada.
- ***Integridad***. Se trata de adoptar las medidas necesarias para garantizar la validez de los datos almacenados, es decir, proteger los datos de cualquier circunstancia capaz de corromper la información almacenada.
- ***Control de la concurrencia***. En la mayoría de entornos, lo más habitual es que sean muchas las personas que acceden a una base de datos para realizar ya sea una consulta o modificación a ellos, de igual manera, dichos accesos pueden ser realizados de forma simultánea. Es el SGDB quien debe controlar este acceso concurrente a la información, que podría derivar en inconsistencias.

Entre los SGDB más populares están Postgres, SQL Server, Sybase, Oracle, MySQL (que es el que se empleara en este módulo), entre otros.

2.6.3 Llave Primaria

Conocida también como PK por sus siglas en inglés *Primary Key.* Cada instancia de una entidad debe ser unívocamente identificable, de manera tal que cada registro de la entidad debe estar separado y ser unívocamente identificable del resto de los registros de esa misma entidad; y quien permite esta identificación es la llave primaria. La llave primaria puede ser un atributo o una combinación de atributos.

En consecuencia en cada tabla sólo podrá existir un único registro que posea un valor determinado para su llave primaria. En otras palabras no puede existir en una tabla un registro que cuente con el mismo valor de otro registro en el campo de la llave primaria; la llave primaria no puede tener valores repetidos para distintos registros.
La llave primaria debe permitirle al SGBD, correctamente proyectado, generar un error si un usuario intenta incluir un nuevo registro cuya llave primaria coincida con la de otro registro ya existente en la tabla.

2.6.4 Llave Foránea

Conocida también como FK por sus siglas en inglés *Foreign Key*. La llave foránea es la que se encarga de relacionar las entidades y está representada por la llave primaria de otra entidad. Este tipo de llave proporciona la integridad referencial. No es requerido que todas las tablas cuenten con una llave foránea.

2.6.5 Restricciones de datos en MySQL

Las restricciones se imponen para asegurar que los datos cumplen con una serie de condiciones predefinidas para cada tabla. Estas restricciones ayudan a conseguir la integridad referencial: todas las referencias dentro de una base de datos son válidas si todas las restricciones se han cumplido.

Las restricciones se van a definir acompañadas por un nombre, lo que permitirá activarlas o desactivarlas según sea el caso; o también mezcladas en la definiciones de las columnas de la tabla. A continuación se muestran las restricciones que MySQL permite.

2.6.5.1 NOT NULL

Establece la obligatoriedad de que esta columna tenga un valor no nulo. Se debe especificar junto a la columna a la que afecta. Los valores nulos no ocupan espacio, y son distintos a 0 y al espacio en blanco. Hay que tener cuidado con los valores nulos en las operaciones, ya que 1 * NULL es igual a NULL.

2.6.5.2 UNIQUE

Evita valores repetidos en una columna, admitiendo valores nulos.

2.6.5.3 DEFAULT

Establece un valor por defecto para esa columna, si no se le asigna ninguno.

2.6.5.4 CHECK

Comprueba que se cumpla una condición determinada al rellenar esa columna. Esta condición sólo debe estar construida con columnas de esta misma tabla.

2.6.5.5 PRIMARY KEY

Establece el conjunto de columnas que forman la clave primaria de esa tabla. Se comporta como única y obligatoria sin necesidad de especificarlo. Sólo puede existir una clave primaria por tabla. Puede ser referenciada como clave forárena por otras tablas.

2.6.5.6 FOREIGN KEY

Establece que el contenido de esta columna será uno de los valores contenidos en una columna de otra tabla maestra. Esta columna marcada como clave foránea puede ser NULL. No hay límite en el número de claves foráneas. La clave foránea puede ser otra columna de la misma tabla. Se puede forzar que cuando una fila de la tabla maestra sea borrada, todas las filas de la tabla detalle cuya clave foránea coincida con la clave borrada se borren también. Esto se consigue añadiendo la coletilla ON DELETE CASCADE en la definición de la clave ajena.

2.6.6 Relaciones

Las relaciones ayudan a dar fuerza a las reglas y afirmaciones en un modelo de datos. Las relaciones determinan cómo los datos están asociados entre dos entidades (tablas). Las relaciones se llevan a cabo a través de llaves foráneas. Las propiedades de estas llaves foráneas dictan cómo se da fuerza a la integridad referencial entre las tablas.

Las relaciones pueden ser:

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

2.6.7 Tipos de Datos

2.6.7.1 Caracter

char(n)	n es el número de caracteres que la variable tendrá como longitud fija
varchar(n)	N es el numero de caracteres que la variable tendrá como máximo, su longitud se ajustara de 0 hasta n
text	Campo de texto muy grande. Las ordenaciones y comparaciones sobre datos de tipo text no son sensibles a mayúsculas y minúsculas.
blob	Campo de texto muy grande. Las ordenaciones y comparaciones sobre datos de tipo blob son sensibles a mayúsculas y minúsculas.

