UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES ARAGÓN

INFORME DEL PROYECTO DE SISTEMA DE TRANSACCIÓN WEB BASADO EN EL DIPLOMADO DE DESARROLLO E IMPLEMENTACIÓN DE SISTEMAS CON SOFTWARE LIBRE EN LINUX

TESIS
PARA OPTAR POR EL GRADO DE

INGENIERO EN COMPUTACIÓN

PRESENTA

JORGE GARRIDO RESÉNDIZ

TUTOR: **SILVIA VEGA MUYTOY**

SINODALES: **JOSÉ GONZÁLEZ BEDOLLA**
BLANCA ESTELA CRUZ LUÉVANO
EVERARDO SOLÍS ALCANTAR
IMELDA DE LA LUZ FLORES DÍAZ

AÑO: **2006**

INFORME DEL PROYECTO DE SISTEMA DE TRANSACCIÓN WEB BASADO EN EL DIPLOMADO DE DESARROLLO E IMPLEMENTACIÓN DE SISTEMAS CON SOFTWARE LIBRE EN LINUX

INDICE

INFORME DEL PROYECTO DE SISTEMA DE TRANSACCIÓN WEB BASADO EN EL DIPLOMADO DE DESARROLLO E IMPLEMENTACIÓN DE SISTEMAS CON SOFTWARE LIBRE EN LINUX

CAPÍTULO I

INFORME GENERAL DEL DIPLOMADO DE MANTENIMIENTO PREVENTIVO Y CORRECTIVO MENOR PARA COMPUTADORAS PERSONALES

OBJETIVO GENERAL

Proporcionar a los participantes los conocimientos técnicos necesarios, así como las estrategias de acción que les permitan solucionar, de manera óptima, los problemas de hardware y software en computadoras personales.

CURSOS

CURSOS	DURACIÓN
Taller de mediciones eléctricas y manejo de herramientas	20 horas
Herramientas de software preventivo y correctivo	30 horas
Taller de mantenimiento correctivo	20 horas
Introducción a las redes locales	20 horas
Actualización de equipo de cómputo	20 horas

1.1 - *Taller de mediciones eléctricas y manejo de herramientas*

La incursión de las PCs en todos los ámbitos de la actividad humana implica nuevos requerimientos para transferencia de datos y energía, por ello, es fundamental conocer las magnitudes y características de los parámetros eléctricos, así como las técnicas para medirlos e interpretarlos. Por otra parte, es de particular importancia determinar cuando el origen de una falla se debe al cableado y las técnicas de reparación, sustitución y creación de cables, para la solución de problemas específicos. Este módulo tiene la finalidad de proporcionar al participante los conocimientos suficientes que le permitan detectar y resolver este tipo de problemas.

Objetivo

Reconocer el manejo adecuado de herramientas para el mantenimiento de equipos de cómputo, además de la construcción y reparación de cables para transmisión de datos.

Contenido

Todo dispositivo eléctrico/electrónico maneja un conjunto de variables eléctricas, las cuales están definidas en el Sistema Internacional de Unidades el cual especifica sus unidades correspondientes.

VARIABLE	UNIDAD	SIMBOLO	EQUIVALENCIA
Diferencia de potencial	Volt	V	J/c
Corriente Eléctrica	Amper	A	c/s
Potencia	Watts	W	J/s
Resistencia	Ohms	Ω	V/A
Frecuencia	Hertz	Hz	1/s

Para medir los diferentes parámetros eléctricos existe una gran variedad de dispositivos capaces de medir tales valores, algunos dispositivos son especializados mientras que otros son más universales. Entre los dispositivos más comunes para medir valores eléctricos está el multímetro. El multímetro típicamente maneja tres dispositivos en uno sólo: un Voltímetro, un Amperímetro y un Ohmetro; sin embargo suele traer más funciones tales como medidor de continuidad, medidor de diodos, punta lógica, frecuenciómetro, inductancia, capacitancia entre otros.
Otro dispositivo muy útil es un osciloscopio, el cual típicamente mide voltaje y corriente tanto alterna como directa, pero a diferencia de un multímetro común, el osciloscopio puede mostrar la forma de onda de la corriente y/o el voltaje.

Los rangos de voltaje en fuentes de computadoras siguen un estándar de colores en su cableado; generalmente utilizan un conector macho de 20 pines para alimentar directamente a la tarjeta madre, aunque modelos muy antiguos llamados PC8 y PC9 utilizaban dos conectores separados de 6 pines cada uno, y las tarjetas madre más exigentes requieren 24 pines para funcionar correctamente. Aún así, los colores y sus respectivos voltajes en los cables siempre se han mantenido iguales.

COLOR DEL CABLE	VOLTAJE NOMINAL(Volts)	RANGO ACEPTABLE	AMPERAJE MANEJABLE
Amarillo	+12	+8.5 a +12.6	0.0 a 2.0
Azul	-12	-8.5 a -12.6	0.0 a .25
Rojo	+5	+4.8 a +5.2	2.3 a 7.0
Blanco	-5	-4.5 a -5.4	0.0 a 0.30

Nota: Los cables negros deberán tener siempre cero voltios pues son la tierra del sistema.

Las computadoras eventualmente se enfocaron a compartir información entre diferentes equipos; la forma más básica de hacerlo es mediante cables de comunicación. Un cable de comunicación permite intercambiar y copiar archivos entre dos computadoras, estos cables emplean los puertos de comunicación u otros dispositivos.

Cable serial

A través de los puertos seriales de ambas computadoras es posible copiar en forma más o menos rápida archivos y directorios. Comúnmente llamado puerto serial, Com1, Com2, ComN o de Mouse. Tiene conector DB de 9 y 25 pines (db9 o db25).
Ventaja:
- Una solución económica y sencilla.

Cable paralelo

A través del puerto paralelo de ambas computadoras es posible copiar en forma más rápida los datos necesarios. Comúnmente se le conoce como LPT1 o LPT2. Tiene conector de DB 25 pines (db25).
Ventaja:
- La comunicación a través del puerto paralelo es más rápida que por el serial.

Cable USB

Como resultado de un intento de dotar al PC de un bus de alta velocidad que ofreciera las características ideales PnP de universalidad, facilidad de conexión y desconexión, incluso en caliente ("Hot Swappable"), y sobre todo, que consumiese pocos recursos, Intel y otros líderes de la industria diseñaron el denominado puerto **USB Universal Serial Bus**, que como su nombre indica, es un bus serie, bidireccional y de bajo coste; diseñado como una extensión en la arquitectura estándar del PC, orientado principalmente en la integración de periféricos, que aparecen como un solo puerto en lo que se refiere a utilización de recursos.

Conectores más comunes en una computadora

CONECTOR DE VIDEO VGA DB15	
Es un conector macho que se emplea en los cables de monitores. VGA = Video Graphics Adapter/Array	
CONECTOR DIN	
El conector DIN (o mini DIN) se emplea para la conexión de teclados a la computadora. Transmite los datos en modo serial.	
CONECTOR PS/2	
El conector PS/2 se emplea para la conexión de los ratones a la computadora. Operan en modo serial	

Con este módulo se obtuvieron las herramientas básicas para poder analizar y actuar acorde a las estructuras de hardware que se presenten en un equipo.

1.2 - *Herramientas de software preventivo y correctivo*

Es fundamental contar con las herramientas de software para garantizar la integridad de la información almacenada en una computadora y, en caso necesario, realizar la reparación de fallas en el sistema operativo o aplicaciones en los equipos de cómputo. Además de proteger, es importante saber como organizar, administrar y recuperar dicha información.

Objetivo

Utilizar diversas herramientas de software para proteger, administrar, respaldar y optimizar el manejo de información; también, para determinar las características de un equipo de cómputo.

Contenido

Microsoft Windows es un sistema operativo multitarea que permite a múltiples aplicaciones o procesos correr al mismo tiempo. Cada uno es gobernado por una cantidad determinada de tiempo, llamada *time slice,* donde se le permite a la aplicación tener el control del sistema sin interrupciones de otros procesos. El encargado de determinar este time slice, es el administrador de tareas, que otorga permisos de tiempo, y determina prioridades para cada uno de los procesos. Básicamente es el administrador del tiempo del sistema operativo, ya que se asegura de que cada tarea tenga el tiempo y la prioridad adecuada. Cada proceso puede romperse en sub-partes llamadas threads(hilos), donde cada uno puede ejecutar su propio código para permitir una serie de multitareas de una misma aplicación, su manejo es similar al de los procesos.

Windows funciona mediante archivos y carpetas, en donde una carpeta es simplemente una dirección a la localidad de los archivos; todas las carpetas están contenidas en el directorio root, o la letra de unidad del disco duro. Para controlar el hardware de la computadora, Windows utiliza drivers, que son programas que controlan la forma en que se comunica la computadora con un determinado dispositivo, como puede ser una impresora o un Mouse. El sistema operativo es muy amigable con el usuario para presentarle los dispositivos que tienen conflictos con su driver (o controlador), o que simplemente no están funcionando.

Existe un problema serio en cuanto a la seguridad en las computadoras, debido al valor de la información que se almacena en ellas, es de vital importancia combatir a los programas maliciosos que puedan afectarla. Los virus, gusanos, spyware, ... son programas informáticos que se ejecutan normalmente sin el consentimiento del legítimo propietario y que tienen la características de ejecutar recursos, consumir memoria e incluso eliminar o destrozar la información. Una característica adicional es la capacidad que tienen de propagarse. Otras características son el robo de información, la pérdida de esta, la capacidad de suplantación, que hacen que reviertan en pérdidas económicas y de imagen. Existen múltiples medios de intentar combatir el problema, sin embargo hay que ser realistas, conforme nuevos programas y sistemas operativos se introduzcan en el mercado más difícil va a ser tener controlados a todos y más sencillo va a ser que a alguien se le ocurran nuevas formas de infectar el sistema. Ante este tipo de problemas esta el software llamados antivirus. Estos antivirus tratan de descubrir las trazas que ha dejado un software malicioso, para eliminarlo o detectarlo, y en algunos casos contener o parar la contaminación. Básicamente, un antivirus compara el código de cada archivo con una base de datos de los códigos (también conocidos como firmas o vacunas) de los virus conocidos, por lo que es importante actualizarla periódicamente a fin de evitar que un virus nuevo no sea detectado. También se les ha agregado funciones avanzadas, como la búsqueda de comportamientos típicos de virus (técnica conocida como Heurística) o la verificación contra virus en redes de computadores. Normalmente un antivirus tiene un componente que se carga en memoria y permanece en ella para

verificar todos los archivos abiertos, creados, modificados y ejecutados en tiempo real. Es muy común que tengan componentes que revisen los adjuntos de los correos electrónicos salientes y entrantes, así como los scripts y programas que pueden ejecutarse en un navegador Web (ActiveX, Java, JavaScript).

Con este módulo se aprendió a controlar el funcionamiento correcto de los equipos de cómputo y a mantenerlos asegurados contra intrusiones de software.

1.3 - *Taller de mantenimiento correctivo*

La actividad más compleja en el ámbito del mantenimiento a computadoras personales es la de reemplazar componentes de hardware, ya que la elección inadecuada de un componente puede generar problemas adicionales.

Objetivo

Reconocer las principales fallas de hardware que presentan los equipos PC e identificar los componentes que las ocasionan, a través de herramientas de software de detección, aplicando los criterios y consideraciones necesarios para elegir y reemplazar adecuadamente un componente.

Contenido

Cuando un componente de hardware de la computadora está fallando y requiere sustituirse, se vuelve prioritario conocer su funcionamiento y características, con objeto de efectuar el reemplazo con base en una serie de pruebas realizadas a la computadora, además tomando en cuenta las fallas existentes.

La manera más eficiente de detectar la causa del problema es teniendo una base de conocimientos elementales acerca de los componentes que podrían estar ocasionándolo. Existen fallas de hardware y fallas de software, aunque a veces es difícil detectar cual es la que está afectando; la forma más sencilla es probar primero el cableado.

Revisar que todos los sockets y cables estén conectados correctamente puede parecer simplista, sin embargo, es esencial hacerlo ya que no toma mucho tiempo, y en ocasiones la causa del problema es muy simple.
Una vez que se cerciora completamente de que los cables están en su lugar, y que no existe ningún falso contacto, entonces es necesario comenzar a analizar el hardware pertinente al problema –si el problema es con la red, la tarjeta de red puede ser la causante; si es de sonido, la tarjeta de sonido, etc. – conectándolo a un equipo diferente para comprobar que esté funcionando correctamente. En caso de no ser posible conectarlo a otro equipo, existen herramientas capaces de “aislar” dispositivos en un sistema, es decir, eliminar otras variables en el ambiente de operación para poder comprobar que no existan conflictos entre dispositivos.
De esa manera se elimina la posibilidad de que el cableado o el hardware sean los culpables, lo cual deja al software. El software puede causar fallas si los drivers o controladores no están actualizados, o tal vez hay un conflicto de interrupciones entre dispositivos. Muchas veces es la configuración más básica la que tiene la causa del error al que se enfrenta.

Para resolver un problema en un equipo, es necesario ampliar la visión del mismo para tener en cuenta todos los posibles causantes del mismo. La mejor forma de resolución es una aproximación global al equipo, que gradualmente se vuelva más minuciosa y localizada. Muchas veces la solución a un problema es mucho más simple de lo que se imagina.

1.4 - *Introducción a las redes locales*

Las redes constituyen una importante herramienta para la transferencia de información y el aprovechamiento adecuado de los recursos de hardware, en ese sentido, la incorporación de nuevas tecnologías y la velocidad de transferencia de datos las hacen parte medular de las organizaciones.

Objetivo

El participante implementará una red de datos local.

Contenido

La LAN (por su acrónimo en inglés Local Area Network) más antigua y popular, ARCnet, fue lanzada en 1977 por Datapoint y fue originalmente diseñada para compartir múltiples discos de almacenamiento Datapoint 2200. Como todas las LANs antiguas, ARCnet era originalmente específica según cada vendedor. Los esfuerzos de estandarización por parte del IEEE resultaron en la serie IEEE 802. Actualmente hay dos tecnologías comunes de cableado para LAN, Ethernet y Token Ring. Tecnologías sin cables también existen y son convenientes para usuarios de equipos móviles.

Ethernet es la tecnología de red LAN más usada, resultando idónea para aquellos casos en los que se necesita una red local que deba transportar tráfico esporádico y ocasionalmente pesado a velocidades muy elevadas. Las redes Ethernet se implementan con una topología física de estrella y lógica de bus, y se caracterizan por su alto rendimiento a velocidades de 10-100 Mbps. El origen de las redes Ethernet hay que buscarlo en la Universidad de Hawai, donde se desarrollo, en los años setenta, el Método de Acceso Múltiple con Detección de Portadora y Detección de Colisiones, CSMA/CD (Carrier Sense and Multiple Access with Collition Detection), utilizado actualmente por Ethernet. Este método surgió ante la necesidad de implementar en las islas Hawai un sistema de comunicaciones basado en la transmisión de datos por radio, que se llamó Aloha, y permite que todos los dispositivos puedan acceder al mismo medio, aunque sólo puede existir un único emisor encada instante. Con ello todos los sistemas pueden actuar como receptores de forma simultánea, pero la información debe ser transmitida por turnos. El centro de investigaciones PARC (Palo Alto Research Center) de la Xerox Corporation desarrolló el primer sistema Ethernet experimental en los años 70, que posteriormente sirvió como base de la especificación 802.3 publicada en 1980 por el Institute of Electrical and Electronic Engineers (IEEE). Las redes Ethernet son de carácter no determinista, en la que los hosts pueden transmitir datos en cualquier momento. Antes de enviarlos, escuchan el medio de transmisión para determinar si se encuentra en uso. Si lo está, entonces esperan. En caso contrario, los hosts comienzan a transmitir. En caso de que dos o más hosts empiecen a transmitir tramas a la vez se producirán encontronazos o choques entre tramas diferentes que quieren pasar por el mismo sitio a la vez. Este fenómeno se denomina colisión, y la

porción de los medios de red donde se producen colisiones se denomina dominio de colisiones. Una colisión se produce pues cuando dos máquinas escuchan para saber si hay tráfico de red, no lo detectan y, acto seguido transmiten de forma simultánea. En este caso, ambas transmisiones se dañan y las estaciones deben volver a transmitir más tarde. Para intentar solventar esta pérdida de paquetes, las máquinas poseen mecanismos de detección de las colisiones y algoritmos de postergación que determinan el momento en que aquellas que han enviado tramas que han sido destruidas por colisiones pueden volver a transmitirlas. Existen dos especificaciones diferentes para un mismo tipo de red, Ethernet y IEEE 802.3. Ambas son redes de broadcast, lo que significa que cada máquina puede ver todas las tramas, aunque no sea el destino final de las mismas. Cada máquina examina cada trama que circula por la red para determinar si está destinada a ella. De ser así, la trama pasa a las capas superiores para su adecuado procesamiento. En caso contrario, la trama es ignorada. Ethernet proporciona servicios correspondientes a las capas física y de enlace de datos del modelo de referencia OSI, mientras que IEEE 802.3 especifica la capa física y la porción de acceso al canal de la capa de enlace de datos, pero no define ningún protocolo de Control de Enlace Lógico.
Ethernet es una tecnología de broadcast de medios compartidos. El método de acceso CSMA/CD que se usa en Ethernet ejecuta tres funciones:

1. Transmitir y recibir paquetes de datos.
2. Decodificar paquetes de datos y verificar que las direcciones sean válidas antes de transferirlos a las capas superiores del modelo OSI.
3. Detectar errores dentro de los paquetes de datos o en la red.

Tanto Ethernet como IEEE 802.3 se implementan a través de la tarjeta de red o por medio de circuitos en una placa dentro del host.

Existen diferentes tipos de topologías de redes, unas más comunes que otras:

Bus.- la topología de bus tiene sus nodos conectados directamente a un enlace, y no tiene ninguna otra conexión entre nodos. Físicamente, cada host está conectado a un cable común, por lo que se pueden conectar directamente, aunque la ruptura del cable hace que los hosts queden desconectados. La topología de bus permite que todos los dispositivos de la red puedan ver todas las señales de todos los demás dispositivos, lo que puede ser ventajoso si desea que todos los dispositivos obtengan esta información. Sin embargo, puede representar una desventaja, ya que es común que se produzcan problemas de tráfico y colisiones, que se pueden arreglar segmentando la red en varias partes. Es la topología más común en las LANs pequeñas, con un hub o switch final en cada extremo.

Anillo.- Una topología de anillo se compone de un solo anillo cerrado formado por nodos y enlaces, en el que cada nodo está conectado solamente con los dos nodos adyacentes. Los dispositivos se conectan directamente entre sí por medio de cables en lo que se denomina una cadena margarita. Para que la información pueda circular, cada estación debe transferir la información a la estación adyacente.

Anillo Doble.- Una topología en anillo doble consta de dos anillos concéntricos, donde cada host de la red está conectado a ambos anillos, aunque los dos anillos no están conectados directamente entre sí. Es análoga a la topología de anillo, con la diferencia de que, para incrementar la confiabilidad y flexibilidad de la red, hay un segundo anillo

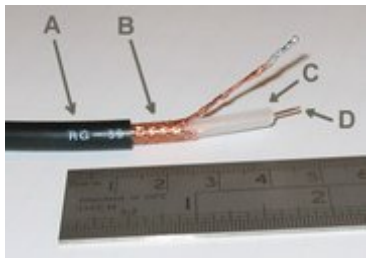
redundante que conecta los mismos dispositivos. La topología de anillo doble actúa como si fueran dos anillos independientes, de los cuales se usa solamente uno por vez.

Estrella.- La topología en estrella tiene un nodo central desde el que se irradian todos los enlaces hacia los demás nodos. Por el nodo central, generalmente ocupado por un hub, pasa toda la información que circula por la red. 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.

Estrella Extendida.- La topología en estrella extendida es igual a la topología en estrella, con la diferencia de que cada nodo que se conecta con el nodo central también es el centro de otra estrella. Generalmente el nodo central está ocupado por un hub o un switch, y los nodos secundarios por hubs. La ventaja de esto es que el cableado es más corto y limita la cantidad de dispositivos que se deben interconectar con cualquier nodo central. La topología en estrella extendida es sumamente jerárquica, y busca que la información se mantenga local. Esta es la forma de conexión utilizada actualmente por el sistema telefónico.

Árbol.- La topología en árbol es similar a la topología en estrella extendida, salvo en que no tiene un nodo central; tiene en cambio, un nodo de enlace troncal, generalmente ocupado por un hub o switch, desde el que se ramifican los demás nodos. El enlace troncal es un cable con varias capas de ramificaciones, y el flujo de información es jerárquico. Conectado en el otro extremo al enlace troncal generalmente se encuentra un host servidor.

Malla Completa.- En una topología de malla completa, cada nodo se enlaza directamente con los demás nodos. 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. 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 los enlaces se torna abrumadora.
Red Celular.- La topología celular está compuesta por áreas circulares o hexagonales, cada una de las cuales tiene un nodo individual en el centro. La topología celular es un área geográfica dividida en regiones (celdas) para los fines de la tecnología inalámbrica. En esta tecnología no existen enlaces físicos; sólo hay ondas electromagnéticas.
La ventaja obvia de una topología celular (inalámbrica) es que no existe ningún medio tangible aparte de la atmósfera terrestre o el del vacío del espacio exterior (y los satélites). Las desventajas son que las señales se encuentran presentes en cualquier lugar de la celda y, de ese modo, pueden sufrir disturbios y violaciones de seguridad.
Como norma, las topologías basadas en celdas se integran con otras topologías, ya sea que usen la atmósfera o los satélites.
Irregular.- En este tipo de topología no existe un patrón obvio de enlaces y nodos. El cableado no sigue un modelo determinado; de los nodos salen cantidades variables de cables. Las redes que se encuentran en las primeras etapas de construcción, o se encuentran mal planificadas, a menudo se conectan de esta manera.
Es importante conocer las opciones que se tienen en cuanto a cableado para una red; existen dos principales tipos de cables con sus respectivas ventajas y desventajas:

CABLE	DESCRIPCIÓN	ASPECTO
Cable Coaxial	Está formado por dos conductores concéntricos. El conductor central o núcleo está formado por un hilo sólido de cobre (llamado positivo o vivo), rodeado por una capa aislante (llamado dieléctrico) que lo separa del externo, formado por una malla trenzada de cobre o aluminio, este conductor produce un efecto de recubrimiento y además sirve como retorno de las corrientes. Todo el conjunto está protegido por una cubierta aislante.	
Cable UTP	(Unshielded Twisted Pair) Es un cable de cobre que consta de uno o más pares, ninguno de los cuales está apantallado. Cada par es un conjunto de dos conductores aislados con un recubrimiento plástico; este par se trenza para que las señales transportadas por ambos conductores no generen interferencias ni resulten sensibles a emisiones.	