2.6.7.2 Numéricos

tinyint [sin signo]	1 byte
smallint [sin signo]	2 bytes
mediumint [sin signo]	3 bytes
int [sin signo]	4 bytes
bigint [sin signo]	8 bytes
float (e,d)	Números decimales donde e especifica el total de cifras para números enteros mientras d especifica el total de cifras para los decimales
double (e,d)	Igual que float pero de doble precisión 8 bytes
decimal (e,d)	Igual que float pero decimal se usa para almacenar cantidades en ormato monetario.

2.6.8 SQL

Lenguaje de Consulta Estructurado (de sus siglas en inglés *Structured Query Language*). Es un lenguaje de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones sobre las mismas con el fin de recuperar o manipular información de interés de una base de datos, de una forma sencilla.

SQL proporciona una gran funcionalidad más allá de la simple consulta (o recuperación) de datos. Asume el papel de Lenguaje de Definición de Datos (DDL) y el Lenguaje de Manipulación de Datos (DML), entre otros.

2.6.8.1 Lenguaje de Manipulación de Datos

El lenguaje de Manipulación de datos, en ingles *Data Manipulation Language* (DML), es el que se encarga de la modificación de los datos dentro de la base de datos. Mediante este grupo de comandos, es posible consultar y modificar todos los datos de la base de datos. Es el principal componente del SQL. Existen cuatro operaciones básicas: INSERT, UPDATE, DELETE y SELECT.

Cabe mencionar que los SGDB no son sensibles a mayúsculas y minúsculas en cuestión de las sentencias de SQL pero algunos si lo son a los nombres de las tablas.

A continuación se muestran las sintaxis y ejemplos de cada una de las sentencias DML:

2.6.8.1.1 INSERT

Este comando SQL inserta registros en una tabla específica. Se pueden insertar valores específicos o valores provenientes de otra tabla

Sintaxis:

INSERT INTO *nombre_tabla* (*campo1, campo2, campoN*) VALUES (*valor1, valor2, valor3*)

Ejemplo:

INSERT INTO alumno (noCuenta, nombre, apellidoPaterno) VALUES (1234, 'Gabriel', 'González');

Las variables que corresponden a cadenas son encerrados por el carácter de comilla simple (').

2.6.8.1.2 UPDATE

Este comando SQL modifica los valores de los campos de registros ya existentes en una tabla especifica.

Sintaxis:

UPDATE *nombre_tabla* SET *campo1 = valor1*, *capo2 = valor2*, *campoN = valorN* WHERE *condiciones*

Ejemplo:

UPDATE calificaciones SET calificación = 10 WHERE noCuenta = 1234;

2.6.8.1.3 DELETE

Este comando SQL elimina registros de una tabla especifica.

Sintaxis:

DELETE FROM *nombre_tabla* WHERE *condiciones*

Ejemplo:

DELETE FROM alumno WHERE noCuenta = 1234

2.6.8.1.4 SELECT

Este comando SQL permite devolver información de una o más tablas de la base de datos. Este comando es utilizado junto con la cláusula WHERE.

Este comando permite el uso de el comodín asterisco (*) para seleccionar todas las columnas de la tabla especificada por la cláusula FROM.

Sintaxis:

SELECT *campo1, campo2, campoN* FROM *nombre_tabla*

Ejemplo:

SELECT nombre, apellidoPaterno FROM alumno

SELECT * FROM alumno WHERE noCuenta = 1234

2.6.8.1.5 Cláusula WHERE

La cláusula WHERE le dice a SQL cual es el criterio que determina que fila o filas hay que obtener. La estructura de WHERE es:

WHERE *columna operador valor*

SQL proporciona un amplio rango de operadores de comparación como se muestran en la siguiente tabla:

Operador	Significado
=	Igual
>	Mayor que
<	Menor que
>=	Mayor o igual que
<=	Menor o igual que
<>	Distinto que
LIKE	Que cumpla con una subcadena

2.6.8.2 Lenguaje de Definición de Datos

El lenguaje de Definición de datos, en ingles *Data Definition Language* (DDL), es el que se encarga de la modificación de la estructura de los objetos de la base de datos. Existen tres operaciones básicas: CREATE, ALTER y DROP.

2.6.8.2.1 CREATE

Este comando crea un objeto dentro de la base de datos. Puede ser una tabla o cualquier otro objeto que el motor de la base de datos soporte.

Sintaxis:

CREATE *tipo_objeto nombre <definicion>*

En caso de usar la sentencia CREATE para crear una tabla, la definición de este objeto será la de especificar los nombres de las variables, el tipo de dato que almacenara y opcionalmente sus restricciones.

Ejemplo:

```
CREATE TABLE alumno(
        noCuenta varchar(15) NOT NULL PRIMARY KEY
        nombre varchar(20) NOT NULL
        apellidoPaterno(20) NOT NULL
)
```

2.6.8.2.2 ALTER

Este comando permite modificar la estructura de un objeto. Se pueden agregar/quitar campos a una tabla, modificar el tipo de un campo, agregar/quitar índices a una tabla, etc.

ALTER TABLE *nombre_tabla*
        ALTER COLUMN <defínción_de_la_columna> |
        ADD <definición_de_la_coumna> |
        DROP COLUMN *nombre_de_la_columna* |
        ADD CONSTRAINT <restricción_tabla>

Ejemplo:

ALTER TABLE alumno ADD COLUMN apellidoMaterno varchar(20)

2.6.8.2.3 DROP

Este comando elimina un objeto de la base de datos. Puede ser una tabla o cualquier otro objeto que el motor de la base de datos soporte. Se puede combinar con la sentencia ALTER.

Sintaxis:

DROP TABLE *nombre_tabla*

Ejemplo:

DROP TABLE alumno

2.7 Introducción a la Seguridad en Cómputo

La definición de la seguridad informática es pues como lograr adquirir, almacenar, procesar y transmitir información en un entorno de este tipo preservando lo más que se pueda los servicios de seguridad:

- ***Confidencialidad***. Sólo el propietario del secreto es capaz de descifrar la información.
- ***Integridad***. Se asegura que la información se mantenga inalterada desde su creación.
- ***Autenticación y Autorización (control de Acceso).*** Se asegura que la información estará disponible sólo para determinados usuarios con la posibilidad de manejar diferentes niveles de acceso para cada uno de ellos.

Se puede realizar mediante alguno de los siguientes mecanismos:
  - ✓ "algo que se sabe" (contraseñas)
  - ✓ "algo que se tiene" (tarjetas inteligentes)
  - ✓ "algo que se es" (sistemas biométricos)
- ***No Repudio.*** Se asegura que el emisor de la información no puede negar haberla enviado.
- ***Disponibilidad*****.** Asegurar que los usuarios legítimos puedan usar la información cuando lo requieran.

En general, la seguridad de un sistema tiene que ver con cualquier técnica, procedimiento o medida que reduce la vulnerabilidad del mismo. La seguridad tiene como objetivos principales lograr la confidencialidad, integridad, autenticidad de la información y garantizar la disponibilidad de la misma y de los recursos de cómputo.

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

2.7.1 Criptografía Simétrica

Requiere que el emisor y el receptor compartan una clave secreta "Llave", la cual es utilizada para cifrar el mensaje cuando se envía y descifrar el mismo mensaje al recibirlo. El gran inconveniente se presenta en el proceso de intercambiar de forma segura la Llave.

Dado que toda la seguridad está en la clave, es importante que sea muy difícil adivinar el tipo de clave. Esto quiere decir que el abanico de claves posibles, o sea, el espacio de posibilidades de claves, debe ser amplio.

Las computadoras actuales pueden adivinar claves con extrema rapidez, y ésta es la razón por la cual el tamaño de la clave es importante en los sistemas modernos. Por ejemplo, el algoritmo de cifrado DES usa una clave de 56 bits, lo que significa que hay 2 elevado a 56 claves posibles. Esto representa un número muy alto de claves, pero una computadora de uso general puede comprobar todo el espacio posible de claves en cuestión de días. Una máquina especializada lo puede hacer en horas.

Por otra parte, algoritmos de cifrado de diseño más reciente como 3DES, Blowfish e IDEA usan todos claves de 128 bits, lo que significa que existen 2 elevado a 128 claves posibles. Esto representa muchísimas más claves haciendo que el tiempo y costo para obtener la clave sea demasiado elevado.

En este esquema de seguridad es importante que el emisor disponga de un canal seguro para realizar la entrega de la llave al inicio de la transmisión.

Como ejemplos de algoritmos de criptografía simétrica están: DES, 3DES, Blowfish, IDEA

2.7.2 Criptografía Asimétrica

La distribución de llaves en la criptografía asimétrica O de llave pública es tal que cada participante debe difundir su llave pública a todos sus corresponsales, quienes ya conocen el algoritmo que se está empleando. Adicionalmente se puede establecer un

directorio (servidor) de llaves públicas para que una persona que desee enviar información privada a alguien al que no conoce lo pueda hacer.

En el esquema de llave pública todos los usuarios tienen una llave pública y una privada. Si alguien quiere enviarte un mensaje, obtiene una copia de tu llave pública con la cual cifra el mensaje que sólo podrá descifrarse con tu llave secreta. Los mensajes cifrados con la llave pública no se pueden descifrar con la misma llave pública.

Este esquema resuelve varios de los problemas que tiene el cifrado de llave secreta, ya que todas las comunicaciones a una entidad usan la llave pública de la entidad, reduciendo así el número de llaves que emplea el sistema. Las entidades no se preocupan por mantener secreta su llave, al contrario la hacen pública. Se reduce sustancialmente el riesgo de que una llave privada caiga en manos de una persona malintencionada, en virtud de que cada usuario se preocupa por mantener secreta su llave descifradora que es su llave privada. El cifrado de llave pública está basado en funciones matemáticas cuya complejidad hace poco posible que con un tiempo y potencia de cómputo razonable, conociendo sólo el mensaje cifrado y la llave pública pueda deducirse la llave privada y con ella obtener el mensaje original.

Como ejemplo de algoritmos de criptografía asimétrica están: Rivest Shamir Adleman (RSA) y Diffie – Hellman (D-H).