Entre las ventajas del cable UTP: es más económico, flexible, delgado y fácil de instalar; además no necesita mantenimiento, ya que ninguno de sus componentes precisa ser puesto a tierra. Entre las desventajas: presenta menor protección frente a interferencias electromagnéticas, pero la que ofrece es suficiente para la mayoría de instalaciones.
La ventaja del cable coaxial es que no es habitualmente afectado por interferencias externas, y es capaz de lograr altas velocidades de transmisión en largas distancias. Sin embargo es un cable caro, y muchas veces rígido; es propenso a trozarse con torceduras bruscas o cuando se le dobla.

El Protocolo de Control de Transmisión (TCP en sus siglas en inglés, Transmission Control Protocol) es uno de los protocolos fundamentales en Internet. Muchos programas dentro de una red de datos compuesta por ordenadores pueden usar TCP para crear conexiones entre ellos a través de las cuales enviarse datos. El protocolo garantiza que los datos serán entregados en su destino sin errores y en el mismo orden en que se

transmitieron. También proporciona un mecanismo para distinguir distintas aplicaciones dentro de una misma máquina, a través del concepto de puerto. TCP soporta muchas de las aplicaciones más populares de Internet, incluidas HTTP, SMTP y SSH.

Cada computadora es identificada por un número único, esta función la cumple el IP, se puede asignar IPs dinámicos o estáticos; para una red LAN, siempre es recomendable tener IPs estáticos para mantener un control más preciso sobre los equipos y las tareas que están llevando a cabo.

1.5 - *Actualización de equipos de cómputo*

El constante y acelerado desarrollo tecnológico ha impactado a los equipos de cómputo. Día con día, es natural encontrar actualizaciones de dispositivos que continúan haciendo mucho más amigable el uso de computadoras personales.
La finalidad de este módulo es actualizar y agregar componentes, establecer criterios para actualización de equipos, además de adquirir las habilidades que le permitan detectar fallas y conflictos en la instalación de dispositivos y en la configuración de la computadora.

Objetivo

Aplicar los elementos teóricos y técnicos indispensables para actualizar e instalar componentes adicionales en una PC.

Contenido

Un bus es un subsistema que transfiere datos o electricidad entre componentes del ordenador dentro de una computadora; puede conectar mediante lógica varios periféricos utilizando el mismo conjunto de cables. Existen varios tipos de buses:

PCI	En el tiempo de arranque del sistema, las tarjetas PCI y el BIOS interactúan y negocian los recursos solicitados por la tarjeta PCI. Tienen un Reloj de 33 Mhz. La tasa de transferencia máxima es de 133 MB por segundo y el ancho de bus puede ser de 32 ó 64 bits.
ISA	Tienen un acho de bus de 16 bits y un reloj de 8 Mhz. Este ancho de banda es insuficiente para las necesidades actuales, tales como tarjetas de vídeo de alta resolución, por lo que el bus ISA es obsoleto actualmente.
AGP	Con una velocidad 533 Mhz y tasa de transferencia de 2 GB/s; el bus AGP actualmente se utiliza exclusivamente para conectar tarjetas gráficas.

Cualquier dispositivo interno en una computadora utiliza los buses para comunicarse con el procesador, excepto por los dispositivos de almacenamiento, que utilizan una interfaz IDE o SATA.

La interfaz IDE (Integrated Device Electronics) o ATA (Advanced Technology Attachment) casi siempre están incluidas en la placa base, normalmente dos conectores para dos dispositivos cada uno. De los dos discos duros, uno tiene que estar como esclavo y el otro como maestro para que la controladora sepa de qué dispositivo mandar y recibir los datos. La configuración se realiza mediante jumpers. Habitualmente, un disco duro puede estar configurado de una de estas tres formas:

1. Como maestro (master). Si es el único dispositivo en el cable, debe tener esta configuración. Si existe otro dispositivo en el cable, este último debe estar configurado como esclavo.
2. Como esclavo (slave). Debe haber otro dispositivo que sea maestro.
3. Selección por cable (cable select). El dispositivo será maestro o esclavo en función de su posición en el cable. Ambos dispositivos en dicho cable deben estar configurados como CS (en caso de haberlos). Para distinguir el conector en el que se conectará el primer bus IDE (IDE 1) se utilizan colores distintos.

Este diseño (dos dispositivos a un bus) tiene el inconveniente de que mientras se accede a un dispositivo el otro dispositivo del mismo conector IDE no se puede usar.

La interfaz SATA se diferencia de la IDE en que utiliza un cable mucho más angosto, de solo 7 hilos, lo que mejora la ventilación del conjunto; así mismo, cada disco se conecta directamente a la tarjeta madre (también llamado punto a punto), lo que elimina completamente el problema de dos discos simultáneos en IDE. Este nuevo estándar es compatible con el sistema IDE actual. Como su nombre indica (Serial ATA) es una conexión tipo serie como USB o FireWire. La primera versión ofrece velocidades de hasta 150MB/s, con Serial ATA II permitiendo 300MB/s. Se espera que alcance los 600MB/s alrededor de 2007.

Actualizar y mantener en funcionamiento óptimo una computadora personal es sencillo si se tiene en cuenta el funcionamiento básico del conjunto de hardware y software. Una actualización generalmente supone una restauración de datos previos a la misma, por esto es recomendable llevar un archivo de respaldo constante. Aunque la compatibilidad entre componentes es muy amplia, y rara vez se presentan conflictos insolubles, es imprescindible informarse acerca de todas las características de las piezas y componentes del equipo o equipos y tener una idea clara de cómo se conectarán unos con otros.

CAPÍTULO II

INFORME GENERAL DEL DIPLOMADO DE DESARROLLO E IMPLEMENTACION DE SISTEMAS CON SOFTWARE LIBRE EN LINUX

En la actualidad es necesario contar con sistemas que permitan la automatización del seguimiento y control de procesos que se desarrollan en diversas empresas y oficinas. Con esta automatización se tiene un uso más eficiente de los recursos y un control más preciso de los mismos, lográndose con ello compartir información entre diversas áreas y oficinas a través de la red local e incluso empleando Internet o la intranet corporativa.

El desarrollo de estos sistemas puede ser sustentado empleando software libre como LINUX, PHP, Apache Server y MySQL, que son herramientas que han demostrado tener un alto desempeño, gran estabilidad y seguridad y por el hecho de ser libres, permiten reducir los costos que se generan por las licencias de uso de software, logrando aprovechar al máximo los equipos de cómputo con que se cuente.

Debido a que el creciente uso de las computadoras implica correr riesgos de seguridad en el acceso a la información, se ha vuelto necesario hacer conciencia de que la seguridad es una responsabilidad compartida; por lo que todos los usuarios de computadoras requieren un conocimiento esencial de los elementos de la seguridad que les ofrece Linux.

OBJETIVO GENERAL

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

CURSOS

CURSOS	DURACIÓN
Sistema Operativo Linux	20 horas
Instalación y Administración de Linux	20 horas
Editores para la Creación de Páginas Web	10 horas
Administración de Servidores WWW con Linux	20 horas
Programación con PHP	20 horas
Interacción de WWW con Bases de Datos	20 horas
Introducción a la Seguridad en Cómputo	20 horas

2.1 - Sistema Operativo Linux

Objetivo

El participante conocerá el manejo del sistema operativo Linux, así como utilerías básicas, manipulará archivos, directorios, editores de texto y nociones básicas de administración.

Contenido

Linux es la denominación de un sistema operativo y el nombre de un núcleo. Es uno de los paradigmas del desarrollo de software libre (y de código abierto), donde el código fuente está disponible públicamente y cualquier persona, con los conocimientos informáticos adecuados, puede libremente usarlo, modificarlo y redistribuirlo.

El término Linux estrictamente se refiere al núcleo Linux, pero es más comúnmente utilizado para describir al sistema operativo tipo Unix (que implementa el estándar POSIX), que utiliza primordialmente filosofía y metodologías libres (también conocido como GNU/Linux) y que está formado mediante la combinación del núcleo Linux con las bibliotecas y herramientas del proyecto GNU y de muchos otros proyectos/grupos de software (libre o no libre). El núcleo no es parte oficial del proyecto GNU (el cual posee su propio núcleo en desarrollo, llamado Hurd), pero es distribuido bajo los términos de la licencia GPL (GNU General Public License). La expresión Linux también es utilizada para referirse a las distribuciones Linux, colecciones de software que suelen contener grandes cantidades de paquetes además del núcleo. El software que suelen incluir consta de una enorme variedad de aplicaciones, como: entornos gráficos, suites ofimáticas, servidores web, servidores de correo, servidores FTP, etcétera. Coloquialmente se aplica el término Linux a éstas, aunque en estricto rigor sea incorrecto, dado que la distribución es la forma más simple y popular para obtener un sistema Linux. La marca Linux (Número de serie: 1916230) pertenece a Linus Torvalds y se define como "un sistema operativo para computadoras que facilita su uso y operación". Desde su lanzamiento, Linux ha incrementado su popularidad en el mercado de servidores. Su gran flexibilidad ha permitido que sea utilizado en un rango muy amplio de sistemas de cómputo y arquitecturas: computadoras personales, supercomputadoras, dispositivos portátiles, etc.

Los sistemas Linux funcionan sobre más de 20 plataformas diferentes de hardware; entre ellas las más comunes son las de los sistemas compatibles con PCs x86 y x86-64, computadoras Macintosh, PowerPC, Sparc y MIPS.
Asimismo, existen Grupos de Usuarios de Linux en casi todas las áreas del planeta.
Una distribución es un conjunto de aplicaciones reunidas por un grupo, empresa o persona para permitir instalar fácilmente un sistema Linux. Es un 'sabor' de Linux. En general se destacan por las herramientas para configuración y sistemas de paquetes de software a instalar.
Existen numerosas distribuciones Linux (también conocidas como "distros"), ensambladas por individuos, empresas y otros organismos. Cada distribución puede incluir cualquier número de software adicional, incluyendo software que facilite la instalación del sistema. La base del software incluido con cada distribución incluye el núcleo Linux, al que suelen adicionarse también varios paquetes de software.
Las herramientas que suelen incluirse en las distribuciones de este sistema operativo se obtienen de diversas fuentes, incluyendo de manera importante proyectos de código abierto o libre, como el GNU y el BSD. Debido a que las herramientas que en primera instancia volvieron funcional al núcleo de Linux provienen de un proyecto anterior a Linux; Richard Stallman (fundador del proyecto GNU) pide a los usuarios que se refieran a dicho sistema como GNU/Linux. A pesar de esto, la mayoría de los usuarios continúan llamando al sistema simplemente "Linux" y las razones expuestas por

Richard Stallman son eterno motivo de discusión. La mayoría de los sistemas Linux incluyen también herramientas procedentes de BSD.

Según Richard Stallman, GNU/Linux es la denominación para el sistema operativo que utiliza el kernel Linux en conjunto con las aplicaciones de sistema creadas por el proyecto GNU. Comúnmente este sistema operativo es denominado simplemente Linux. Desde 1984, Richard Stallman y algunos voluntarios están intentando crear un sistema operativo libre con un funcionamiento similar al UNIX, recreando todos los componentes necesarios para tener un sistema operativo funcional que se convertiría en el sistema operativo GNU. En el comienzo de los años 1990, después de seis años, GNU tenía muchas herramientas importantes listas, como compiladores, depuradores, intérpretes de comando etc. excepto por el componente central: el núcleo. En ese momento Linus Torvald estaba estudiando en la Universidad de Helsinki y estaba trabajando en su proyecto de Linux. Con el surgimiento del kernel Linux, esta laguna fue llenada y surgió el sistema operativo con el kernel Linux en conjunto con las herramientas GNU. De esta manera, Stallman juzga que este sistema operativo es una "versión modificada" del sistema GNU y por lo tanto debe tener la denominación GNU/Linux. Esta denominación resolvería la confusión entre el núcleo y el sistema operativo completo a que puede llevar, y de hecho ha llevado, la denominación Linux en solitario. Stallman también espera que con el aporte del nombre GNU, se dé al proyecto GNU que él encabeza el reconocimiento que merece por haber creado las aplicaciones de sistema imprescindibles para ser un sistema operativo compatible con UNIX. Es conocida la cita de Stallman: «It's GNU/Linux, dammit!» («¡Es GNU/Linux, maldita sea!»).
Richard Stallman ha reconocido que desde que existe Linux el desarrollo de un núcleo específico del proyecto GNU (el Hurd) ya no es prioritario. Esto explica que después de dos décadas desde el anuncio del proyecto GNU, un sistema únicamente GNU no esté acabado.

Algunas distribuciones apoyan esta denominación, e incluyen GNU/Linux en sus nombres, tal es el caso de Debian GNU/Linux o GNU/LinEx. En el proyecto Debian también existe Debian GNU/Hurd y Debian GNU/BSD que combinan las aplicaciones de sistema de GNU con esos núcleos. En ocasiones, el proyecto KDE ha utilizado una tercera denominación: GNU/Linux/X para enfatizar los tres proyectos sobre los que se apoya su entorno de escritorio.

Algunos sectores de la comunidad de usuarios del sistema operativo han rechazado la denominación GNU/Linux por varias razones, entre ellas que ya se había empezado a denominar Linux al sistema operativo antes de que Richard Stallman promocionase esta denominación. Otras personas se oponen a la postura ideológica de Stallman radicalmente en contra del software no libre y por ello son contrarios al uso de este nombre para evitar la promoción de las ideas del fundador del proyecto GNU. Otros sectores de la comunidad han reconocido la conveniencia de este nombre.
Hay que señalar que, al igual que es una simplificación denominar al sistema que usa el usuario final Linux, obviando las aplicaciones GNU que completan el sistema operativo, el conjunto Linux+GNU representa solamente una parte (aunque importante) del software encontrado en una distribución Linux. Existe una gran cantidad de software original del sistema operativo BSD o producido independientemente de los proyectos GNU y Linux por otras personas u organizaciones, como por ejemplo Apache, el X Window System, Samba, GNOME, KDE, OpenOffice.org y miles de otros.

Usualmente se utiliza la plataforma XFree86 o la Xorg para sostener interfaces gráficas (esta última es un fork de XFree86, surgido a raíz del cambio de licencia que este proyecto sufrió en la versión 4.4 y que lo hacía incompatible con la GPL).
Con la adopción por numerosas empresas fabricantes de PCs, muchas computadoras son vendidas con distribuciones Linux preinstaladas, y Linux ha comenzado a tomar su lugar en el vasto mercado de las computadoras de escritorio.
Con entornos de escritorio, Linux ofrece una interfaz gráfica alternativa a la tradicional interfaz de línea de comandos de Unix. Existen en la actualidad numerosas aplicaciones gráficas, ya sean libres o no, que ofrecen funcionalidad que está permitiendo que Linux se adapte como herramienta de escritorio.

Algunas distribuciones permiten el arranque de Linux directamente desde un disco compacto (llamados LiveCDs) sin modificar en absoluto el disco duro de la computadora en la que se ejecuta Linux. Para este tipo de distribuciones, en general, los archivos de imagen (archivos ISO) están disponibles en Internet para su descarga.
Otras posibilidades incluyen iniciar el arranque desde una red (ideal para sistemas con requerimientos mínimos) o desde un disco flexible o disquete.

El hecho de que Linux (o GNU/Linux) sea software libre es lo que permite su existencia. Sólo es posible hacer conjeturas acerca del costo del proyecto, o mejor dicho, de cualquier proyecto basado en Linux si hubiera sido desarrollado de manera convencional. Un estudio sobre la distribución Red Hat Linux 7.1 reveló que ésta en particular posee más de 30 millones de líneas de código real. Utilizando el modelo de cálculo de costos COCOMO, puede estimarse que esta distribución requeriría 8.000 programadores por año para su desarrollo. De haber sido desarrollado por medios convencionales de código cerrado, hubiera costado más de mil millones de dólares en los Estados Unidos. La mayor parte de su código (71%) pertenecía al lenguaje C, pero fueron utilizados muchos otros lenguajes para su desarrollo, incluyendo C++, Bash, Lisp, Ensamblador, Perl, Fortran y Python. Alrededor de la mitad de su código total (contado en líneas de código) fue liberado bajo la licencia GPL.
El núcleo de Linux contenía entonces 2,4 millones de líneas de código, correspondiente al 8% del total, demostrando que la vasta mayoría del sistema operativo no pertenece al núcleo del mismo.
En un estudio posterior, Counting potatoes: the size of Debian 2.2, el mismo análisis fue hecho para Debian GNU/Linux versión 2.2. Esta distribución contiene más de cincuenta y cinco millones de líneas de código fuente, y habría costado 1.900 millones de dólares (año 2000) el desarrollo por medios convencionales (no libre). Miles de programadores voluntarios alrededor del mundo han participado en el proyecto, mejorándolo continuamente. Torvalds y otros desarrolladores de los primeros días de Linux adaptaron los componentes de GNU para trabajar con el núcleo de Linux, creando un sistema operativo completamente funcional.
Cabe mencionar que a pesar de que el núcleo de Linux se libera bajo los términos de la licencia GPL, no es parte oficial del proyecto GNU.
La creciente popularidad de Linux se debe a las ventajas que presenta ante otros tipos de software. Entre otras razones se debe a su estabilidad, al acceso a las fuentes (lo que permite personalizar el funcionamiento y auditar la seguridad y privacidad de los datos tratados), a la independencia de proveedor, a la seguridad, a la rapidez con que incorpora los nuevos adelantos (IP v6, microprocesadores de 64 bits), a la escalabilidad (se pueden crear clusters de cientos de ordenadores), a la activa comunidad de

desarrollo que hay a su alrededor, a su interoperabilidad y a la abundancia de documentación relativa a los procedimientos.
Hay varias empresas que comercializan soluciones basadas en Linux: IBM, Novell, Red Hat, así como miles de PYMES que ofrecen productos o servicios basados en esta tecnología.
Dentro del segmento de supercomputadoras, la más grande de Europa se llama MareNostrum. Desarrollado por IBM, está basado en un cluster Linux (Presentación de MareNostrum en IBM). Hay muchas más supercomputadoras funcionando con Linux.
Linux tiene una amplia cuota en el mercado de servidores de Internet debido, entre otras cosas, a la gran cantidad de soluciones que tiene para este segmento.

Después de conocer un poco de la historia y la evolución de Linux, es necesario analizar su funcionamiento. Linux es un sistema multiusuario, lo que permite tener muchas cuentas diferentes de usuarios en la computadora.

ADMINISTRACION DE ARRANQUE

El centro del sistema de ingreso de Linux es el demonio* de logging o syslogd. Este demonio normalmente se arranca desde los archivos script de arranque (rc) cuando el sistema entra a un nivel 1 de funcionamiento. Una vez corriendo, casi cualquier parte del sistema, incluyendo aplicaciones, controladores y también otros demonios pueden hacer entradas en la bitácora. Inclusive existe una interfaz de línea de comando para que el usuario haga sus propias entradas. Con Windows NT, cada sistema mantiene sus propios archivos de bitácora; no hay un lugar central donde puedan ser almacenadas. Aunque existe el visor de eventos para acceder a las bitácoras de otras máquinas, esto generalmente lleva mucho tiempo, especialmente cuando hay muchas entradas y se tiene una conexión lenta. En lugar de eso, syslogd puede ser configurado para mandar todas (o algunas) las señales a una máquina remota, que procesa y escribe en ellas. Lo que hace posible ingresar todas las bitácoras de un tipo en una red a un solo archivo, para facilitar la administración. Otra ventaja es que debido al hecho de que syslogd guarda información de la configuración y entradas de bitácora en archivos de texto, es muy simple reescribir los scripts para separarlos, alterarlos, o procesarlos de diferentes formas. Parte de esta habilidad se encuentra en el formato estándar de cada entrada de bitácora. Aunque es posible que un programa maligno pueda escribir información en cualquier orden, todos los demonios de sistema y muchos programas siguen el estándar de: fecha, hora, sistema, lugar, mensaje.

Lo que se hace y cuando se hace es determinado por el archivo syslogd.conf, que usualmente se encuentra en /etc/. Este es un archivo típico de configuración Linux con un item por línea y comentarios que comienzan con un símbolo de gato (#). Cada item consiste de una porción selectora, que determina los eventos a los que reaccionar, y una porción de acción, que determina lo que se va a hacer. La porción de selección está dividida en dos partes, que se separan por un punto. La primera parte dice que aspecto del sistema se va a alterar, y la segunda que nivel de mensajes son los que servirán como reactores. Para ambos campos, hay un comodín que puede usarse, un asterisco indica cualquier facilidad o prioridad. La palabra “none” se usa para referir ninguna prioridad para el campo en que se especifica; es en otras palabras, un valor nulo.

Cuando se escriba a archivos, hay que considerar que el sistema va a añadir información al disco con cada evento. Esto asegura que la entrada efectivamente afecte al archivo en

caso de un colapso del sistema. El problema en escribir cosas en el disco duro es que toma tiempo. Por ello el sistema normalmente salva un número de escrituras antes de mandarlas al disco.

** Un "demonio" es un pequeño programa encargado de realizar una tarea rutinaria, el término puede parecer desconocido o nuevo para usuarios de Windows, sistema que llama a los mismos "Servicios"*

ARCHIVOS Y SISTEMAS DE ARCHIVOS

Siempre que se accede a un sistema Linux, ya sea localmente o a través de una red, tanto los archivos como los sistemas de archivos son importantes. Cada programa que se corra comienza como un simple archivo que muchas veces también se están leyendo o escribiendo. Debido a que los archivos (tanto programas como archivos de datos) residen en los discos, cada vez que se accesan se involucra también al sistema de archivos.
Saber que es y como se representa un archivo en el disco y como el sistema interpreta los contenidos del mismo es útil para entender lo que el sistema está haciendo. Con este conocimiento, se puede evaluar tanto el sistema como el comportamiento de la aplicación para determinar si son los debidos.
Hay que mantener presente que es un error pensar que un directorio "contiene" otros archivos y directorios. Lo que realmente pasa es que el directorio contiene información necesaria para localizar su "contenido". Así, cuando se pide un listado de los archivos con los tamaños de cada archivo, los directorios no incluyen el tamaño colectivo de todos sus "contenidos", sino simplemente muestra cuanto espacio toma en el disco duro dicho directorio. Para obtener el espacio total de todos los subdirectorios y archivos se utiliza el comando du –s <directorio>.