2.7.3 Funciones Hash

Las funciones de resumen o funciones Hash son unidireccionales y operan sobre un mensaje de longitud arbitraria entregando un valor de resumen o valor Hash de longitud fija.

El objetivo de las funciones Hash es producir un identificador único de cualquier documento digital en forma eficiente y segura. Es lo equivalente a producir algo parecido a una huella digital.

Es decir, todo documento procesado mediante una función Hash, debe producir sólo un resultado o resumen de la aplicación de esta función sobre el documento. Entonces dicho resumen se puede asociar indivisiblemente al mensaje original, para certificar que dicho mensaje es auténtico, si un sólo caracter del mensaje original cambia el Hash será totalmente distinto.

Ejemplos de algoritmos de Hash: SHA1, SHA2, MD5

2.7.4 Firma Digital

La firma digital de un documento es el resultado de aplicar una función Hash a su contenido. Esta función asocia un valor dentro de un conjunto finito a su entrada. Cuando la entrada es un documento, el resultado de la función es un número que identifica casi unívocamente al texto.

Una vez que se tiene este Hash, el firmante cifra el Hash (también podría firmar el documento sin necesidad de aplicarle una función de resumen) con su clave privada y cualquiera que quiera comprobar la firma y ver el documento, no tiene más que usar la clave pública del firmante para descifrarla.

Si el documento se modifica, la comprobación de la firma fallará.

Con la firma electrónica se obtiene:

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

2.7.5 Certificados Digitales

Un Certificado Digital es un documento digital mediante el cual un tercero confiable (una autoridad de certificadora) garantiza la autenticidad entre la identidad de un sujeto o entidad y su clave pública.

Existen varios formatos de certificado digital, los más comúnmente empleados se rigen por el estándar UIT-T X.509v3. El certificado contiene usualmente el nombre de la entidad certificada, un número de serie único que idéntica al certificado, fecha de expiración, una copia de la clave pública del titular del certificado (utilizada para la verificación de su firma digital), y la firma digital de la autoridad emisora del certificado de forma que el receptor pueda verificar la autenticidad de esta última.

En un medio de comunicación masiva inseguro como lo es Internet, el mecanismo para poder tener la certeza de la identidad de la persona con quien se está intercambiando información son los certificados digitales.

2.7.6 HTTPS

HTTPS es la versión segura del protocolo HTTP. El sistema HTTPS utiliza un cifrado basado en las Secure Socket Layers (SSL) para crear un canal cifrado cuyo nivel de cifrado depende del servidor remoto y del navegador utilizado por el cliente.

Este protocolo es más apropiado para el intercambio de información sensible. Cabe mencionar que el uso del protocolo HTTPS no impide que se pueda utilizar HTTP. Es aquí, cuando el navegador advertirá sobre la carga de elementos no seguros (HTTP), estando conectados a un entorno seguro (HTTPS).

2.7.7 PKI

La infraestructura de clave pública (PKI por sus siglas en inglés *Public Key Infrastructure*) es una combinación de hardware y software, políticas y procedimientos que permiten asegurar la identidad de los participantes en un intercambio de datos usando criptografía pública.

El termino PKI se utiliza para referirse tanto a la autoridad certificadora y al resto de componentes.

Una PKI permite a los usuarios autenticarse frente a otros usuarios y usar la información de los certificados digitales (por ejemplo, las claves públicas de otros usuarios) para cifrar y descifrar mensajes. En general, una PKI consiste en un software para los clientes, un software de servidor (como una autoridad certificadora), hardware

(por ejemplo, tarjetas inteligentes o smart cards) y unos procedimientos operacionales. Un usuario puede firmar digitalmente mensajes usando su clave privada, y otro usuario puede validar dicha firma usando la clave pública del usuario contenida en el certificado que ha sido emitido por una autoridad certificadora. Esto permite a dos o más entidades establecer una comunicación que garantiza la confidencialidad y la integridad del mensaje y la autenticación de los usuarios sin tener que intercambiar previamente ninguna información secreta.

Los componentes más habituales de una infraestructura de clave pública son:

- Autoridad Certificadora (CA de sus siglas en inglés *Certificate Authority*): es la encargada de emitir y revocar certificados. Es la entidad de confianza que da legitimidad a la relación de una clave pública con la identidad de un usuario o servicio.
- Autoridad Registradora (RA de sus siglas en inglés *Registration Authority*): es la responsable de verificar el enlace entre los certificados (concretamente, entre la clave pública del certificado) y la identidad de sus titulares.
- Los repositorios: son las estructuras encargadas de almacenar la información relativa a la PKI. Los dos repositorios más importantes son el repositorio de certificados y el repositorio de listas de revocación de certificados. En una lista de revocación de certificados (o, en inglés, CRL, *Certificate Revocation List*) se incluyen todos aquellos certificados que algún motivo han dejado de ser válidos antes de la fecha establecido dentro del mismo certificado.
- La autoridad de validación (o, en inglés, VA, *Validation Authority*): es la encargada de comprobar la validez de los certificados digitales.
- La autoridad de sellado de tiempo (o, en inglés, TSA, *TimeStamp Authority*): es la encargada de firmar documentos con la finalidad de probar que existían antes de un determinado instante de tiempo.
- Los usuarios y entidades finales son aquellos que poseen un par de claves (pública y privada) y un certificado asociada a su clave pública. Utilizan un conjunto de aplicaciones que hacen uso de la tecnología PKI (para validar firmar digitales, cifrar documentos para otros usuarios, etc.)

2.7.8 Ataques

Un ataque es una acción o conjunto de acciones que tienen por objetivo el que cualquier parte de un sistema de información automatizado, deje de funcionar de acuerdo con su propósito definido. Esto incluye cualquier acción que causa la destrucción, modificación o retraso del servicio.

2.7.8.1 Ataques Pasivos

En los ataques pasivos el atacante no altera los datos ni la comunicación, sino que únicamente la escucha o monitorea, para obtener información que está siendo transmitida.

Sus objetivos son la intercepción de datos y el análisis del tráfico de red, 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 que están siendo monitoreados.
- Control del volumen de tráfico intercambiado entre las entidades monitorizadas, obteniendo así información acerca de actividad o inactividad inusuales.

- Control de las horas habituales de intercambio de datos entre las entidades de la comunicación, para extraer información acerca de los períodos de actividad.

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

- Recopilación de la información de las bases de datos.
- Análisis de trazados de ruta
- Sniffers
- Analizadores de Tráfico

2.7.8.2 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, como por ejemplo ingresar dinero repetidas veces en una cuenta dada.
- Modificación de mensajes: una porción del mensaje legítimo es alterada, o los mensajes son retardados o reordenados, para producir un efecto no autorizado.
- 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 falsos.

Entre los ataques activos más comunes están:

- Virus
- Caballos de Troya
- Exploits (backdoors, bombas lógicas)
- Fuerza bruta
- Spoofing
- Negación de Servicion (DoS)

3. Informe del Proyecto de Sistema de Control de Certificados Digitales

3.1 Introducción

La necesidad del intercambio de información a través de Internet de una forma más segura es una necesidad para la agilización de algunos trámites que pueden ser realizados por este medio reduciendo los tiempo, gasto en el almacenamiento de comprobantes físicos y el desplazo de la gente involucrada en dicho proceso. Estos procesos adquieren un marco más seguro cuando se implemente la firma electrónica, pero esta a su vez, requiere de un certificado digital para poder garantizar que una persona es realmente quien dice ser en un medio electrónico y que esta no podrá negar la autoría del mensaje de datos emitido.

3.2 Objetivo

Desarrollar una aplicación que permita la administración de los certificados digitales emitidos por la Autoridad Certificadora de la UNAM, para poder llevar un control de los lugares donde se emiten, fecha y hora, solicitud de revocación e impresión o reimpresión de la carta de responsabilidad que se entregara al usuario que adquiere el certificado, resguardando todas estas aplicaciones exclusivamente a usuarios autorizados.

3.3 Alcance

La aplicación se compondrá de los siguientes módulos:

- Módulo de autenticación y selección de opciones.
- Generación de la carta compromiso.
- Solicitud de la revocación del certificado digital
- Consulta de la carta compromiso.

3.4 Módulos

3.4.1 Módulo de Autenticación

La autenticación del operador del sistema se hará a través de un usuario y contraseña figura 22. Las contraseñas se almacenan cifradas en una base de datos lo que impide que se puedan reconocer los caracteres que la conforman al hacer una consulta. En base a los privilegios de la cuenta usada, se mostraran las opciones con las que el usuario podrá interactuar, figura 23.

Figura 22 – Pantalla de Autenticación

Figura 23 – Pantalla de Selección de Opciones

3.4.2 Módulo de Generación de la Carta Compromiso

Una vez que el operador del sistema se autentique con la cuenta correspondiente, podrá acceder al módulo para generar una carta compromiso por cada uno de los certificados emitidos por la Autoridad Certificadora de la UNAM. El operador sólo vera los nombres de los usuarios que corresponden a su facultad o escuela a los que aún no se les emite una carta, figura 24.