En Linux, así como Unix, los sistemas de archivos separados que el sistema pueda usar no son accesados por identificadores de dispositivos (como una letra o número de drive), sino que se combinan en un solo árbol jerárquico que representa el sistema de archivos como una entidad solitaria. Linux añade cada nuevo dispositivo a este árbol como si fuera montado en un directorio. Una de las características más importantes de Linux es su soporte para muchos tipos de sistemas. Esto lo hace muy flexible y fácil de coexistir con otros sistemas operativos. El sistema más popular utilizado por Linux es el EXT2.

Un sistema de archivos le da al usuario una vista sensata de los archivos y directorios en el disco duro del sistema, sin importar su tipo o características físicas. Linux acepta de forma transparente y sencilla muchos de ellos, como MS-DOS, FAT 32, o incluso NTFS, aunque este último sólo como lectura.

Un sistema de archivos no solo contiene los datos de los archivos que tiene, sino también la estructura de los mismos. Linux tenía un sistema llamado Minix cuando comenzó, un poco restrictivo y lento; sus nombres de archivos no podían ser mayores de 14 caracteres (aunque aún era mejor que el límite de 8.3 de DOS) y el máximo tamaño de archivo es de 64 Megas. Aunque a primera vista puede parecer suficiente, las bases de datos pueden superar fácilmente ese límite.

El primer sistema de archivos específicamente diseñado para Linux fue el Extended File System, o EXT, introducido en abril de 1992 y que logró solucionar muchas dificultades existentes, aunque aún era un poco lento. En 1993, el segundo EXT, o EXT2 fue añadido.
Cuando sucedió esto, un desarrollo importante tomó lugar en Linux. Los sistemas de archivos fueron separados del operativo y del sistema por una interfaz llamada Virtual File System, o VFS. VFS permite a Linux soportar todos los sistemas que tiene, cada uno presentándose a su vez como una interfaz de software ante sí. Todos los detalles de estos sistemas de archivos se traducen por software para que todos parezcan idénticos al resto del kernel de Linux. La VFS guarda información temporalmente en la memoria de cada sistema de archivos cuando se monta y se usa. Se debe tener mucho cuidad en actualizar el sistema correctamente, ya que los datos en este almacén se modifican mientras se crean y mueven los directorios y archivos dentro del sistema. El más importante de estos componentes es el Buffer Cache, que se integra en el camino en que los sistemas de archivos tradicionales accesan a sus bloques. Cuando se accede a un bloque, este se integra en el buffer y se mantiene en varias colas dependiendo de su estado.

En todas las distribuciones actuales, se incluye una copia del sistema de ventanas Xfree86, o X-Windows. Esta utilidad es gratuita, como todos los componentes de la licencia pública GNU, y provee de una interfaz gráfica al sistema operativo.

Para arrancar X-Windows, simplemente se introduce el comando xwindows en el shell. Una vez que esté corriendo, se puede utilizar el programa de configuración zf86config para traer un asistente de configuración de video. Es importante notar que no se está corriendo un solo programa; muchos diferentes están activos al mismo tiempo, cuál se utiliza depende de las opciones especificadas durante la configuración.

El archivo de configuración generalmente se encuentra en /etc/XF86Config o en /etc/X11/XF86Config. Es un archivo de texto que se genera cada vez que se corre el asistente. El anterior se divide en tres secciones: la sección de pantalla es la primaria, y generalmente viene al último, define lo que se ve en la pantalla basándose en las otras dos secciones; la sección de dispositivos describe la tarjeta de video (dispositivo de video); y la sección de monitor describe el mismo.

Dentro de la sección de pantalla, el servidor puede ofrecer muchos campos para cada tipo de video, como el color, la resolución, y si es una pantalla “lógica” o no.

En la sección de video, se tienen tres diferentes sub secciones, que definen la profundidad del color, el número de colores, y el modo de resolución que se prefiera.

La sección de monitor muestra las características físicas del monitor que se describa, podría darse el caso de tener varias secciones de monitor, si se está utilizando más de uno. También aquí se incluye la velocidad de refresco horizontal y vertical.

USUARIOS

Los usuarios ganan acceso al sistema solamente después de que el administrador ha creado cuentas de usuario para cada uno de ellos. Estas cuentas son más que simples

nombres de usuario y passwords; también definen el ambiente bajo el que se trabaja, incluyendo el nivel de acceso que se tiene sobre el sistema.
Se pueden añadir usuarios a Linux de una de dos maneras. Se pueden crear las entradas necesarias en el archivo apropiado, crear los directorios y copiar los archivos de inicio manualmente. También se puede usar la herramienta adduser, que lo hace por nosotros. Cuando se añade un nuevo usuario, un shell se le asigna, junto con los demás archivos de configuración que le pertenecen. Al usuario también se le otorga un directorio home, el cual es el directorio default cuando ingresan.
Cada usuario recibe un nombre de usuario, que esta asociado con un número ID, un grupo al menos, con uno de ellos designado como su grupo de ingreso. Cada grupo tiene su propio ID de grupo; ambos IDs mantienen registro del usuario y determinan que archivos y directorios puede acceder.
En general, los programas y comandos que interactúan con el factor humano reportan información del usuario por el nombre de login o de grupo. Sin embargo, muchos procesos de identificación del sistema operativo se hacen por medio de los IDs.
Las cuentas de usuario se definen en /etc/passwd, y los grupos en /etc/group.
Las contraseñas escritas en estos archivos están encriptadas; al principio de cada una, hay una semilla. Utilizando esta semilla, el sistema crea la versión codificada.

EDITORES DE ARCHIVOS

De todos los editores presentes en Linux, el más básico y, algunos argumentan, poderoso es Vi. Los usos y beneficios de un editor como Vi son muchos. Muchas veces, las razones para preferir un editor sobre otro son puramente materia de gusto personal. Cada uno ofrece sus propias ventajas sobre otros.
Algunas versiones de UNIX proveen otros editores, como emacs; sin embargo, lo bueno de vi es que todos los dialectos de UNIX lo tienen, es por esta razón principalmente por la que vale la pena usarlo.
El problema de vi es que puede llegar a ser intimidante, sin embargo después de manejarlo unos minutos se puede llegar a la conclusión de que todos los comandos tienen un orden lógico. Vi puede corregir automáticamente palabras que se escriban mal frecuentemente, aceptar comandos personalizados por el usuario, entre otras cosas.

Vi tiene dos modos de empleo: comando y escritura. Mientras se está en comando, cada tecla se considera parte de un comando. Normalmente se empieza en este modo cuando se arranca vi. En modo escritura, todas las teclas presionadas se consideran texto. Para pasar a modo escritura, se utiliza Insert o la tecla I, para pasar a modo comando, se presiona Esc.

Linux ofrece un número de herramientas para poder respaldar el sistema. Tal vez la más común de todas sea tar, por su simplicidad. Tar comprime y descomprime información a un solo archivo; su sintaxis es bastante simple.

DOCUMENTACIÓN

La documentación de software es un tema controversial. Continúa siendo debatido por todos los foros. A menos que el producto sea muy intuitivo, si no esta propiamente documentado, puede convertirse en un desperdicio; y aún si es intuitivo, muchas de sus funciones permanecerán escondidas a menos de que exista una documentación buena.

Desafortunadamente, Linux no es muy intuitivo. Por lo mismo, una buena documentación es esencial para poder usarlo al máximo. A diferencia de una implementación comercial de UNIX, Linux no provee un conjunto de manuales impresos para referirse a ellos. La documentación que está disponible se encuentra en un gran número de documentos que se proveen con la distribución. Ya que la documentación fue desarrollada por muchas personas diferentes en muchos lugares diferentes, no existe una entidad que las maneje a todas.
El proyecto de documentación de Linux (LDP) fue organizado por esta misma razón. Mientras más se desarrolla Linux, más se producen documentos. Hay muchos HOW TO's (archivos de ayuda) presentes que ayudan a hacer diferentes tareas. Típicamente son bastante largos, pero entran en detalle necesario para no sólo resolver problemas específicos, sino ayudar a configurar aspectos detallados del sistema. En muchos casos, estos fueron escritos por las personas que desarrollaron el programa mismo, y es posible descargarlos del sitio web de LDP (www.tldp.org).
También incluyen frecuentemente una sección de preguntas y respuestas (faqs), que son listas de preguntas frecuentes sobre implementación, errores, o características del producto.

Afortunadamente, mucha de la información necesaria esta disponible en forma de paginas man. Dentro del sistema, está programado el comando para poder leerlas: man. Tecleando man <comando>, se puede encontrar detalles acerca del uso y el funcionamiento sobre el mismo comando. Cuando se refiere a un comando en particular en Linux, se sigue al nombre con una letra o número en paréntesis, tales como ls(1) por ejemplo. Esto indica que el comando ls puede ser encontrado en la sección 1 de las páginas man; tal información data del tiempo en que las páginas man venían impresas en libros, cuando incluían la sección, se podía localizar más fácilmente lo que se estaba buscando.

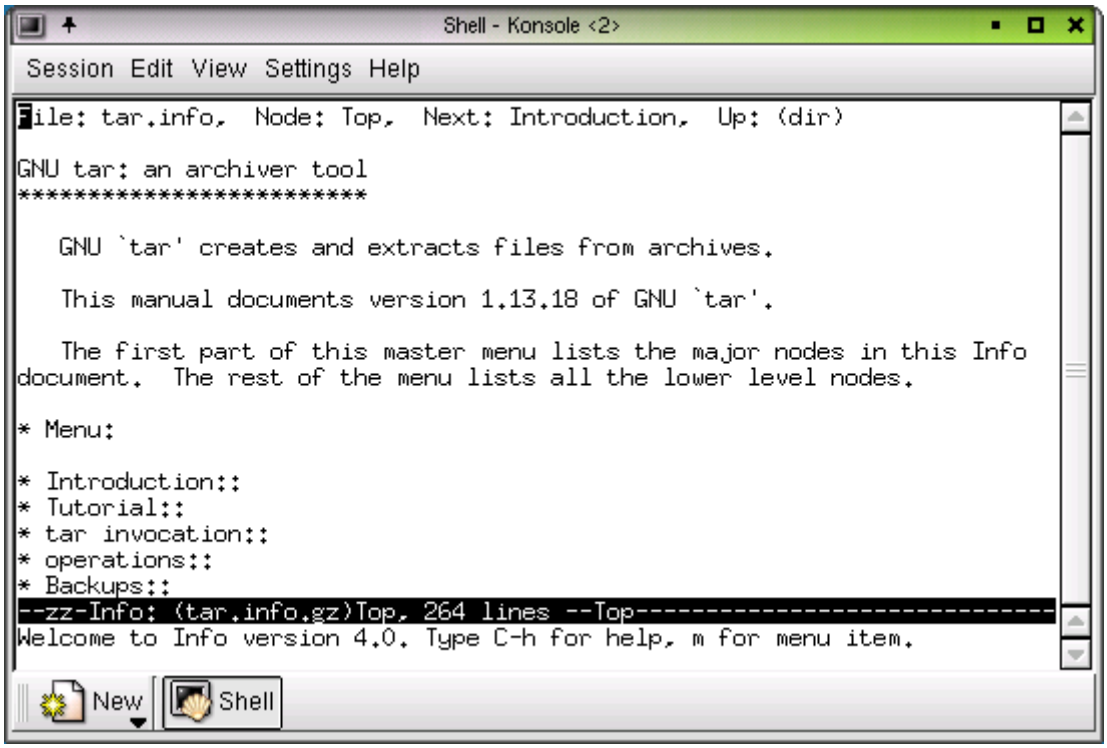
La siguiente tabla lista las secciones disponibles. Algunas veces existen múltiples man en diferentes secciones.

Sección	Descripción
1	Comandos, utilidades, y otros ejecutables típicamente relacionados con el usuario
2	Llamadas de sistema
3	Llamadas de librerias
4	Archivos especiales, típicamente en /dev/
5	Formatos de archivos y sus típicas convenciones
6	Juegos
7	Paquetes macro
8	Comandos administrativos de sistema
9	Rutinas del kernel

Las páginas man usualmente tienen el mismo formato básico, aunque no todas en las diferentes secciones están ahí para cada una. En el tope esta el nombre de la sección, que es simplemente el nombre del archivo o comando en discusión. A continuación viene la sinopsis, que provee un resumen breve del mismo. Si la página se refiere a un comando o utilidad, esta sección usualmente lista ejemplos generales de cómo el

comando funciona. La sección de descripción es, como indica su nombre, una descripción del comando, es exhaustiva. Bajo opciones se encuentran detalles sobre varios parámetros y líneas si es que existen que se pueden usar con el comando o archivo. Por último, la página puede listar otras man que puedan ofrecer más documentación relacionada y bugs conocidos.

Muchas veces las páginas man no se guardan en su forma original, sino en una forma previamente formateada llamada "cat pages". Esto se hace para hacer más rápido su despliegue, ya que las páginas no tienen que ser procesadas cada vez que se les llama. Si algún sistema no tiene estas creadas por default, cada llamada a una man regresará "No manual entry". Para resolver esto, simplemente hay que correr el comando catman.
Para muchos comandos, así como para mucha información en general del sistema, existe información adicional a la que se puede acceder mediante el comando info. Aunque no hay tantas info como comandos, aquellas contienen información en más aspectos del sistema en particular.
Por ejemplo, si se busca la info del comando tar, la respuesta sería esta página interactiva:



Como se puede apreciar, las info proveen mucha más información que una simple página de documentación man, HOW TO o faq. La principal diferencia es que son interactivas, lo que quiere decir que se puede navegar a través de ellas de manera similar a navegar en la red.
También salta a la vista que tiene un tutorial sobre el comando, algo que no está presente en una documentación estándar.

Con los conocimientos básicos acerca de la historia, el funcionamiento, y sobre todo el desarrollo continuo de Linux como parte de la comunidad de software libre se puede, ahora sí, comenzar a administrar, controlar, y desarrollar un sistema o grupo de sistemas.

2.2 - Instalación y Administración de Linux

Objetivo

El participante conocerá el proceso de instalación del sistema operativo Linux, enfocándose a la distribución Slackware, configurará los dispositivos del sistema, instalará la red y pondrá el sistema en funcionamiento total.

Contenido

Slackware es una de las distribuciones más viejas de Linux. A través de los años, es de las pocas que se ha mantenido firme en su objetivo original. En las palabras de su autor, Patrick Volkerding:

"Desde su primera aparición en Abril de 1993, el proyecto Slackware Linux ha tratado de producir la distribución Linux más parecida a UNIX que haya en el mercado. Slackware cumple con los estándares Linux publicados, tales como el LFSS (sobre sistemas de archivos). Siempre hemos considerado la simplicidad y la estabilidad lo más importante, y como resultado, Slackware se ha convertido en una de las distribuciones disponibles más populares, amigables y estables."[1]

Generalmente se dice que Slackware era de las más complicadas de manejar, asi que el término "amigable" puede resultar confuso. Es verdad que esta distribución no maneja grandes interfaces gráficas en sus asistentes, o en su instalación. Sin embargo, en esencia, Slackware es una de las distribuciones más simples que hay si se es un poco hábil con un sistema Linux. Aún en el caso contrario, un poco de perseverancia lo resolverá.

La razón por la que es sencillo para un usuario avanzado es, primero que nada, por todos los scripts de init y archivos de configuración, son muy sencillos de seguir. Generalmente están bien comentados, y es muy fácil hacer cambios utilizando un editor de texto ordinario.

No solo eso, sino que también se están incluyendo las versiones completas y estándar de software en esta distribución, instaladas de forma adecuada y lógica. Así, cuando se va a instalar software que no viene incluido con todo el paquete, no existirán muchos obstáculos.

La paquetería de Slackware es rápida y simple. Sus paquetes son básicamente archivos tar.gz, que tienen scripts de instalación para que las utilerías de empaque los ejecuten. No existe un chequeo de dependencia, lo que puede ser bueno o malo, dependiendo de cómo se vea. Puede ser ventajoso debido a que no molestarán los paquetes que no se instalen debido a un mecanismo burdo que busque ciertas cosas en un solo lugar específico; la desventaja es que también hay que ser cuidadoso al instalar software de sistema.

[1] Referencia de http://www.slackware.com/info

Slackware también provee un ambiente excelente para construir software personal desde códigos fuente.

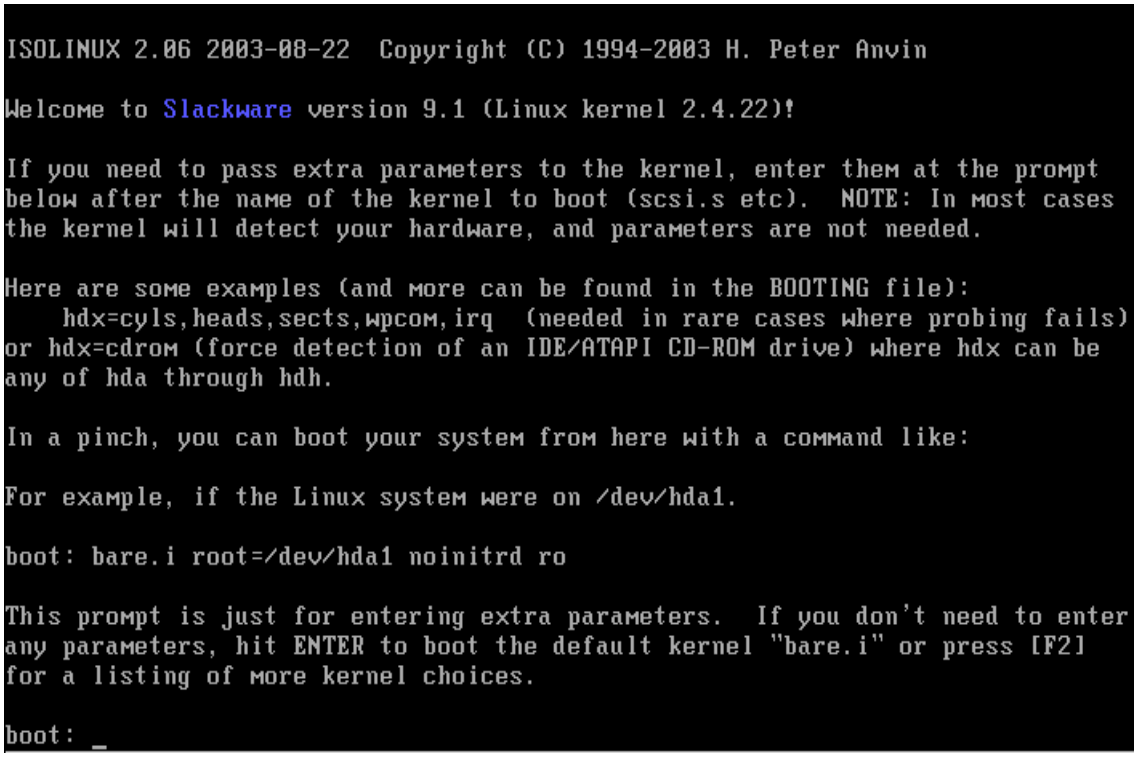
Al momento de esta redacción, Slackware está en la versión 10.2, y la rutina de instalación no ha cambiado apreciablemente, es lógico pensar que tardará mucho tiempo en presentar algún cambio considerable.

PREPARANDO LAS PARTICIONES

Antes que nada, si se desea tener un sistema operativo adicional, como Windows, hay que encargarse de él primero. Si se comienza con un disco duro nuevo, es necesario crear una partición para Windows (o cualquier sistema operativo), y dejar el resto sin particionar. De esta forma se instala Windows primero.

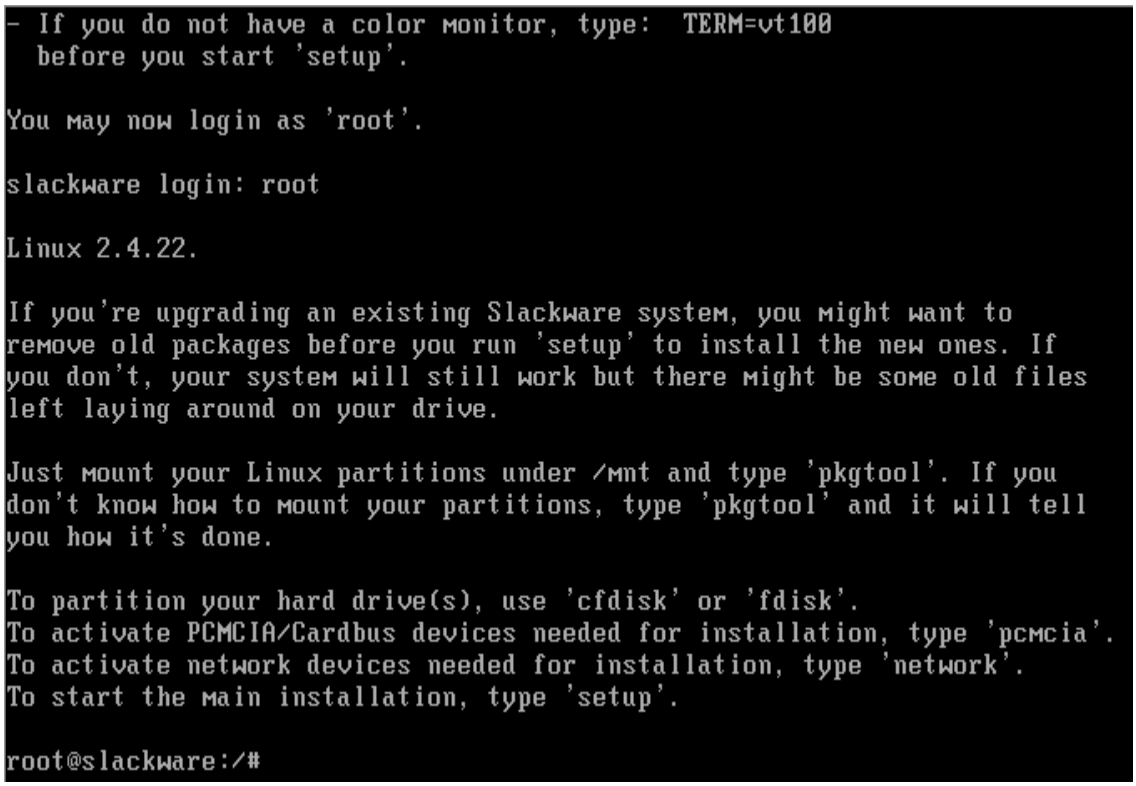
El CD 1 de Slackware es de arranque (en caso de que se haya descargado de Internet, se necesitan traducir todas las ISOs a CDs). En caso de que la computadora destino sea incapaz de arrancar desde el CD ROM por alguna razón, es posible crear un floppy de arranque listo para la instalación. Esto se logra con las aplicaciones que se encuentran en el directorio bootdisks del CD 1.