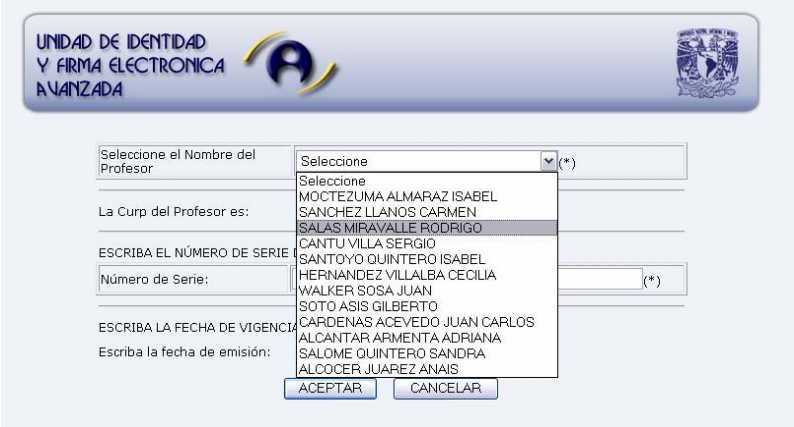


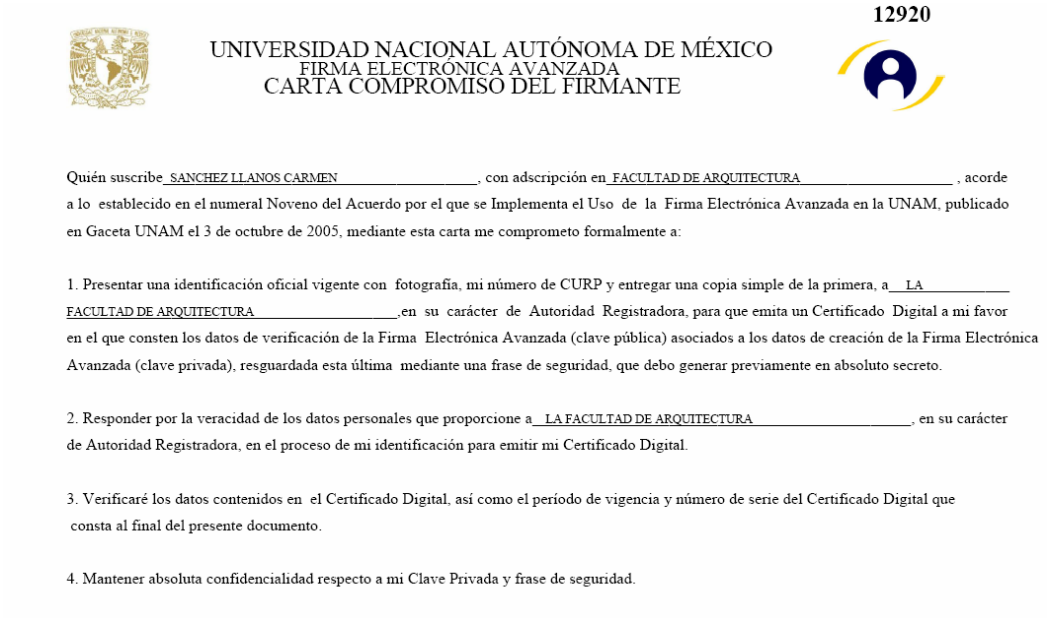
Figura 24 – Generación de la Carta compromiso

Al seleccionar el nombre de algún usuario, sus datos se mostrarán de forma automática. Es necesario ingresar el número de serie del certificado generado el cual podrá ser consultado en el navegador, figura 25

Figura 25 – Recopilación de datos del usuario y certificado

Al hacer clic en el botón Aceptar, el sistema mostrara una ventana de confirmación, figura 26 y al aceptar se generara la carta compromiso figura 27.

Figura 26 – Confirmación de datos



12920

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
FIRMA ELECTRÓNICA AVANZADA
CARTA COMPROMISO DEL FIRMANTE

Quién suscribe SANCHEZ LLANOS CARMEN, con adscripción en FACULTAD DE ARQUITECTURA, acorde a lo establecido en el numeral Noveno del Acuerdo por el que se Implementa el Uso de la Firma Electrónica Avanzada en la UNAM, publicado en Gaceta UNAM el 3 de octubre de 2005, mediante esta carta me comprometo formalmente a:

1. Presentar una identificación oficial vigente con fotografía, mi número de CURP y entregar una copia simple de la primera, a LA FACULTAD DE ARQUITECTURA, en su carácter de Autoridad Registradora, para que emita un Certificado Digital a mi favor en el que consten los datos de verificación de la Firma Electrónica Avanzada (clave pública) asociados a los datos de creación de la Firma Electrónica Avanzada (clave privada), resguardada esta última mediante una frase de seguridad, que debo generar previamente en absoluto secreto.

2. Responder por la veracidad de los datos personales que proporcione a LA FACULTAD DE ARQUITECTURA, en su carácter de Autoridad Registradora, en el proceso de mi identificación para emitir mi Certificado Digital.

3. Verificaré los datos contenidos en el Certificado Digital, así como el período de vigencia y número de serie del Certificado Digital que consta al final del presente documento.

4. Mantener absoluta confidencialidad respecto a mi Clave Privada y frase de seguridad.

Figura 27 – Carta Compromiso

3.4.3 Módulo de Solicitud de Revocación del Certificado Digital

Este módulo esta restringido a un operador con más privilegios ya que a través de esta opción es como se solicita a la Unidad de Identidad y Firma Electrónica Avanzada la baja (revocación) de cualquier certificado digital emitido en determinada escuela o facultad por cuestiones de olvido de contraseña, perdida, seguridad comprometida, etc. La solicitud envía los datos del certificado en cuestión, la fecha y hora de la solicitud y procedencia de la misma.

Este proceso impedirá que se realicen nuevas transacciones con este certificado digital desde el momento en que la CA de la UNAM lo agregue en su lista de certificados revocados.

Para iniciar este proceso, es necesario seleccionar el nombre del profesor, el motivo por el cual se esta solicitando la revocación y el número de serie del certificado, figura 28.

Figura 28 – Módulo de Revocación

Será necesario confirmar los datos como en el módulo anterior y al hacer clic en aceptar se mostrara la pantalla final del proceso.

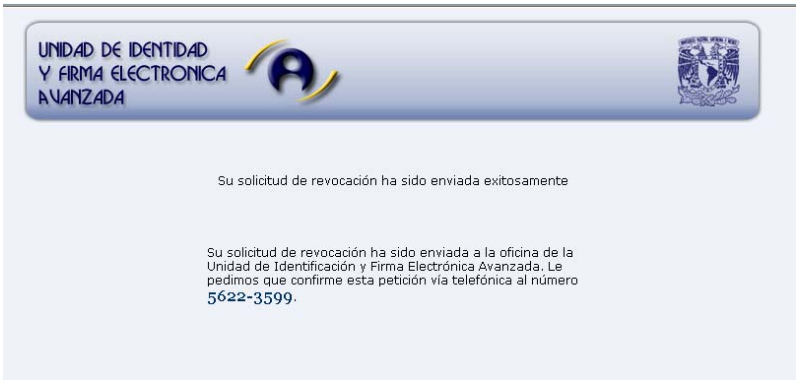


Figura 29 – Solicitud de revocación enviada

3.4.4 Modulo de Consulta de la Carta Compromiso

Este es el único módulo al que se puede acceder sin importar el tipo de cuenta con el que se haya ingresado al sistema. Esta opción permite consultar o reimprimir cualquier carta compromiso de un certificado activo que la escuela o facultad haya emitido.

El proceso de reimpresión de estas cartas es similar al proceso que se realiza cuando se imprime una carta por primera vez como se muestra en las figuras 25 y 26 a diferencia que ya no será necesario ingresar el número de serie del certificado ya que este ya se encuentra almacenado en la base de datos.

3.5 Marco Tecnológico

Se busco tener un entrono seguro y estable para montar el sistema. Por las características que se han mencionado anteriormente se eligió a Linux Slackware como el sistema operativo más adecuado.

Con respecto al servidor Web, se selecciono a Apache por ser un sistema robusto y altamente configurable que se apega por completo a las necesidades de la aplicación. Su amplia gama de directivas permiten asignar la cantidad de recursos necesarios para poder responder a todas las peticiones que los usuarios puedan llegar a hacer y sobre todo, que el desempeño de este servidor Web sólo se ve limitado por las capacidades de hardware en el que esta instalado y no en la propia aplicación.

Apache también permite el manejo de sitios virtuales lo cual resuelve el problema de contar con varias direcciones IP, en caso de poner más aplicaciones Web en línea es posible ocupar la misma dirección únicamente asignando diferentes puertos.

Además de ser un sistema libre, otra buena característica por la cual se opto por esta opción es que Apache despacha de forma más rápida las peticiones web que otros servidores como Tomcat o IIS.

Junto con Apache se empleo el lenguaje de programación PHP versión 5 por su integración transparente con el servidor Web y por ser una herramienta muy efectiva para la creación de páginas web dinámicas y su gran número de funciones para conexión a bases de datos, funciones de resumen, sesiones, etc.

Como almacén de datos se eligió a MySQL versión 5, esta nueva versión implementa funciones que optimizan sus procesos que lo dejaban en desventaja ante otros manejadores de bases de datos relacionales como Postgres. MySQL también incorpora sus propias versiones de resumen basadas en los algoritmos MD5 y SHA1, lo que permite una compatibilidad con PHP y ofrece un mecanismo para asegurar la información sensible como contraseñas almacenadas.

El conjunto de estas herramientas permiten diseñar e implementar una solución completa, segura y estable con software libre además de darnos la confianza de que todos estos componentes son enriquecidos día a día gracias a la colaboración de miles de usurarios alrededor del mundo que descubren posibles huecos de seguridad o nuevas formas de mejorar los funcionamientos ya existentes.

Conclusiones

En la actualidad, el uso de sistemas que automaticen el manejo de grandes cantidades de información se vuelve una necesidad para mantener el control de las operaciones que las empresas o negocios realizan, al igual que se debe garantizar la estabilidad y confiabilidad de sistemas que poco a poco se vuelven críticos.

Cada vez es más común oír sobre instituciones que emplean el sistema operativo Linux junto con software libre para la implementación de sus aplicaciones. El número de usuarios de Linux en estos días está aumentando considerablemente a comparación de un par de años, según Linux Counter Organization estima que actualmente existen 18 millones de usuarios en el mundo.

La principal ventaja de usar este sistema operativo es que es gratuito, puede ser encontrado y descargado casi desde cualquier sitio junto con lenguajes de programación, manejadores de bases de datos, etc. En primera instancia esto hace muy atractivo el uso de software libre, pero la principal desventaja que esto trae es que no hay soporte técnico para las paqueterías que son distribuidas de forma libre así como tampoco se cuenta con un respaldo ante algún inconveniente ocurrido durante su uso.