Una vez que arranca el medio de instalación adecuado, aparece esta primera pantalla:



Muchos casos con sistemas IDE normales pueden simplemente teclear ENTER aquí, para cargar la imagen de kernel bare.i. Sin embargo, si se tienen discos SCSI o SATA, es necesario averiguar la imagen correcta que contenga los controladores correctos; esto se puede leer en el archivo README.TXT en el directorio bootdisks del CD. Las imágenes son: adaptec.s, scsi.s, scsi2.s y scsi3.s, cada una para controladores diferentes.

De cualquier forma, el kernel va a arrancar, y después de un momento, pedirá que se ingrese como root.



Tan solo se necesita teclear root y presionar ENTER. Debido a que no existe una contraseña en este momento, no pedirá ninguna.

Ahora sigue particionar el disco. Probablemente esta sea la parte que más confunde a un nuevo usuario acerca de la instalación, no sólo de Slackware, sino de cualquier distribución de Linux. Utilizar la utilidad incluida Fdisk al principio puede parecer intimidante, pero es fácil de operar y es raro que se cometan errores si se siguen los pasos correctamente.

En este ejemplo, se está utilizando un disco duro de 40 Gb. Tiene asignada una partición de 8 Gb con formato NTFS para Windows XP. El resto del disco está completamente sin formato.

Fdisk debe de ser invocado con el nombre de dispositivo del disco duro que se necesita particionar. En este caso, se está usando el disco primario maestro, así que se utiliza el nombre /dev/hda. Aquí está la lista de cómo funcionan los nombres de discos IDE:

/dev/hda	–	Primario	Maestro
/dev/hdb	–	Primario	Esclavo
/dev/hdc	–	Secundario	Maestro
/dev/hdd	–	Secundario	Esclavo

Cabe notar que estos nombres NO se refieren a particiones o sistemas de archivos, sino a los dispositivos de discos duros que están conectados a la tarjeta madre.

Los discos SCSI se llaman /dev/sda, /dev/sdb, etc. de acuerdo a cómo están enumerados en el bus.
En este caso, se necesita escribir fdisk /dev/had

```
bash-2.05b# fdisk /dev/hda

The number of cylinders for this disk is set to 4865.
There is nothing wrong with that, but this is larger than 1024,
and could in certain setups cause problems with:
1) software that runs at boot time (e.g., old versions of LILO)
2) booting and partitioning software from other OSs
   (e.g., DOS FDISK, OS/2 FDISK)

Command (m for help):
```

El mensaje sobre el número de cilindros se refiere a un problema con discos grandes que se presentaba en programas de arranque maestro muy antiguos. A menos que se utilice una distribución de Linux muy antigua, no tiene importancia.

Si se presiona m, aparece una lista de comandos.

```
Command (m for help): m
Command action
   a   toggle a bootable flag
   b   edit bsd disklabel
   c   toggle the dos compatibility flag
   d   delete a partition
   l   list known partition types
   m   print this menu
   n   add a new partition
   o   create a new empty DOS partition table
   p   print the partition table
   q   quit without saving changes
   s   create a new empty Sun disklabel
   t   change a partition's system id
   u   change display/entry units
   v   verify the partition table
   w   write table to disk and exit
   x   extra functionality (experts only)

Command (m for help):
```

Lo primero que se tiene que hacer es presionar p para imprimir en pantalla la tabla de partición. Es buena costumbre hacer esto después de cada paso, para ver el resultado; Nada cambia realmente, hasta que el programa recibe la orden específica de escribir los cambios a la tabla y salir, con la tecla w.

```
Command (m for help): p

Disk /dev/hda: 40.0 GB, 40020664320 bytes
255 heads, 63 sectors/track, 4865 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hda1   *           1        1044     8385898+   7  HPFS/NTFS

Command (m for help):
```

Se puede ver la partición NTFS* de 8 Gb, /dev/hda1. La primera partición en el disco, y también la activa. No se alterará esta partición en todo el proceso.

Las unidades (en los campos Start y End) son cilindros de 8225280 bytes. Cada unidad es aproximadamente 8 Megabytes (7.84 exactamente). También se presentan en bloques de aproximadamente 1 Kb. Sin embargo, para facilitar el proceso, es posible escribir las cantidades en Megas.

Como repartir las particiones depende de las preferencias personales. Todo lo que se necesita para instalar y correr Linux es una partición root, y una partición swap. Es perfectamente aceptable arreglarse así. Sin embargo, se le estará asignando una gran parte del disco, y es posible montar partes del sistema de archivos Linux en particiones separadas.
Siguiendo lo que se acostumbra en estos casos, la regla general es que si existen problemas de espacio en el disco, solamente se crea la partición root, y la swap. Por ejemplo, si se tienen 4 Gb de espacio, una partición root de 3.7 Gb es ideal, y se utiliza el resto para swap.
Utilizar muchas particiones puede ser un derroche, particularmente porque se tiene que dejar espacio en cada partición para que pueda crecer. Esto puede resultar en mucho espacio de disco que no se utilice. Es mejor prevenir y dejar mucho espacio disponible.

En este caso en particular, se va a particionar así:

- Una partición root de 1 Gb (primaria).- el sistema de archivos de root contiene software y librerías, datos de configuración, de estado local, y todos los demás sistemas se montan bajo él.

Una partición extendida utilizando el resto del disco.- después se crean las particiones siguientes en forma lógica.

- 1Gb de partición swap (lógica).- probablemente no se necesite una swap tan grande como esta, pero ya que se cuenta con mucho espacio, es recomendable asegurarse. Esto permite trabajar con archivos enormes, y provee la memoria extra en el caso de que se necesite una sesión muy rápida. Normalmente es suficiente asignar 256 Mb.

- 8 Gb de partición para /usr (lógica).- casi todo el software y librerías se instalan en /usr. Es muy útil tener mucho espacio para esto.

- 2 Gb de partición para /opt (lógica).- el software "opcional" se instala en /opt. Por ejemplo, KDE va en este directorio.

- 18 Gb de partición para /home (lógica).- el resto se usa para /home. Aquí es donde se encuentran los directorios de usuarios, y donde se almacenan los archivos personales. También se puede instalar algún software en /home si se desea.

** Aunque el formato NTFS es soportado por Linux, no es muy recomendable usarla para algo más que simple lectura. El sistema de archivos es privado de Microsoft, y si otro sistema operativo tratara de escribir en el mismo, la partición se destruiría.*

Ahora se crearán estas particiones.

Para crear una partición nueva, se presiona n.

```
   Device Boot      Start         End      Blocks   Id  System
/dev/hda1   *           1        1044     8385898+   7  HPFS/NTFS

Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
```

El sistema pregunta si se desea una primaria o extendida. Se necesita crear una partición primaria.

La tecla p es para crearla

```
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 2
First cylinder (1045-4865, default 1045):
Using default value 1045
Last cylinder or +size or +sizeM or +sizeK (1045-4865, default 4865): +1024M
```

Después se le tiene que asignar un número de partición. Ya que la partición de Windows XP es la 1, el número 2 es el indicado.

Cuando el sistema pregunta por el cilindro inicial, tan sólo se acepta el valor por defecto, que es el primer cilindro libre. Se acepta siempre este valor defecto para todas las particiones que se crearán a continuación. Se especifica el cilindro final, escribiendo el tamaño deseado de la partición en megabytes. Para el valor de "last cylinder", se escribe +1024M para crear una partición de un Gb. Las particiones tienen que terminar en un límite de cilindros (o sector basura), y el software de particionamiento siempre se ajusta a eso.

```
Command (m for help): p

Disk /dev/hda: 40.0 GB, 40020664320 bytes
255 heads, 63 sectors/track, 4865 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hda1   *           1        1044     8385898+   7  HPFS/NTFS
/dev/hda2            1045        1169     1004062+  83  Linux

Command (m for help):
```

Ahora se presiona p para checar nuevamente la tabla de partición. Aquí es posible ver lo que se ha hecho hasta ahora. Si hasta este punto hay algún error, simplemente se presiona d y el número de partición que se desee borrar (no la primera, ya que ahí se encuentra el segundo sistema operativo). Nada ha sido escrito a la tabla aún, así que simplemente se puede borrar la partición que se acaba de crear y repetir el último paso. Por esta razón es buena costumbre ver cada cambio en la tabla de particiones. En cualquier momento se puede presionar q en la línea de comandos para salir sin escribirle nada al disco. Una vez satisfechas las necesidades, se puede continuar.

Ahora se crea la partición extendida, para actuar como un contenedor para los espacios lógicos.

```
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
e
Partition number (1-4): 3
First cylinder (1170-4865, default 1170):
Using default value 1170
Last cylinder or +size or +sizeM or +sizeK (1170-4865, default 4865):
Using default value 4865

Command (m for help): p

Disk /dev/hda: 40.0 GB, 40020664320 bytes
255 heads, 63 sectors/track, 4865 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hda1   *           1        1044     8385898+   7  HPFS/NTFS
/dev/hda2            1045        1169     1004062+  83  Linux
/dev/hda3            1170        4865    29688120    5  Extended

Command (m for help):
```

Se teclea n para crear una partición nueva, y después e para escoger extendida. 3 es el número de partición, y será designada como /dev/hda3. A esta partición en particular nunca se accederá, sino a los espacios que se contienen en ella.

Nota: los números de partición están reservados para particiones primarias. Es una limitación en la arquitectura de las tablas de partición, que solamente 4 sean aceptadas en cada disco. Sin embargo, es posible tener muchos discos lógicos. Estos se comienzan a enumerar desde el 5 en adelante en el esquema de Linux.

Cuando el sistema pregunte por el último cilindro, esta vez simplemente se presiona ENTER. Esto asignará el resto del disco a esta partición, y terminará en el último cilindro, el 4865.

Ahora se crean espacios lógicos hasta llenar la partición extendida, empezando por swap. Es recomendable poner la partición swap entre /root y /usr.

```
Command (m for help): n
Command action
   l   logical (5 or over)
   p   primary partition (1-4)
l
First cylinder (1170-4865, default 1170):
Using default value 1170
Last cylinder or +size or +sizeM or +sizeK (1170-4865, default 4865): +1024M

Command (m for help): p

Disk /dev/hda: 40.0 GB, 40020664320 bytes
255 heads, 63 sectors/track, 4865 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hda1   *           1        1044     8385898+   7  HPFS/NTFS
/dev/hda2            1045        1169     1004062+  83  Linux
/dev/hda3            1170        4865    29688120    5  Extended
/dev/hda5            1170        1294     1004031   83  Linux

Command (m for help):
```

Se crea una nueva partición, pero esta vez presionando l para especificar que es lógica (en este caso no se pueden tener más primarias porque ya se asignó todo el espacio en el disco).

Hay que notar que no pide el sistema por el número de partición, ya que automáticamente le asignará el 5 por ser la primera partición lógica.
Se crea una partición de 1Gb que comience en el cilindro por defecto.

Cuando se presiona p para ver la tabla, es notorio que la nueva partición se llama /dev/hda5. No existe /dev/hda4 porque no hay más que tres particiones primarias en el disco.

También hay que notar la columna Id en la tabla de particiones. Por defecto, cuando se crean particiones son del tipo 83, o Linux native. Se necesita cambiar la recién creada a tipo 82, Linux swap.

```
Command (m for help): t
Partition number (1-5): 5
Hex code (type L to list codes): 82
Changed system type of partition 5 to 82 (Linux swap)

Command (m for help): p

Disk /dev/hda: 40.0 GB, 40020664320 bytes
255 heads, 63 sectors/track, 4865 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hda1   *           1        1044     8385898+   7  HPFS/NTFS
/dev/hda2            1045        1169     1004062+  83  Linux
/dev/hda3            1170        4865    29688120    5  Extended
/dev/hda5            1170        1294     1004031   82  Linux swap

Command (m for help):
```

Se presiona t para cambiar un “Id” de sistema de partición, y después el número de la partición indicada (en este caso 5). Cuando pregunta el Hex Code (Id), se puede presionar L para ver una lista de todas las posibles particiones que Fdisk puede soportar. El tipo deseado es el 82, Linux swap. Como siempre, el cambio es visible en la tabla de particiones.

El resto de las particiones que se creen serán el default, tipo 83, Linux native.

```
Command (m for help): n
Command action
   l   logical (5 or over)
   p   primary partition (1-4)
l
First cylinder (1295-4865, default 1295):
Using default value 1295
Last cylinder or +size or +sizeM or +sizeK (1295-4865, default 4865):+8192M

Command (m for help): p

Disk /dev/hda: 40.0 GB, 40020664320 bytes
255 heads, 63 sectors/track, 4865 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hda1   *           1        1044     8385898+   7  HPFS/NTFS
/dev/hda2            1045        1169     1004062+  83  Linux
/dev/hda3            1170        4865    29688120    5  Extended
/dev/hda5            1170        1294     1004031   82  Linux swap
/dev/hda6            1295        2291     8008371   83  Linux

Command (m for help):
```

La siguiente partición que se creará es la /usr, a la cual se le asignarán 8 Gb escribiendo +8192M como valor del cilindro final.

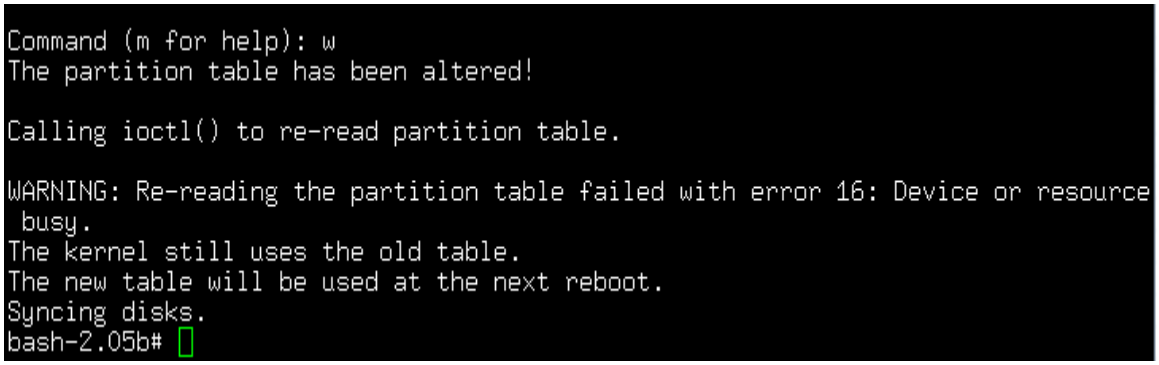
```
Command (m for help): n
Command action
   l   logical (5 or over)
   p   primary partition (1-4)
l
First cylinder (2292-4865, default 2292):
Using default value 2292
Last cylinder or +size or +sizeM or +sizeK (2292-4865, default 4865): +2048M

Command (m for help): p

Disk /dev/hda: 40.0 GB, 40020664320 bytes
255 heads, 63 sectors/track, 4865 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hda1   *           1        1044     8385898+   7  HPFS/NTFS
/dev/hda2            1045        1169     1004062+  83  Linux
/dev/hda3            1170        4865    29688120    5  Extended
/dev/hda5            1170        1294     1004031   82  Linux swap
/dev/hda6            1295        2291     8008371   83  Linux
/dev/hda7            2292        2541     2008093+  83  Linux

Command (m for help):
```

La siguiente partición es /opt, con un valor de 2 Gb. Y finalmente, la última es /home.

```
Command (m for help): n
Command action
   l   logical (5 or over)
   p   primary partition (1-4)
l
First cylinder (2542-4865, default 2542):
Using default value 2542
Last cylinder or +size or +sizeM or +sizeK (2542-4865, default 4865):
Using default value 4865

Command (m for help): p

Disk /dev/hda: 40.0 GB, 40020664320 bytes
255 heads, 63 sectors/track, 4865 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hda1   *           1        1044     8385898+   7  HPFS/NTFS
/dev/hda2            1045        1169     1004062+  83  Linux
/dev/hda3            1170        4865    29688120    5  Extended
/dev/hda5            1170        1294     1004031   82  Linux swap
/dev/hda6            1295        2291     8008371   83  Linux
/dev/hda7            2292        2541     2008093+  83  Linux
/dev/hda8            2542        4865    18667498+  83  Linux

Command (m for help):
```

Esta vez se puede utilizar el valor default, ya que se está utilizando todo el espacio restante en la partición extendida y en el disco.

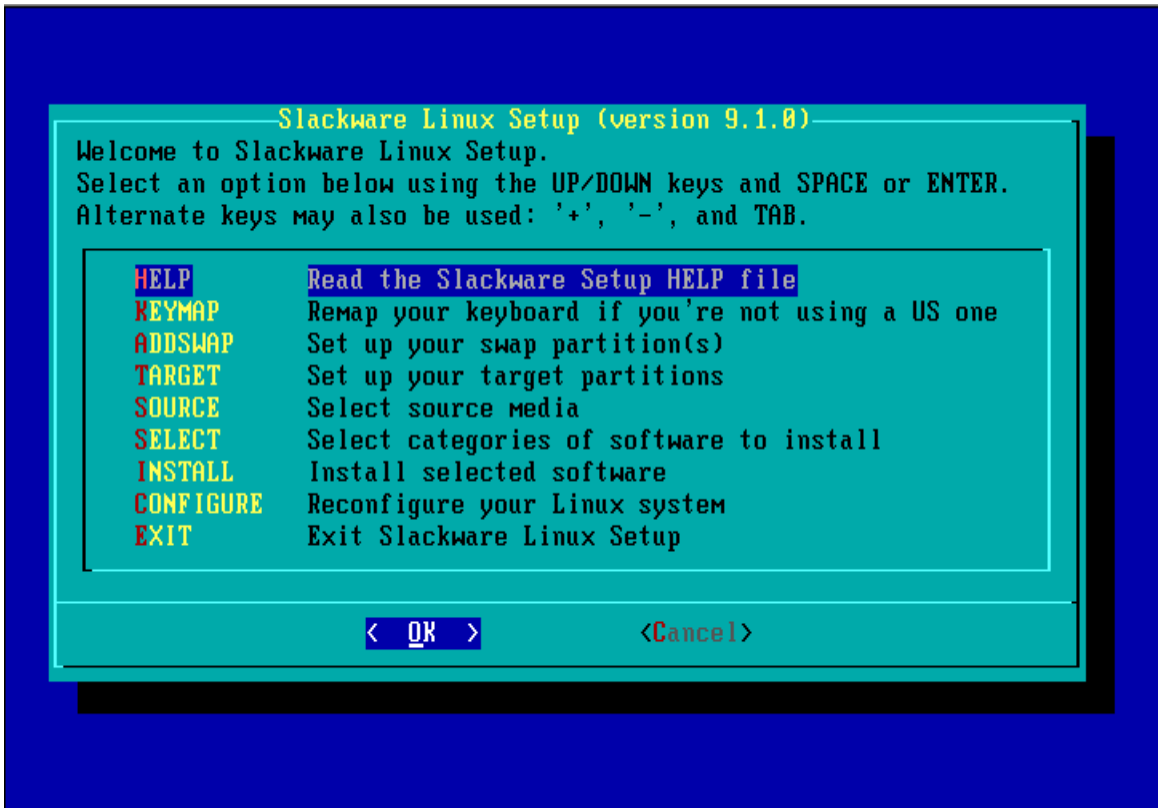
Cuando los cambios estén completos, se teclea w para escribir la tabla al disco, y salir de Fdisk.



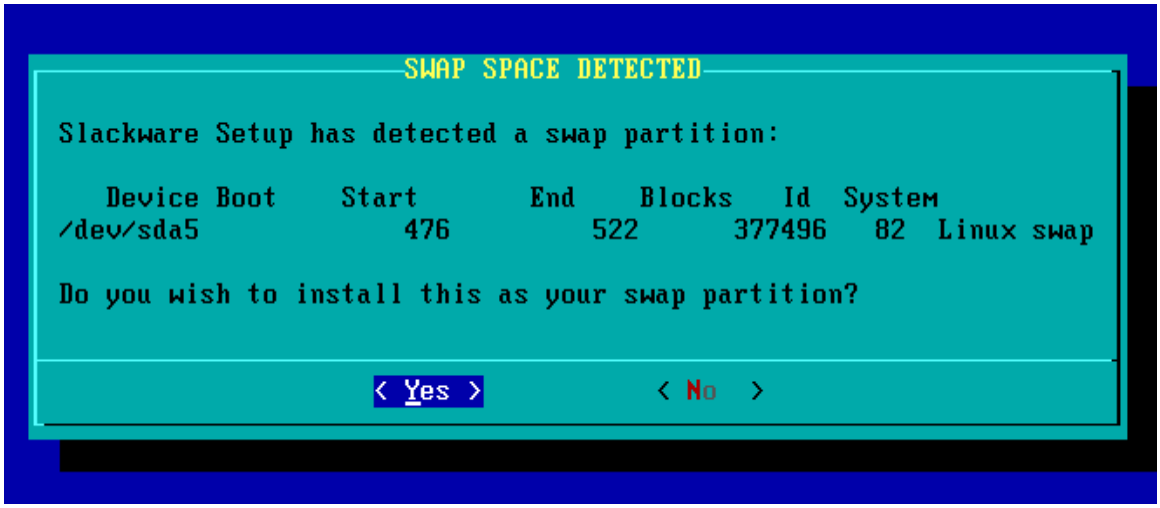
Si aparece un mensaje de emergencia como este, el sistema necesita ser reiniciado (con el CD de arranque). Esto generalmente no sucede, a menos que se haya reescrito la tabla varias veces. Solamente es necesario reiniciar si aparecen mensajes de emergencia.

Hay que tomar nota de que dispositivos creados corresponden con los puntos de montaje, porque hay que especificarlos en la instalación, la cual es el paso siguiente.

Ahora se teclea setup en la línea de comandos.



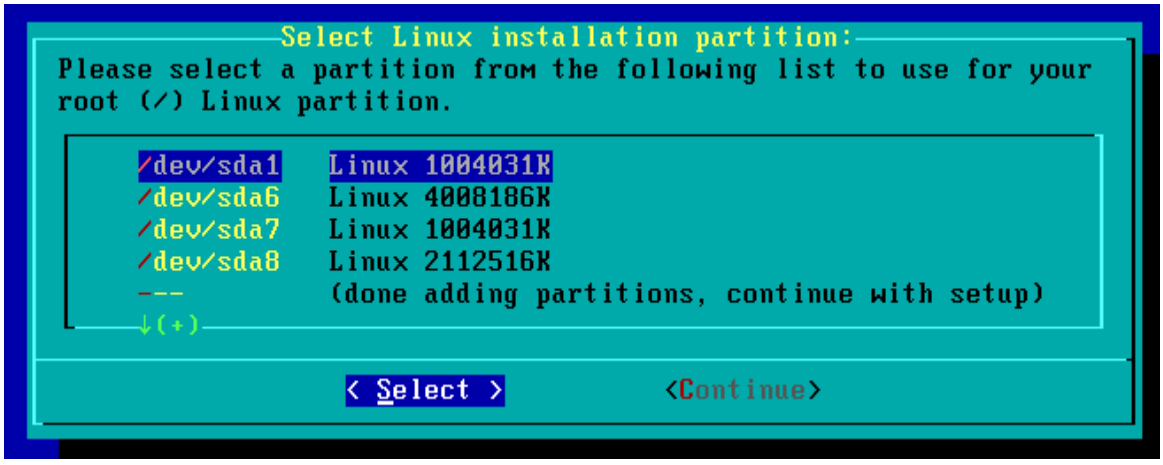
Este es el menú principal de setup. Se puede leer el archivo de ayuda desde aquí si se desea, aunque no es necesario, así como tampoco es necesario cambiar el perfil del teclado, a menos que se necesite utilizar un teclado en español. En todo caso, la instalación comienza cuando se selecciona ADDSWAP.



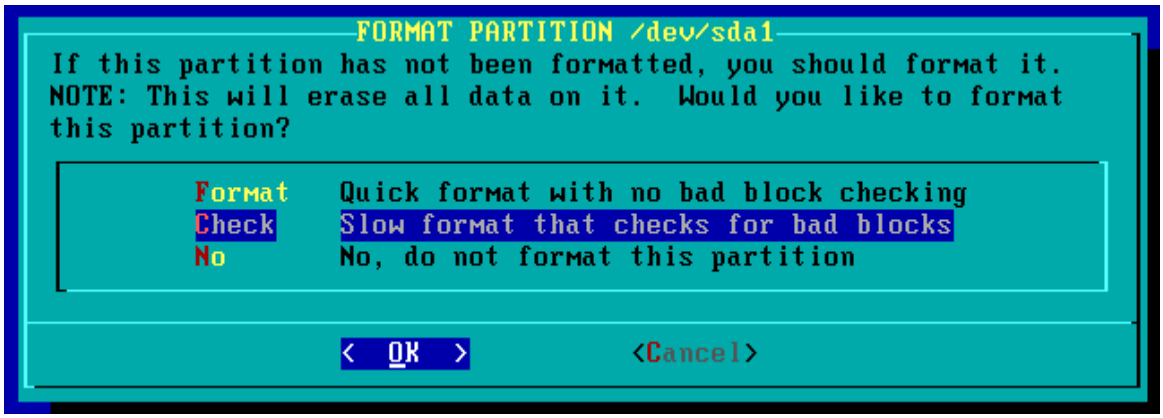
El asistente detecta la partición swap, la formatea y la activa (utilizando internamente los comandos mkswap y swapon).

NOTA: el disco que aparece en las imágenes es llamado /dev/sda. Esto se debe a que para obtener las capturas de pantalla, se utilizó una máquina virtual, que emuló este disco como un SCSI.

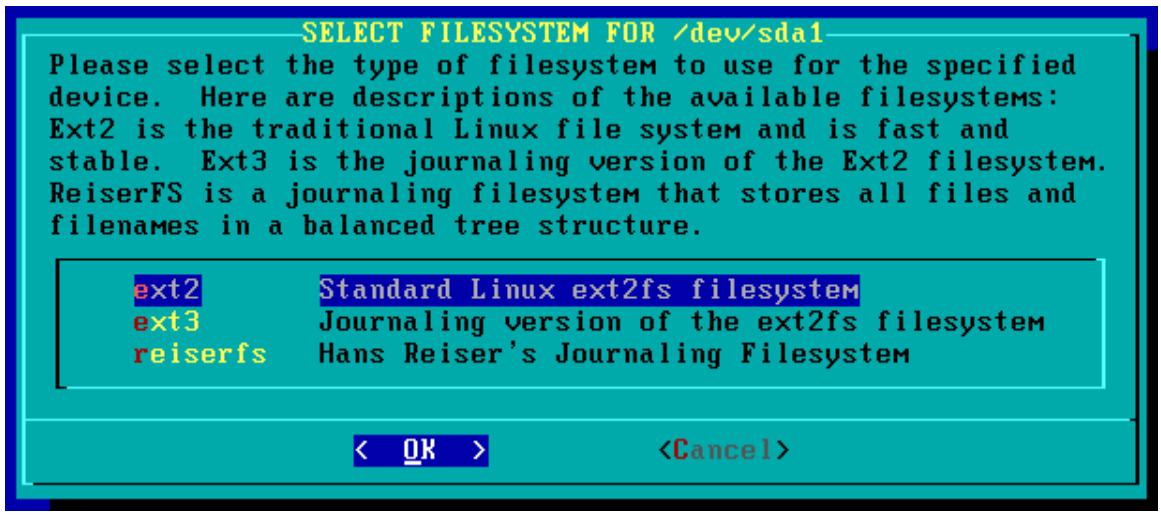
Después de completar un paso, el asistente automáticamente pasa al siguiente en la secuencia. En este caso, es seleccionar las particiones destino. Aquí es donde se selecciona el root, y luego los puntos de montaje para las demás.



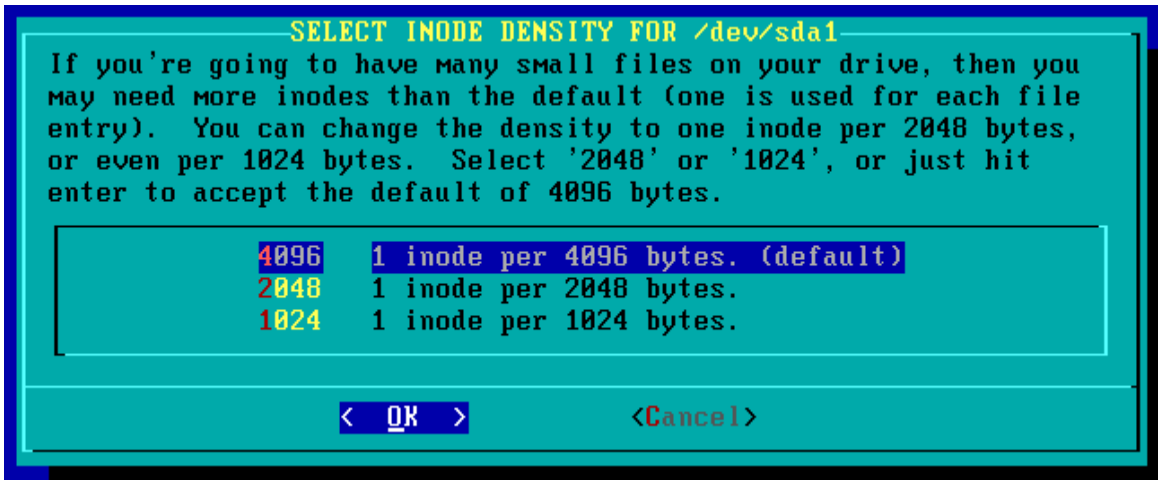
Ahora el sistema preguntará si se desea formatear la partición. Si es la primera vez que se va a hacer (si no es una reinstalación) es recomendable checar los bloques.



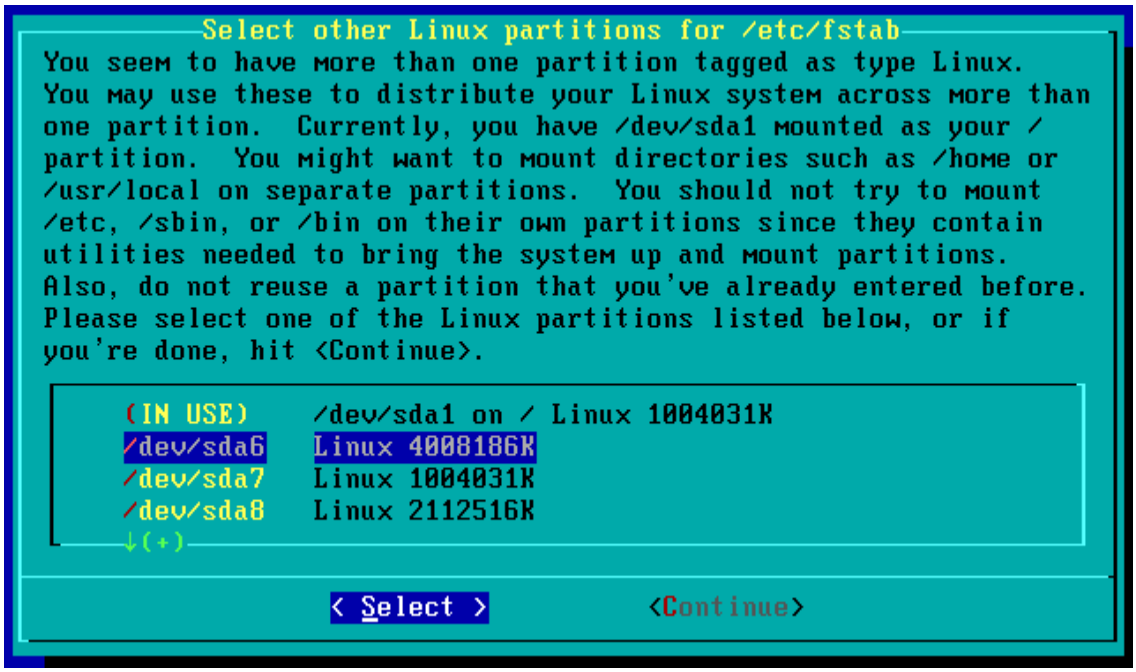
Se selecciona el sistema de archivos deseado. Ext2 es una buena elección, es simple y está muy probado, aunque si se van a utilizar bitácoras, o modelos de RAID en discos SATA o SCSI, se necesitará Ext3 o Reiserfs.



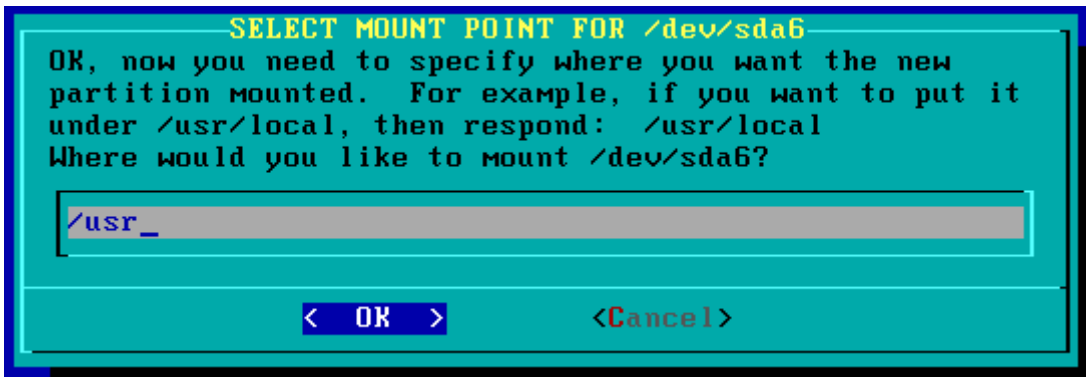
Finalmente se pregunta por la densidad del sistema de archivos. Tan solo se presiona ENTER para seleccionar el defecto de 4096, a menos que haya una razón específica para hacerlo diferente.



Si tan sólo se hubiera creado una partición root y swap, se habría terminado de formatear. Si se crean otras, se deben seleccionar de la siguiente lista y asignarles puntos de montaje.

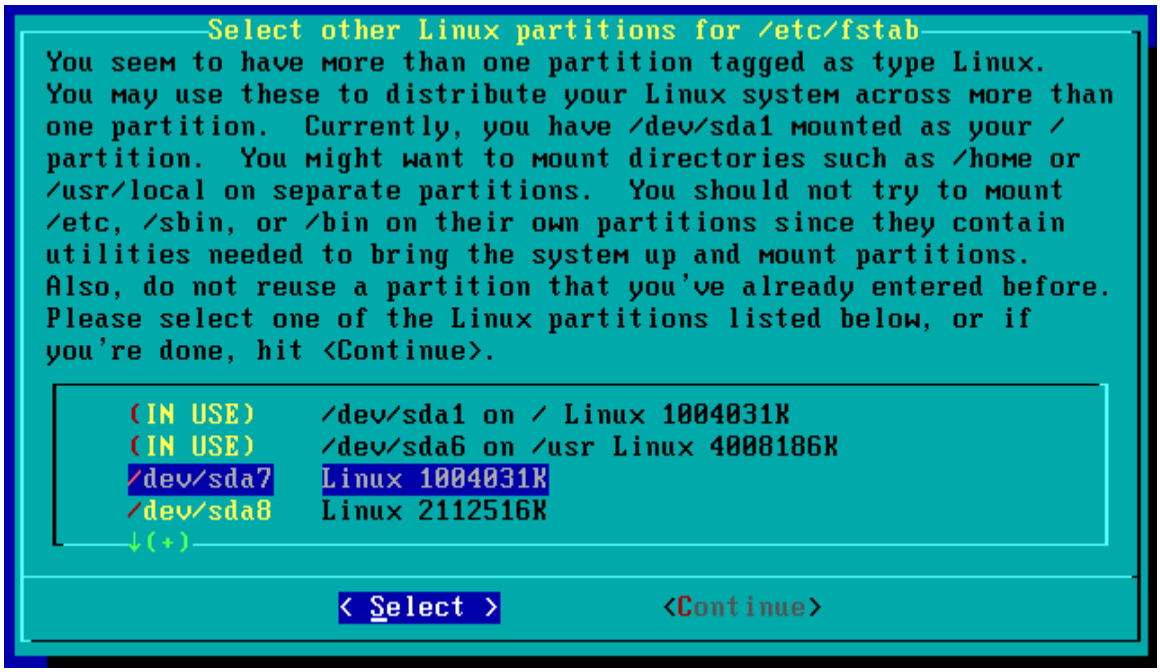


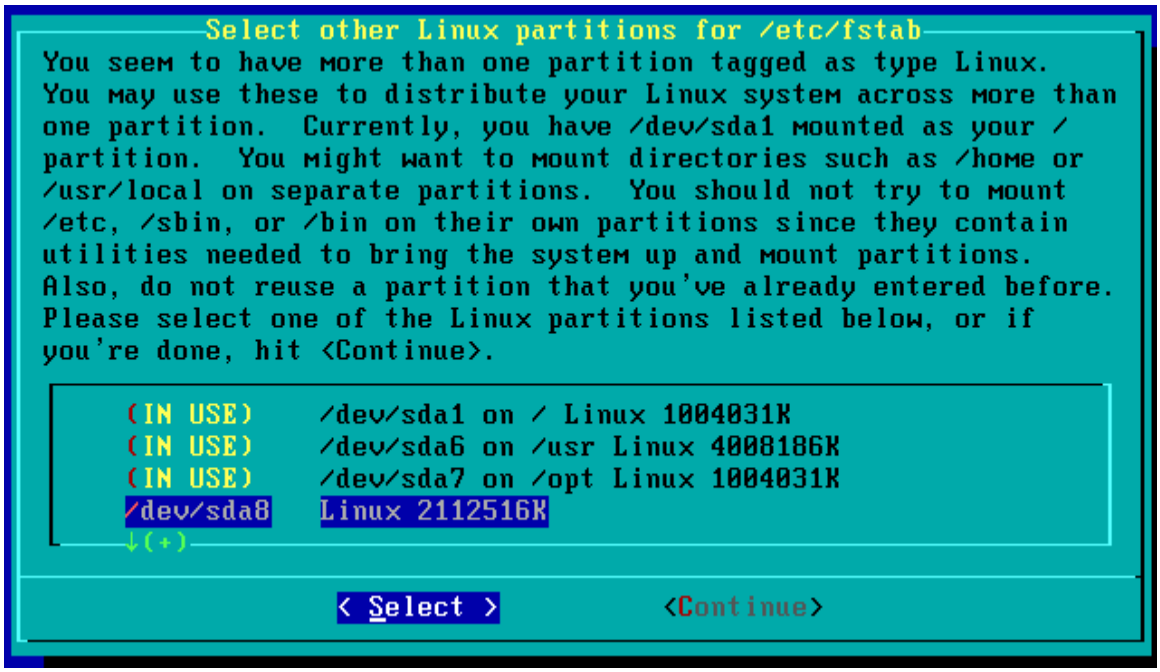
La partición swap no aparece en esta lista.



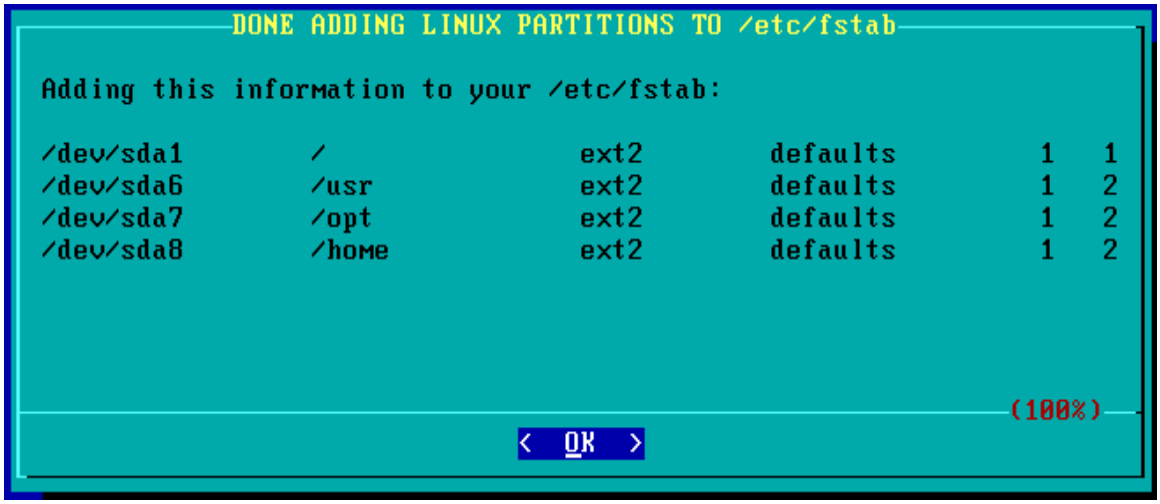
Se selecciona la primera partición, y se escribe su punto de montaje, en este caso /usr.

Continuando, se selecciona y monta de una por una las particiones de la misma forma.

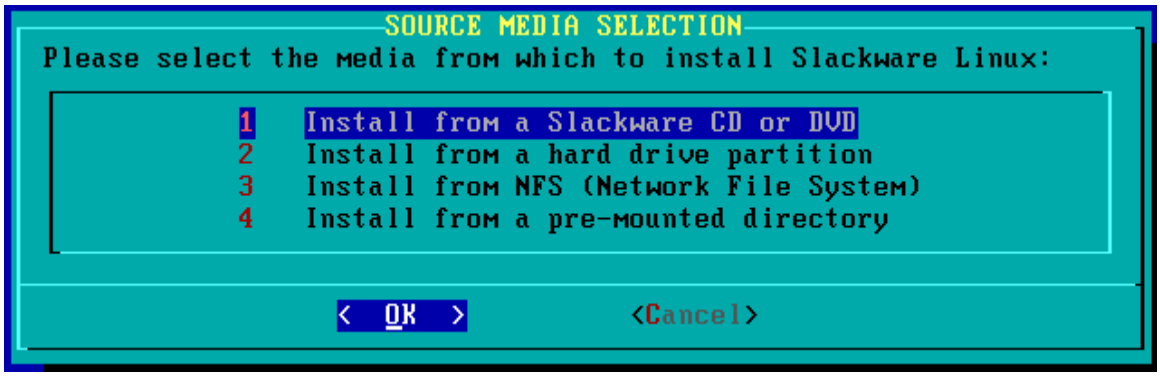




Al final, aparece un sumario de las elecciones.