Linux en cualquiera de sus distribuciones brinda un entorno de desarrollo muy bueno en el que se mantiene el control y monitoreo más “limpio” de los procesos en ejecución en determinado momento con ayuda de comandos y utilerías instaladas por defecto o que pueden ir siendo adaptadas a una necesidad en particular.

Saber cual será la función del sistema operativo permitirá realizar la planeación más adecuada de la distribución de los recursos para las aplicaciones y usuarios. Es recomendable conocer las características de la distribución que será empleada para elegir la configuración más óptima que ayudara a aprovechar más la potencia de esta plataforma. Por ejemplo, si se particiona el disco duro y se montan los directorios en espacios distintos, la administración de la información será más sencilla y en caso de la reinstalación de alguna parte del sistema operativo ésta podrá hacerse de forma modular.

Una de las aplicaciones más comunes en Linux y sistemas Unix son las páginas dinámicas que corren sobre el servidor Web Apache, este servidor es muy popular entre las empresas que se dedican a la renta de dominios en Internet gracias a su versatilidad, robustez, velocidad para despachar peticiones de usuarios y la compatibilidad que tiene con lenguajes de programación como PHP, manejadores de bases de datos como MySQL y Postgres y otros módulos que se pueden integrar de forma dinámica al servidor aumentando su potencia y seguridad. Los manejadores de bases de datos como los mencionados anteriormente logran competir con costosos manejadores como Oracle en cuestión de eficiencia, estabilidad y solidez.

Muchas de las herramientas mas versátiles y potentes para la detección de personas no autorizadas en una red o sistema se encuentran disponibles para Linux, sistemas tales como *snort*, *auditors* y utilerías como *nmap* pueden ser de gran ayuda para realizar auditorias a los sistemas informáticos en busca de vulnerabilidades y corregirlas antes de que alguien extraño lo haga, pero de igual forma, estas herramientas pueden ser utilizadas para realizar grandes ataques.

El mundo de Linux es muy diverso, es posible realizar casi cualquier actividad en él, los proveedores de hardware prestan cada vez más atención a la compatibilidad de los

controladores de sus dispositivos para estas plataformas. Hay que tener en cuenta que el sistema operativo que será mejor es aquel que se apegue a las necesidades que se tendrán como usuario, por ejemplo, es más cómodo tener una computadora con sistema operativo Windows que se destinara al uso casero de procesamiento de textos, consulta de correo electrónico o quizá el entretenimiento al ver una película o pasar el tiempo con un videojuego; al pensar en un equipo que será empleado para dar servicio a un número de usuarios considerable y que trabajaran de forma concurrente en el sistema o montar un servidor de bases de datos o un servidor de correo, entonces lo más conveniente sería tener en mente la instalación de alguna distribución de Linux. , Otra cuestión importante es el nivel de conocimientos que se posean como usuario, Windows proporciona un entorno más amigable e intuitivo para usuarios que no están tan familiarizados con los sistemas de cómputo, Linux requiere que sus usuarios tengan un nivel un poco más profundo.

Bibliografías

Arquitectura Computacional
Irv Eglander
Ed. CECSA

Computación Guía Básica
Graphics Maran
Ed. ST

Computación
Patricia Elguezbal Lopez

Running Linux, 4th Edition
Matt Welsh, Lar Kaufman, Terry Dawson, Matthias Kalle Dalheimer
Ed. O'Reilly

The Linux Cookbook, 2nd Edition: Tips and Techniques for Everyday Use
Michael Stutz

Linux: Configuration and Installation (Mis Press Slackware Series)
Patrick Volkerding, Kevin Reichard, Eric F. Johnson
Ed. Mis Pr

Linux Universe : Installation and Configuration
Stefan Strobel, Rainer Maurer, Stefan Middendorf
Ed. Verlag

Manual Imprescindible De Html 4.1 (Edición Revisada Y Actualizada 2006)
German Galeano Gil
Anaya Multimedia-Anaya Interactiva

Navegar En Internet Html 4: Diseño Y Creación De Paginas Web
Ramón Soria Momparler
Alfaomega Grupo Editor

Desarrollo Web Con PHP, Apache Y MySQL
Michael K. Glass
Anaya Multimedia-Anaya Interactiva

Run Your Own Web Server Using Linux & Apache
Steidler-Dennison
Ed. Sitepoint

Apache Administrator's Handbook
Rich Bowen, Daniel Lopez Ridruejo, Allan Liska
Ed. Sams

Php Y Mysql: Tecnologias Para El Desarrollo De Aplicaciones Web
Angel Cobo
Ed. Ediciones Diaz De Santos

Programming PHP

Rasmus Lerdorf, Kevin Tatroe, Peter MacIntyre
Ed. O'Reilly

Creación De Un Portal Con Php Y Mysql
Jacobo Pavon Puertas
Ed. Ra-Ma

Mysql
Paul Dubois
Ed. Anaya Multimedia-Anaya Interactiva

Computer Security Fundamentals
Chuck Easttom

Computer Security: Art and Science
Matt Bishop

Computer Security Handbook
Seymour Bosworth, Michel E. Kabay

Computer Security Basics
Debby Russell, Sr. G.T Gangemi