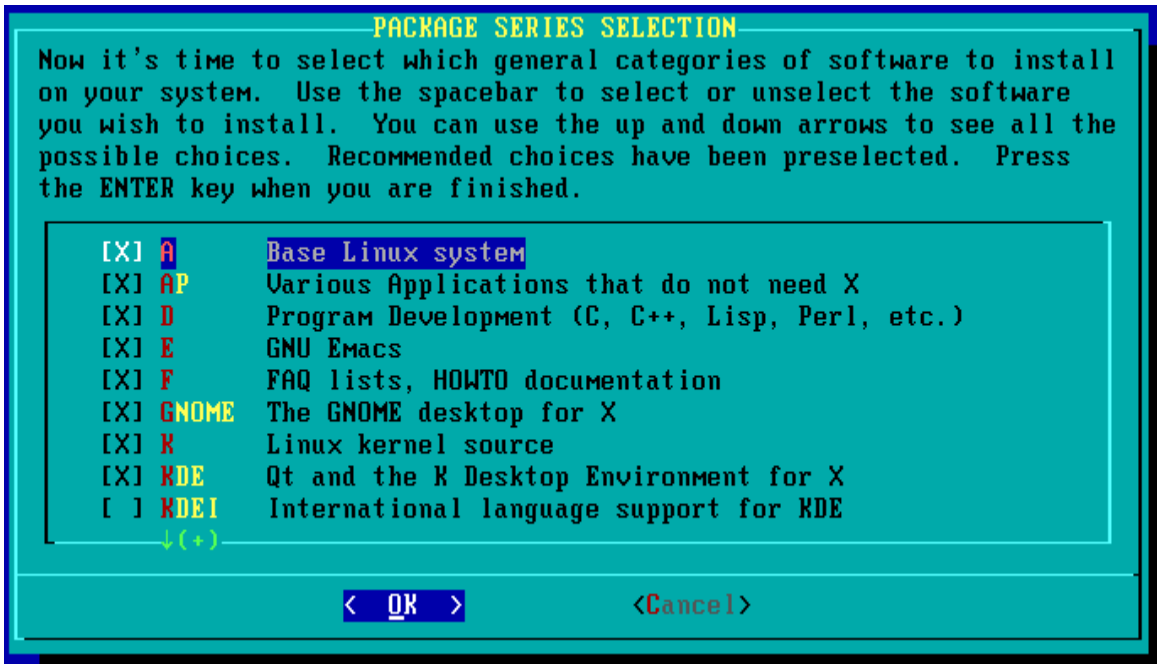


En el siguiente paso pregunta el asistente cuál es el medio fuente.



Cuando se selecciona CD ROM, lo debe detectar automáticamente.

Y se seleccionan categorías de paquetes.



Estas gobiernan que series de paquetes serán instalados en el sistema. Por default, todas las categorías son seleccionadas excepto KDEI (la internacionalización de KDE). Así que si se prefiere que KDE esté presentado en un idioma que no utilice el alfabeto occidental, se tiene que activar esta casilla.

Generalmente es preferible instalar todos los paquetes por defecto, así que se presiona ENTER para que se presente la selección de "prompting", o más específicamente, que tanto hay que interactuar con la instalación de paquetes.

Full, o completa, instala todos los paquetes en las categorías que se seleccionan sin preguntar. Esto es lo más recomendable la primera vez que se instala Slackware. Después se puede fácilmente remover paquetes que no se deseen. Esta es la opción más sencilla.

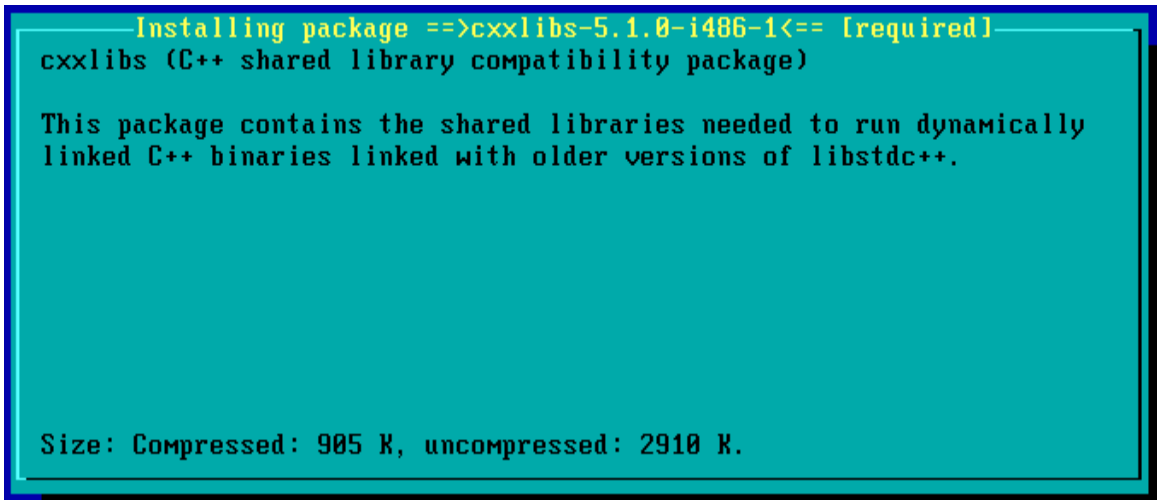
Newbie, o novato, pregunta antes de instalar cada uno de los paquetes. No es muy recomendable, y es bastante tediosa. Además de que algunas veces no es seguro si se necesita algo.

Menu es un poco mejor, ya que permite escoger grupos de cosas relacionadas.

El modo experto permite escoger exactamente que paquetes se desea instalar antes de comenzar la instalación en sí. Esto es muy lineal y sencillo, a diferencia de muchas otras distribuciones.

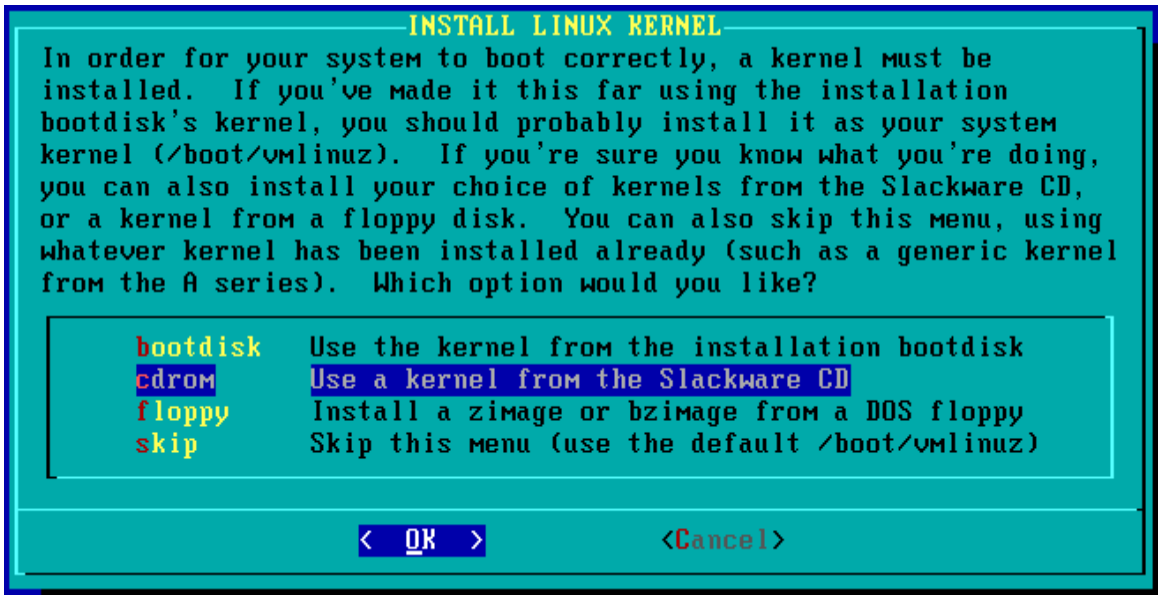
Custom, o personalizado, y tagpath utilizan archivos etiqueta para automatizar la selección de paquetes. Esto es útil si se quisiera hacer la misma instalación en muchas máquinas diferentes.

Se selecciona Full, y los paquetes se instalarán sin preguntar nada. No tarda mucho este proceso, inclusive en máquinas lentas.

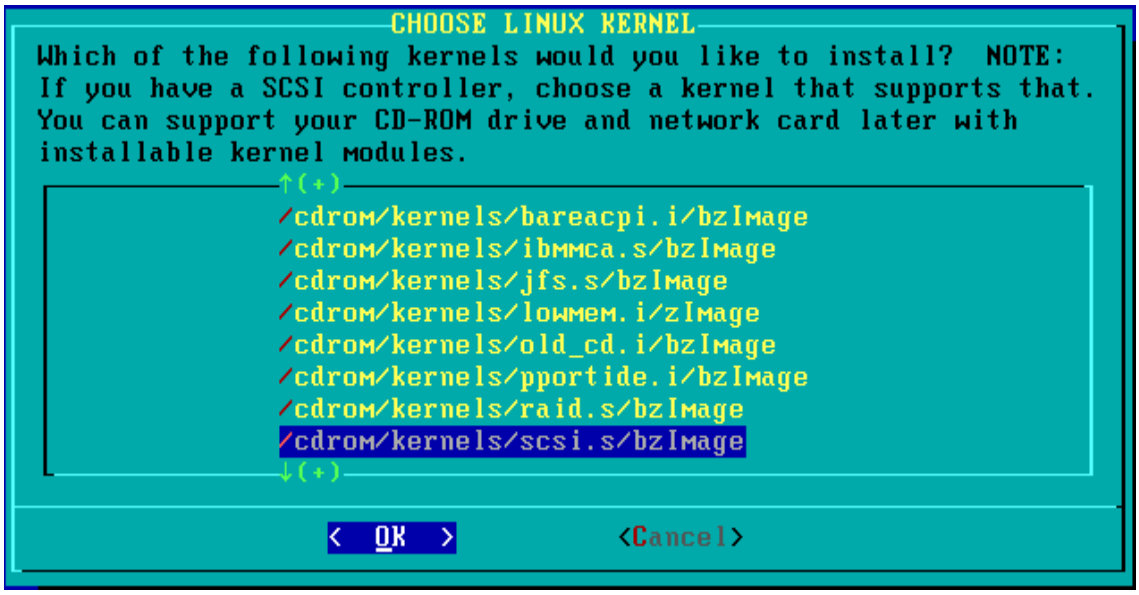


En algún momento de la instalación, el asistente pedirá el Segundo CD.

Cuando se completa la instalación de paquetes, el asistente require de un kernel.

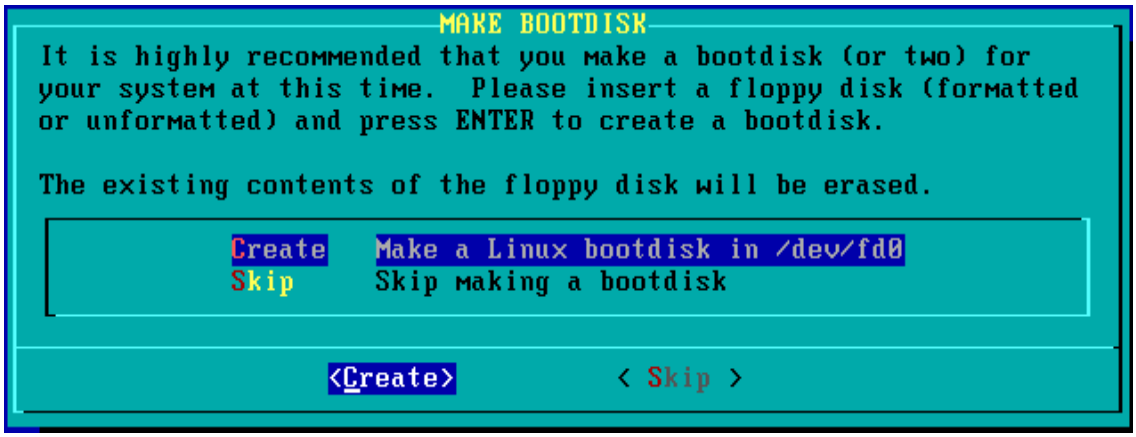


Lo ideal es seleccionar la opción de CD ROM, y el mismo kernel que se usó para el arranque inicial (el default es bare.i).



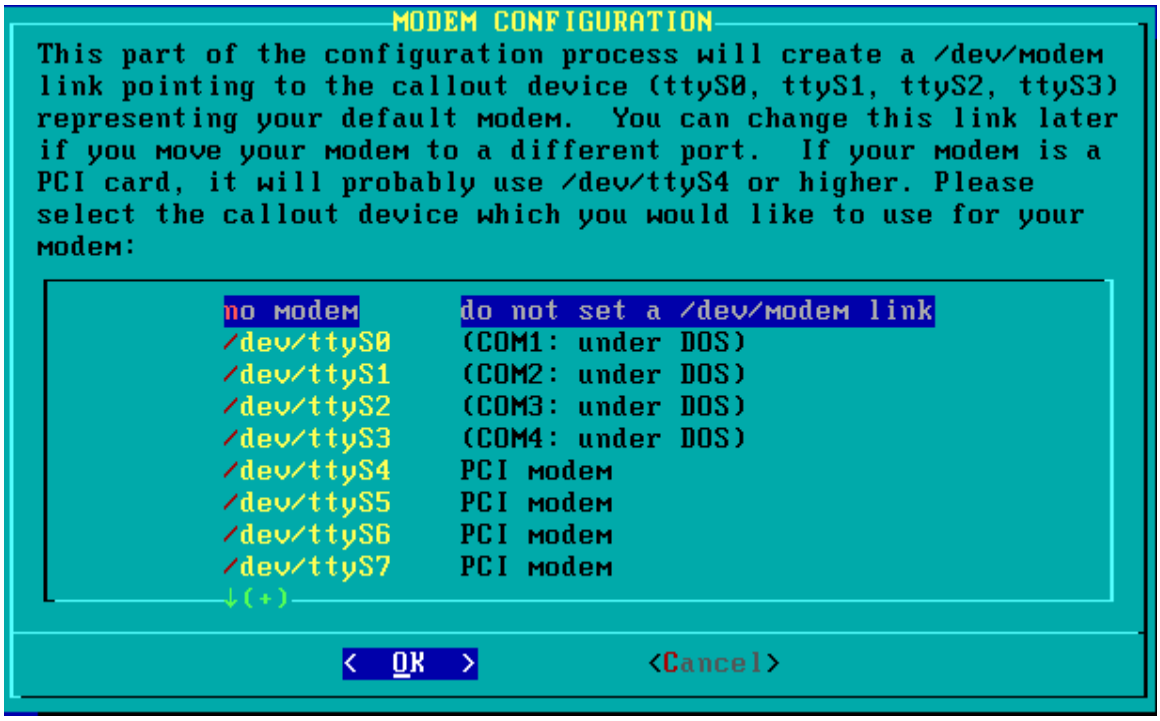
Nota: por el hecho de haber utilizado una máquina virtual, se tuvo que escoger el kernel SCSI.s, pero la mayoría de los sistemas IDE utilizan el estándar bare.i.

Ahora el asistente desea que se cree un floppy de arranque.



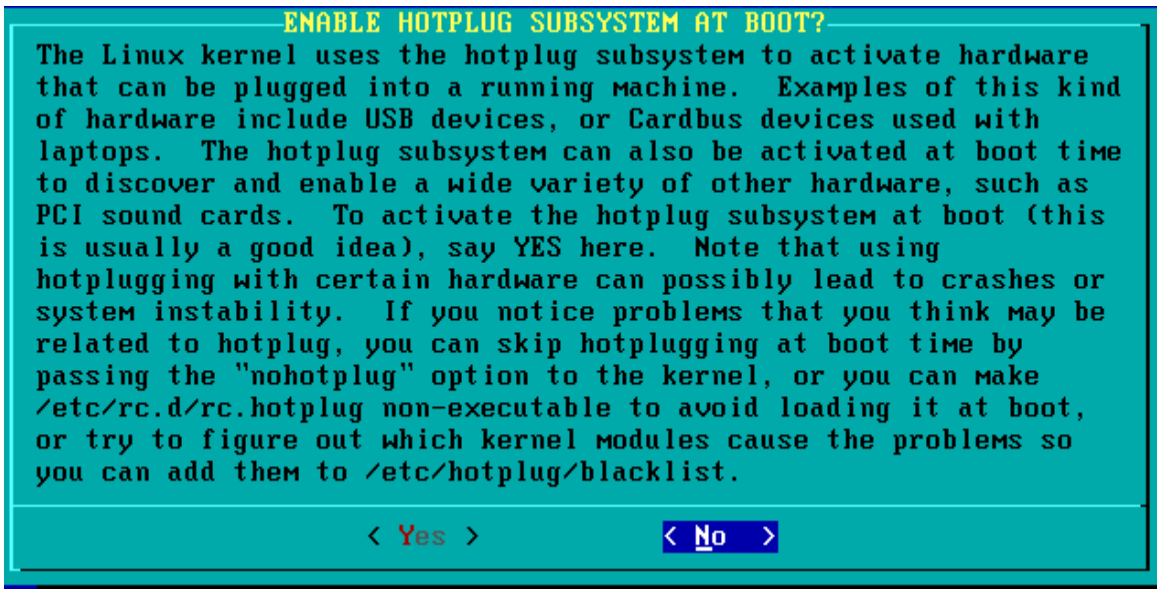
Es altamente deseable crear este disco, ya que puede ser usado para arrancar la distribución si algo pasa con el loader. Utilizando este floppy, es fácilmente rescatable todo el sistema.

El modem necesita una liga simbólica.



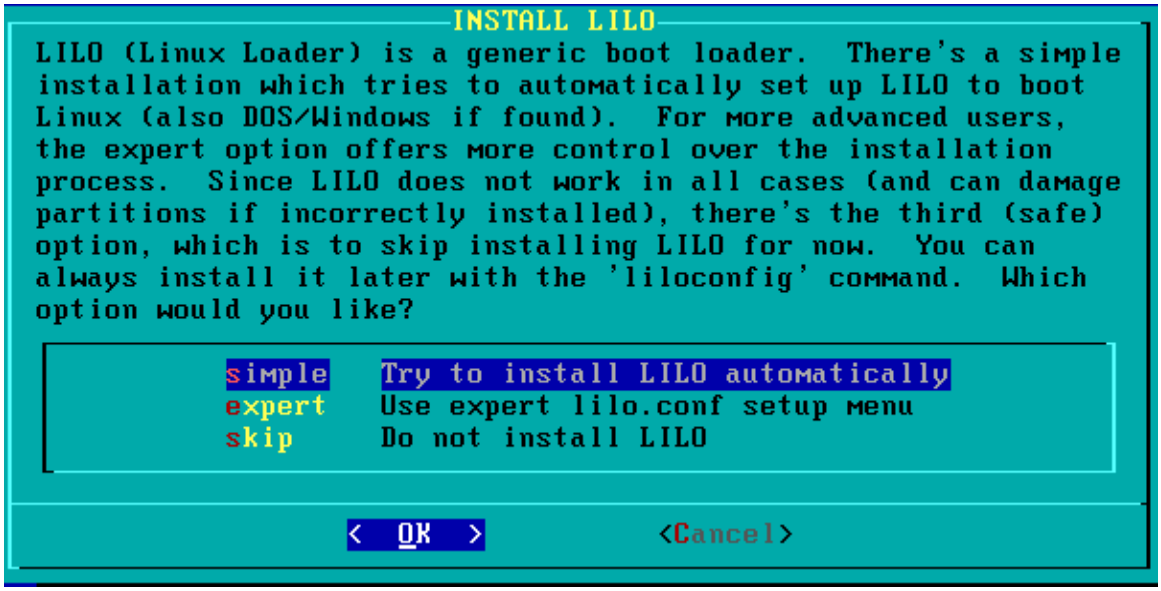
Si se tiene un modem, aquí es donde se declara. Aunque decir en este paso que no existe un MODEM no quiere decir que nunca se pueda instalar, se puede hacer la liga posteriormente, o simplemente utilizar el dispositivo adecuado (/dev/ttyS1 por ejemplo, apunta a COM2).

Ahora es necesario activar el sistema hotplug. Si se tienen dispositivos así, se responde afirmativamente. No es buena idea activar esta opción si no hay hotplugs presentes.



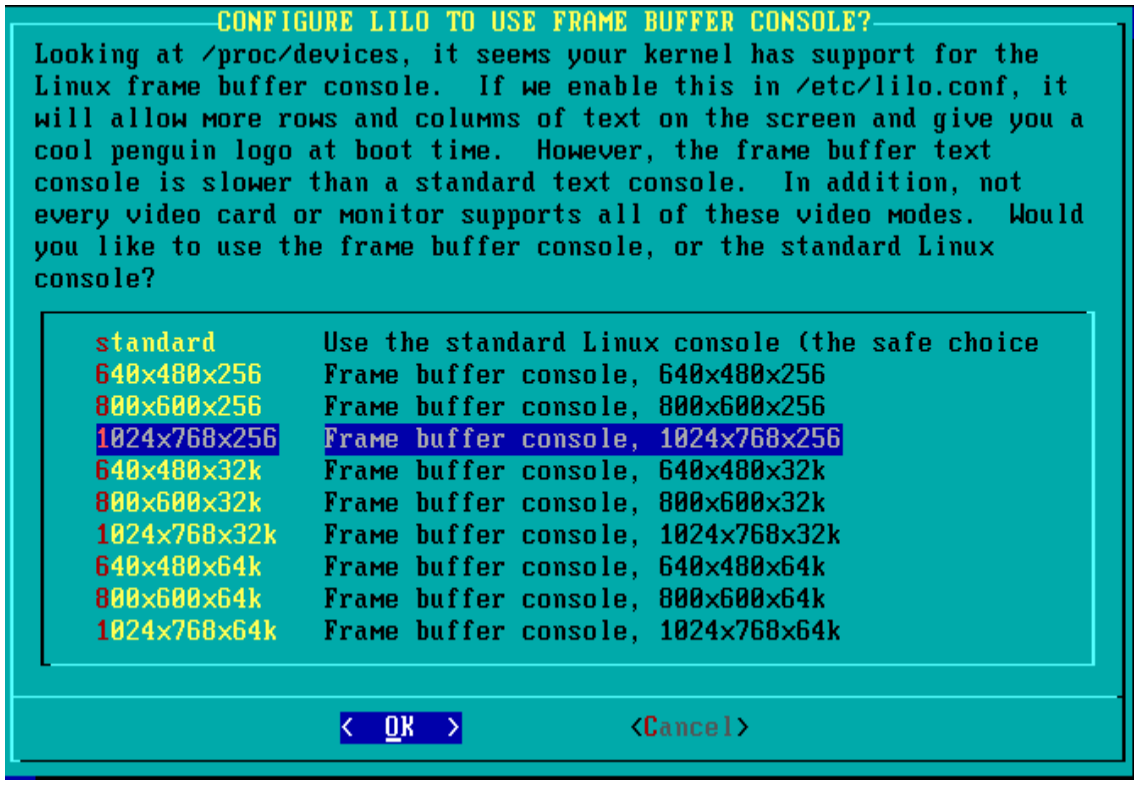
Como se puede leer en esta pantalla, es posible que cause problemas en algunos sistemas. La pantalla también indica como desactivarlo si esto sucede.

El paso siguiente es el loader de Linux: LILO.

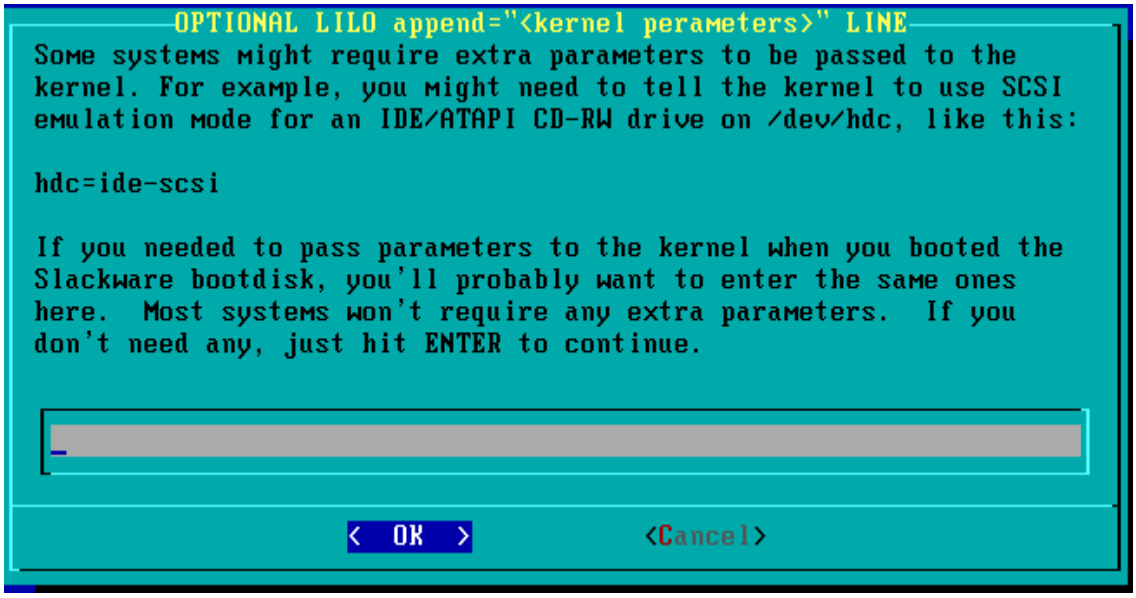


Se selecciona la opcion simple; escoger expert resultaría en tener que modificar el archivo lilo.conf para obtener la funcionalidad deseada. Si se optara por saltar todo este proceso, entonces solamente se podría arrancar Slackware usando el disco de arranque creado en los pasos anteriores.

Cuando pregunte el asistente por un modo VGA (video) para la consola, ya sea por defecto, o alguno de los modos VESA, conviene seleccionar el VGS por defecto, ya que si sucediera algún problema, no se podría ver la pantalla. Como siempre, es posible cambiar esto después.



Ahora se pregunta si se desea introducir parámetros extra para que LILO los pase al kernel.



Imprime un ejemplo muy común de porque se querría hacer esto: si existe un quemador IDE, en la serie 2.4 de kernels, la escritura de CD utiliza una emulación de SCSI y el kernel debe de saber que dispositivo utiliza ese modo.

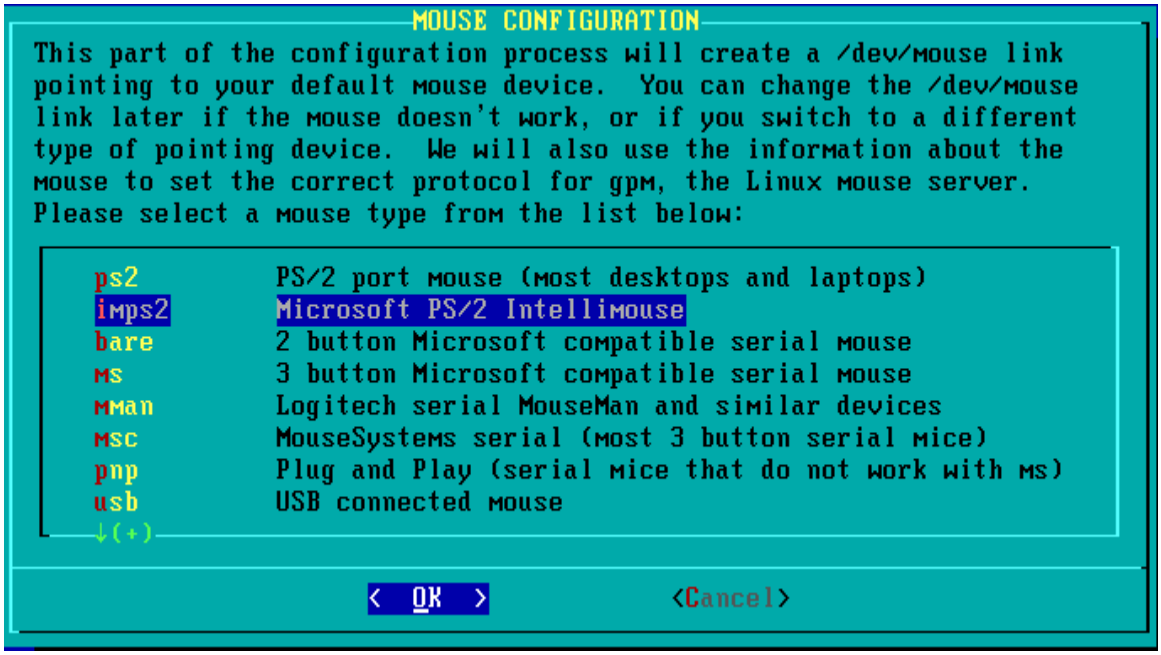
A menos que existan condiciones muy particulares, se deberá escoger el MBR (Master Boot Record) como destino para la instalación de LILO.

Se ignora la advertencia sobre ser posiblemente inseguro al escribir en el MBR; esta se debe a que en ciertas ocasiones, escribir en el MBR es efectivamente inseguro. Por ejemplo, si el BIOS no soporta la capacidad del drive, y existe software de traducción

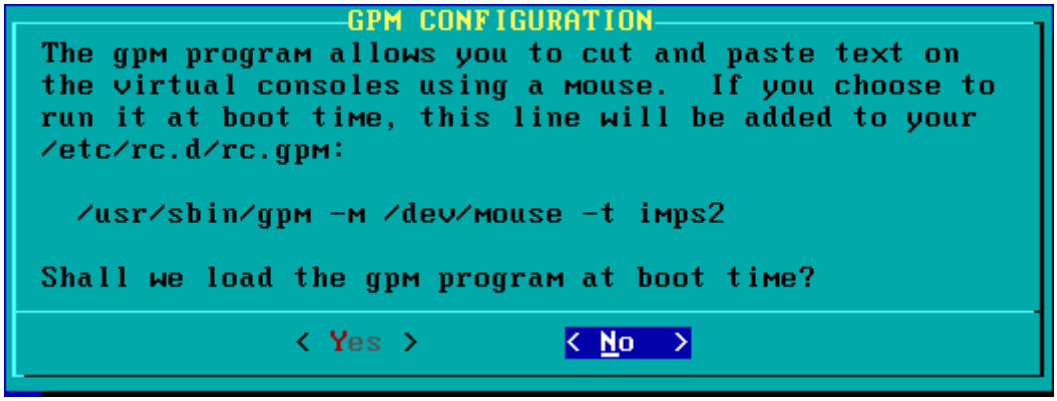
instalado (como MaxBlast o EZBios). Otra razón podría ser si se está utilizando otro boot loader (como System Commander o Boot Magic).
Antes de empezar a escribir nada en el MBR (antes de instalar cualquier sistema operativo), hay que asegurarse que la protección del BIOS al MBR está desactivada (a veces llamada boot virus protection, o Trend Chipaway). Sin embargo, instalar LILO en el MBR es muy común, y muy seguro.

La opción Root instala LILO al superbloque de la partición root; esto se usa principalmente si se desea tener otro administrador de arranque para invocar a LILO.

Ahora se determina la liga simbólica para el mouse.



Aún si no se utiliza gpm, es útil tener una liga simbólica correcta en /dev/mouse. De esta manera se puede simplemente especificar ese dispositivo cuando se configure XFree86 después de que se instale el sistema operativo. La línea de descripción es bastante explicativa, así que no habrá problemas en escoger la adecuada.



No hay muchos usos para esta ventana (no tiene nada que ver con usar mouse dentro de la interfaz gráfica), pero si se desea tener mouse en la consola, se le puede decir que se cargue en el arranque.

Ahora sigue configurar la red. Si sólo se tiene un modem y una conexión telefónica, o no se tiene siquiera una tarjeta de red, simplemente se puede escoger No en esta pregunta por ahora. Aunque preferiblemente se selecciona que si, y se escoge que haga un loopback, para especificar un hostname.

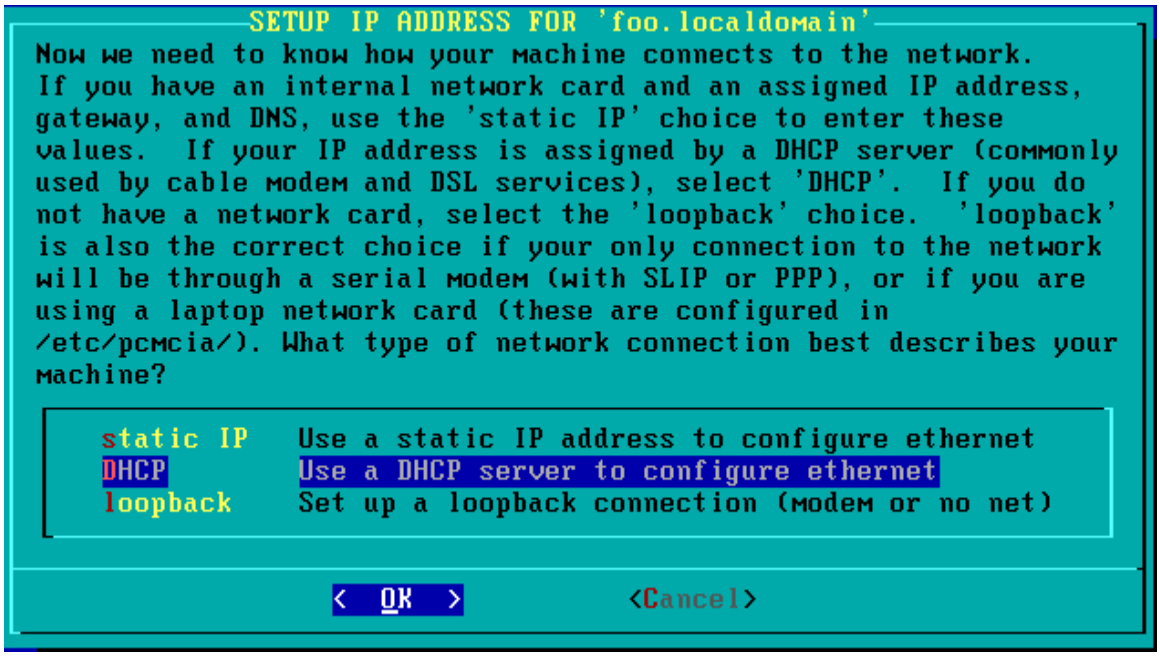
Para configurar una red, el primer campo a llenar es el hostname. Se escribe algo.

A continuación, se necesita el nombre del dominio.

Si se pretende participar en una red que tenga un nombre, entonces aquí se debe de escribir el nombre de dominio calificado, acabando en .com, .org, .edu u otros.

De otra forma, simplemente se escribe localdomain. En los pasos siguientes inclusive se va a poder borrar ese nombre (ya que realmente no se necesita).

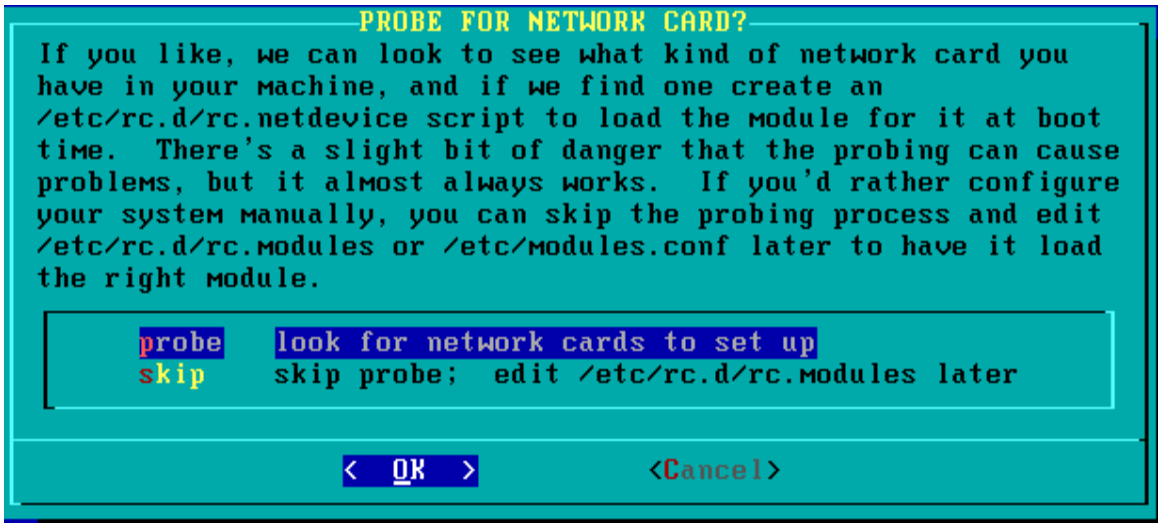
Ahora se escribe la dirección IP de la computadora.



Si la tarjeta de red se conecta a un cable modem o a un router de banda ancha, o utiliza una conexión PPPoE (PPP sobre Ethernet, comúnmente se utiliza en los servicios ADSL), entonces probablemente convenga escoger DHCP para que la información TCP/IP sea asignada automáticamente.

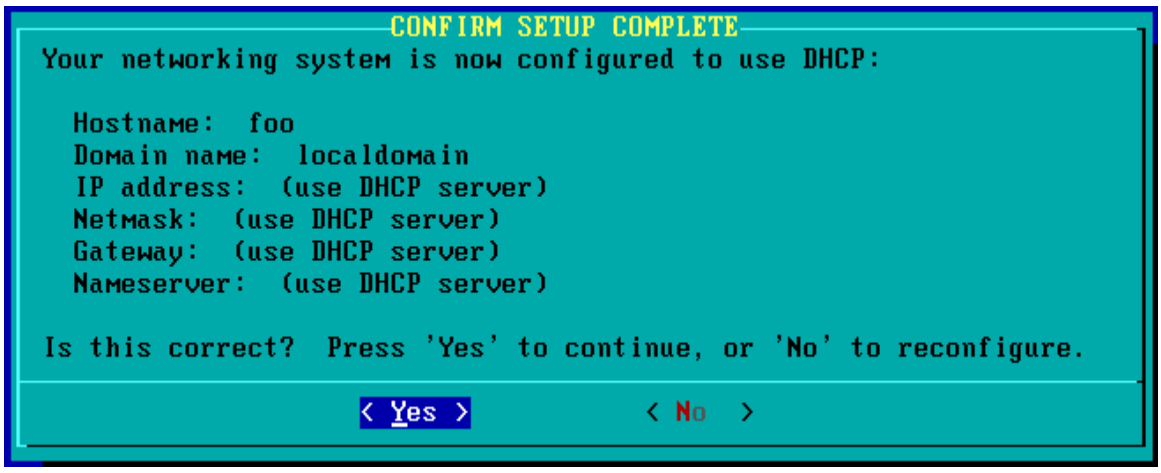
Si se selecciona DHCP, entonces pedirá un hostname de DHCP. Si se conecta a la red directamente por el cable modem, se necesita especificar el ID de usuario aquí.

De otra forma, simplemente se deja en blanco y se continúa a detectar la tarjeta de red.



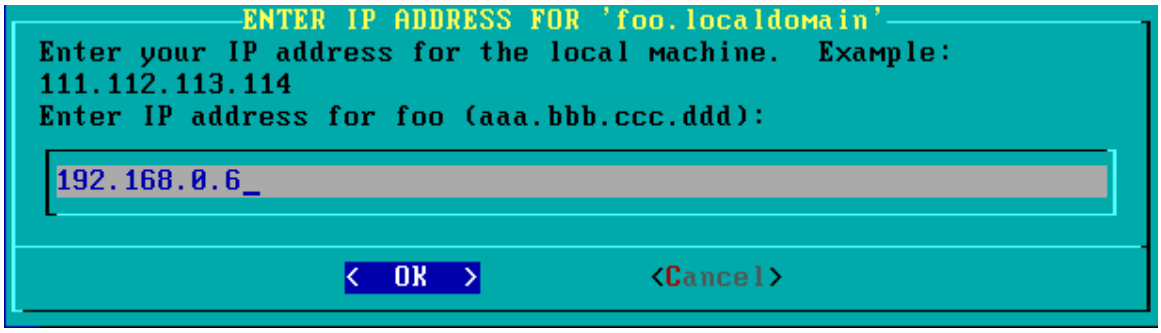
En caso de que la opción probe no la detecte automáticamente, no hay que preocuparse. Tan sólo significa que se tendrá que investigar cuál módulo kernel necesita el adaptador, y configurarlo después.

Si se escogió usar DHCP, una pantalla de confirmación es lo que se verá a continuación. Los pasos para configurar la red están completos.

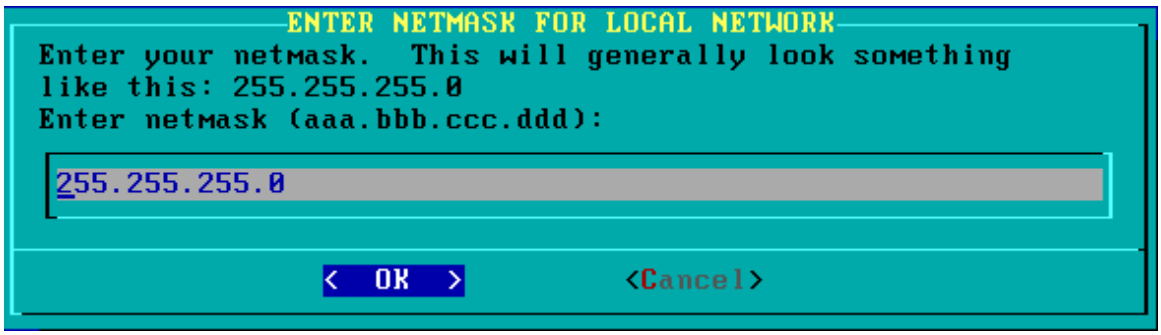


En caso de que se necesite escoger una configuración estática, aquí está el proceso.

Se escribe la dirección IP.

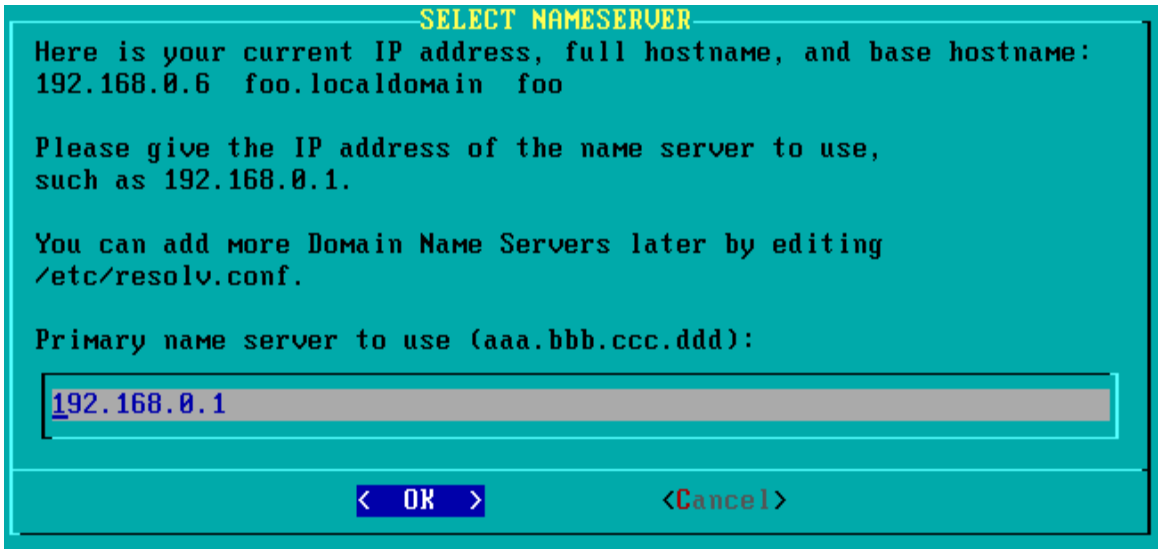


Se escribe la mascara de red.

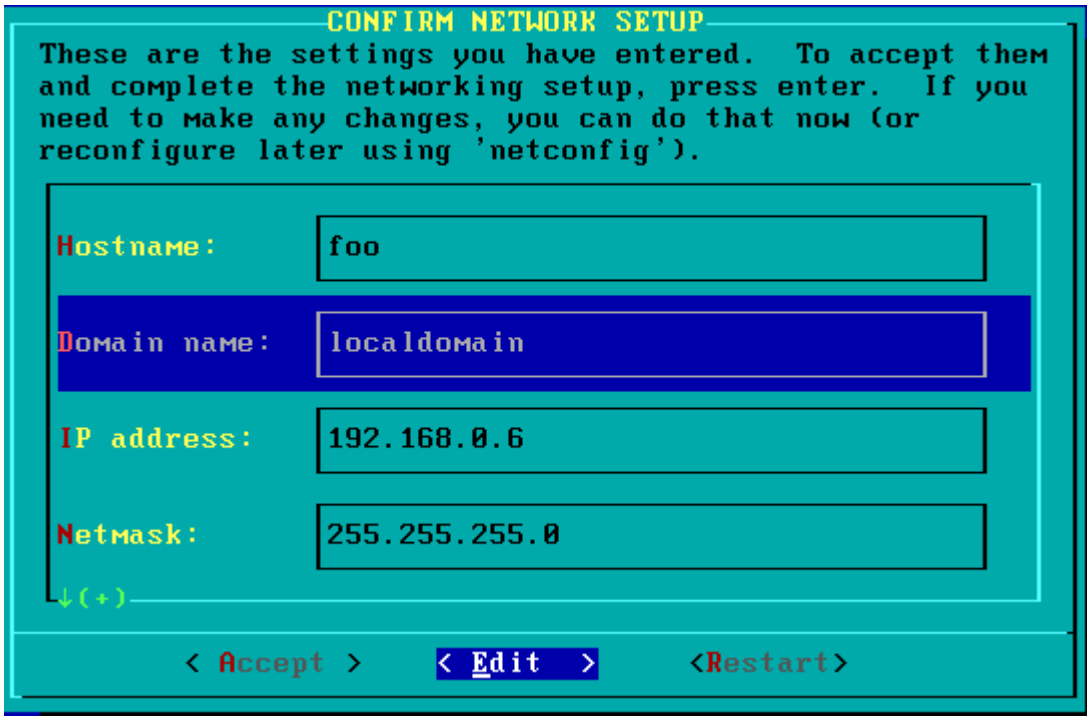


La puerta de enlace.

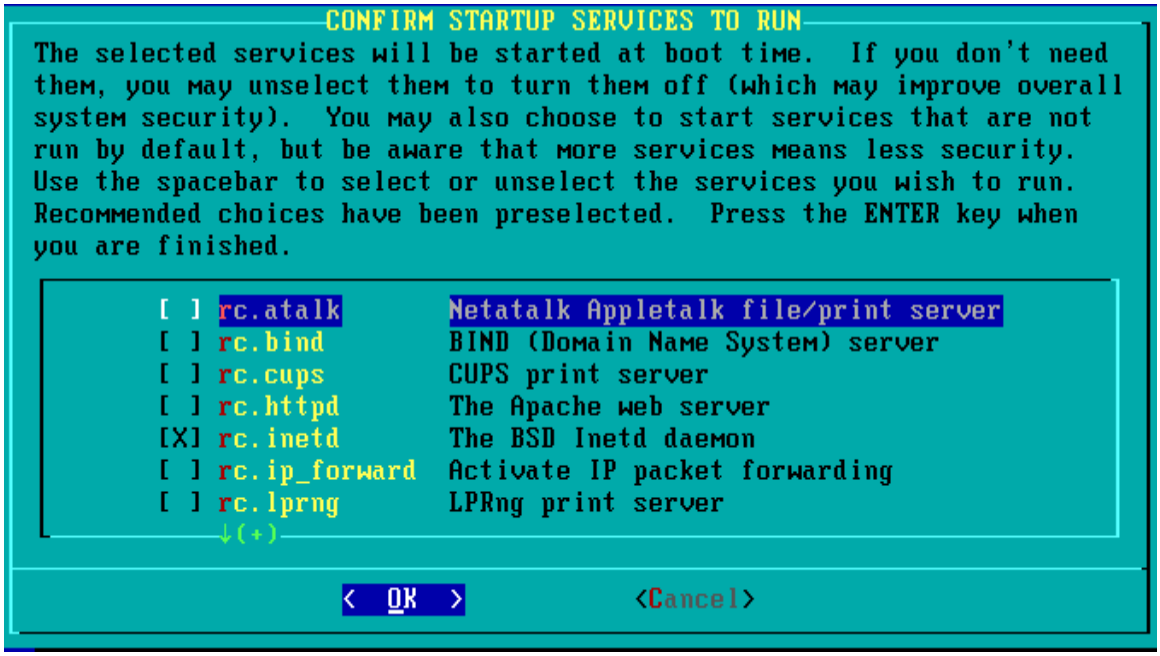
Y el nombre de servidor.



Ahora el sistema necesita que se confirmen las elecciones.

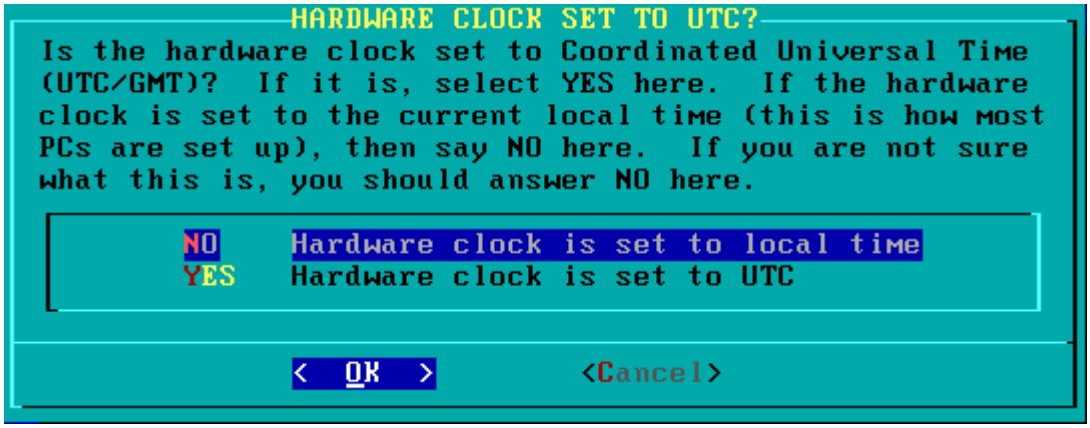


Esto concluye la porción de redes de la instalación. Lo que continúa son los servicios de inicio.

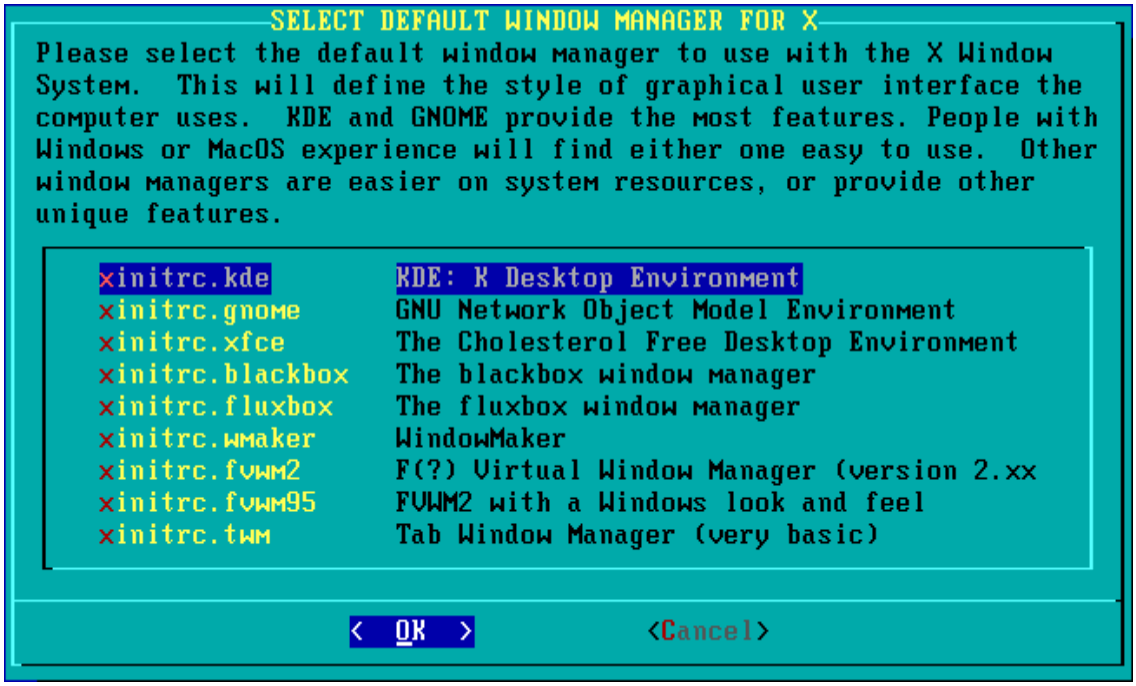


Muchas de estas opciones son demonios de red, y si la computadora va a ser usada como una estación de trabajo, preferiblemente se desactivan todos. Aunque tal vez se necesite el servidor de impresión CUPS.

Ahora se configura el reloj y la zona horaria.

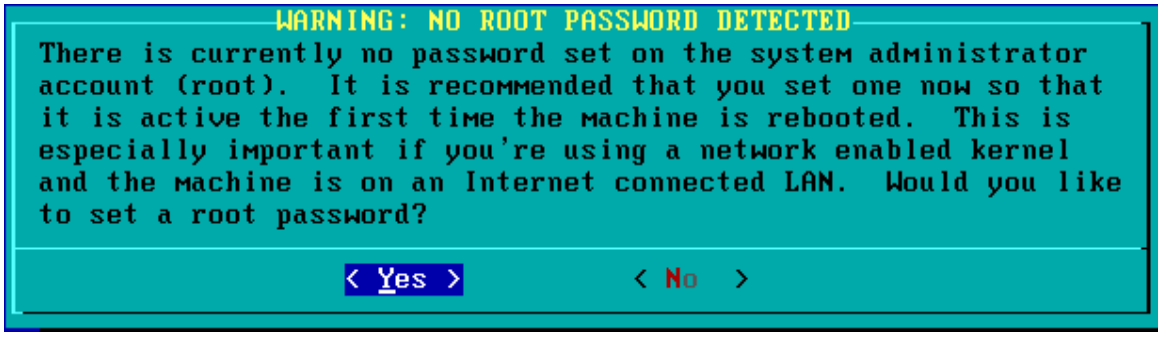


La siguiente pantalla pide que se seleccione una interfaz gráfica para usarla por defecto al invocar a Windows X.



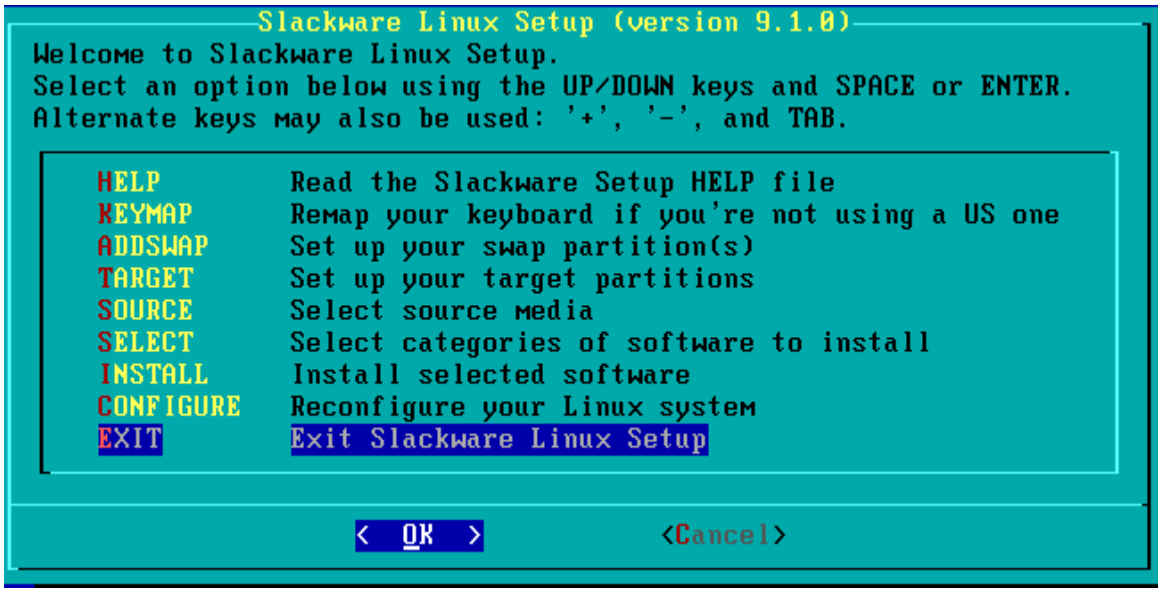
KDE es la más recomendable.

Root necesita una contraseña, y eso es lo que se escribirá a continuación.



Después de confirmar que se desea escribirla, se tiene que hacer en la consola dos veces. Es obviamente recomendable utilizar una contraseña compleja para root.

Por fin terminó la instalación de Slackware. El sistema pedirá que se retire el CD y se salga del asistente.



Se presiona ctrl-alt-delete y el sistema reiniciará para arrancar en Linux por primera vez.

Como se ha visto, la distribución Slackware es una excelente elección para trabajar en Linux, debido a su sencillo, pero poderoso desempeño y configuración. El proceso de instalación es complicado, pero no lo suficiente como para volverse tedioso o lento, y definitivamente es instructivo.

2.3 - Editores para la Creación de Páginas Web

Objetivo

Que el alumno conozca los usos del lenguaje HTML junto con los conceptos básicos de XML para que sea capaz de generar código de http de manera natural y rápida.

Contenido

El HTML, acrónimo inglés de Hypertext Markup Language (lenguaje de formato de documentos de hipertexto), es un lenguaje de marcas diseñado para estructurar textos y presentarlos en forma de hipertexto, que es el formato estándar de las páginas web. Gracias a Internet y a los navegadores del tipo Explorer, Mozilla, Firefox o Netscape, el HTML se ha convertido en uno de los formatos más populares que existen para la construcción de documentos.
HTML es hijo de SGML, aunque hay unas versiones de XHTML que son descendientes de XML y exigen que se escriba mucho más para facilitar la vida a los navegadores, que son aquellos programas que muestran información en pantalla.
El lenguaje HTML puede ser creado y editado con cualquier editor de textos básico, como puede ser el Bloc de Notas de Windows (o Notepad), o cualquier otro editor que admita texto sin formato como GNU Emacs, Microsoft Word, Microsoft Wordpad, TextPad, etc.

HTML utiliza etiquetas o marcas, que consisten en breves instrucciones de comienzo y final, mediante las cuales se determinan la forma en la que debe aparecer en su navegador el texto, así como también las imágenes y los demás elementos, en la pantalla del ordenador. Toda etiqueta se identifica porque está encerrada entre los signos menor que y mayor que (<>), y algunas tienen atributos que pueden tomar algún valor. En general las etiquetas se aplicarán de dos formas especiales:
Se abren y se cierran, como por ejemplo: negrita que se vería en el navegador como el texto negrita en negrita.

Existen algunas marcas que no necesitan cerrarse, como <hr> que se vería en su navegador como una línea horizontal. Algunas si lo necesitan, por ejemplo <p>, que separa texto en párrafos.

Las etiquetas básicas o mínimas son:

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

No hay especificación oficial del HTML 1.0 porque ya existían múltiples estándares informales del HTML cuando se decidió crear un estándar oficial. Los trabajos para crear un sucesor del HTML, posteriormente llamado 'HTML+', comenzaron a finales de 1993. El HTML+ se diseñó originalmente para ser un superconjunto del HTML que permitiera evolucionar gradualmente desde el formato HTML anterior. A la primera especificación formal de HTML+ se le dio, por lo tanto, el número de versión 2.0 para distinguirla de esos "estándares no oficiales" previos. Los trabajos sobre HTML+ continuaron, pero nunca se convirtió en un estándar.
El borrador del estándar HTML 3.0 fue propuesto por el recién formado W3C en marzo de 1995. Con él se introdujeron muchas nuevas capacidades, tales como facilidades para crear tablas, hacer que el texto fluyese alrededor de las figuras y mostrar elementos

matemáticos complejos. Aunque se diseñó para ser compatible con HTML 2.0, era demasiado complejo para ser implementado con la tecnología de la época y, cuando el borrador del estándar expiró en septiembre de 1995, se abandonó debido a la carencia de apoyos de los fabricantes de navegadores web. El HTML 3.1 nunca llegó a ser propuesto oficialmente, y el estándar siguiente fue el HTML 3.2, que abandonaba la mayoría de las nuevas características del HTML 3.0 y, a cambio, adoptaba muchos elementos desarrollados inicialmente por los navegadores web Netscape y Mosaic. La posibilidad de trabajar con fórmulas matemáticas que se había propuesto en el HTML 3.0 pasó a quedar integrada en un estándar distinto llamado MathML. El HTML 4.0 también adoptó muchos elementos específicos desarrollados inicialmente para un navegador web concreto, pero al mismo tiempo comenzó a limpiar el HTML señalando algunos de ellos como 'desaprobados'. Ya no va a haber nuevas versiones del HTML, sin embargo, la herencia del HTML se mantiene en XHTML, que se basa en XML.

Existen además, otros programas para la realización de sitios Web o edición de código HTML, como Microsoft FrontPage, el cual tiene un formato básico parecido al resto de los programas de Office. También existe el famoso software de Macromedia llamado Dreamweaver, que es de los más utilizados en el ámbito de diseño y programación Web. A estos programas se les conoce como editores WYSIWYG o What You See Is What You Get (“lo que ves es lo que obtienes”). Esto significa que son editores que van mostrando el resultado de lo que se esta editando en tiempo real a medida que se va desarrollando el documento. Ahora bien, esto no significa una manera distinta de realizar sitios web, sino que una forma un tanto mas simple ya que estos programas, además de tener la opción de trabajar con la vista preliminar, tiene su propia sección HTML la cual va generando todo el código a medida que se va trabajando. Combinar estos dos métodos resulta muy interesante, ya que de alguna manera se ayudan entre sí. Por ejemplo, si se está editando todo en HTML y de pronto algún código o etiqueta es olvidado, el editor visual o WYSIWYG puede continuar la edición, o viceversa, ya que hay casos en que es mas rápido y fácil escribir directamente el código de alguna característica que se quiera adherirle al sitio, que buscar la opción en el programa mismo.

XML

Su objetivo principal es conseguir una página web más semántica. Aunque una de las principales funciones con las que nace sería suceder al HTML, separando la estructura del contenido y permitiendo el desarrollo de vocabularios modulares, compatibles con cierta unidad y simplicidad del lenguaje (objetivo que se viene desarrollando a través de la especificación XHTML), tiene otras aplicaciones entre las que destaca su uso como estándar para el intercambio de datos entre diversas aplicaciones o software con lenguajes privados como en el caso del SOAP.
Al igual que el HTML, se basa en documentos de texto plano en los que se utilizan etiquetas para delimitar los elementos de un documento. Sin embargo, XML define estas etiquetas en función del tipo de datos que está describiendo y no de la apariencia final que tendrán en pantalla o en la copia impresa, además de permitir definir nuevas etiquetas y ampliar las existentes.
Son varios los vocabularios desarrollados en XML con el fin de ampliar sus aplicaciones. Los más fundamentales son: XHTML, XSL-FO y XSLT, XLink, XPointer y Schema. Además, existen también versiones para usos específicos, como

MathML (fórmulas matemáticas), SVG (gráficos vectoriales), RSS (sindicación de noticias), o XBRL (partes financieros).

ESTILOS Y EFECTOS BÁSICOS

Como ya se mencionó, la estructura lógica del texto y los diferentes efectos que se le apliquen se especifican mediante directivas.

TÍTULOS

Mediante los títulos, en sus diferentes niveles de importancia, se puede definir el esqueleto del documento, su estructura básica.

<h1>Mucha importancia</h1>
<h2>Menos importancia</h2>
<h3>Mucha menos importancia</h3>

Las etiquetas señalan el comienzo y el final de los títulos o encabezados, pueden tener un número del 1 al 7, donde un número bajo indica letras muy grandes, y un número alto representa letras muy pequeñas.

ATRIBUTOS DEL TEXTO

Mediante estos atributos se determina el estilo y el tipo de letra que tendrá la presentación del documento final.
El primero que se debe analizar es el texto normal entendiendo como tal el que no tiene ninguna característica especial. Para definir un párrafo como normal no es necesario poner ninguna etiqueta. Lo único que hay que tener en cuenta es que al presentar el documento se hace caso omiso de los espacios, tabulaciones y retornos de carro que se encuentren en el texto fuente; por ello cuando se quiera forzar un final de línea es necesario utilizar dos directivas especiales: <p> para marcar un fin de párrafo, y
 para un único retorno de carro. La diferencia entre ambas es que la separación de líneas que provoca <p> es algo mayor que la de
, para que los párrafos se distingan bien entre sí. Las dos directivas mencionadas se sitúan en el punto en que se desea poner la separación. Por supuesto, estas dos etiquetas se puede aplicar donde sea, no sólo en el texto normal.
El texto preformateado (etiqueta <pre>) se aplica cuando se quiere que en la presentación final del documento se respeten los espacios y retornos de carro que se hayan puesto en el texto fuente. Además se utilizará un tipo de letra de espaciado fijo, parecido al de una máquina de escribir, más pequeño que el del texto normal. Este estilo de texto puede ser adecuado, por ejemplo, para una tabla numérica sencilla. Para hacer una cita textual dentro del documento, se puede utilizar la directiva <blockquote>. Las direcciones de correo electrónico se suelen marcar con esta directiva: <address>. Se pueden dar también los atributos más tradicionales: negrita y cursiva.<b> e <i>.

Las listas se definen de forma muy sencilla: se dice dónde empieza la lista, dónde empieza cada punto y dónde acaba la lista. Las etiquetas que se utilicen en cada caso deben aparecer al principio de línea, o al menos sin texto por delante (sólo espacios o

tabulaciones). Se puede recurrir a tres tipos distintos de listas, cada una con una presentación diferente: no numeradas, numeradas y listas de definiciones (glosarios).
Las listas se pueden anidar, es decir, en el lugar donde debería ir uno de los términos de la lista se pone una nueva lista, que por supuesto no tiene porqué ser del mismo tipo. Se utilizan las marcas <ul> para comenzarlas, <li> para cada miembro, y para cerrarla. En caso de que no se deseen números, sino puntos, <ol> se usa para comenzar la lista.

La directiva <hr> sitúa en el documento una línea horizontal de separación. Una especie de separador de texto.

Para poner un comentario en un documento HTML, es decir, una aclaración que no aparece en la presentación final del documento, se encierra el texto que formará el comentario entre los símbolos <!-- y -->.

ENLACES Y GRÁFICOS

Además de los muchos estilos y capacidades de presentación que ofrece HTML para estructurar el documento en sí, se dispone de varias directivas que permiten definir relaciones entre diferentes documentos y estructurar todo un conjunto de documentos para crear una unidad lógica. La facilidad para definir este tipo de enlaces es una de las razones de la potencia y versatilidad de HTML.
Los enlaces en HTML se expresan rodeando con la directiva <a> el objeto (que puede ser un fragmento de texto o un gráfico) que vaya a servir como anclaje para el enlace. Por ejemplo, si se marca con <a> un gráfico, cuando en el documento final se pulse con el mouse sobre dicho gráfico se saltará al objeto referenciado en el enlace: otro documento, un vídeo musical, o un servidor de información meteorológica.

Para especificar de manera uniforme el objeto al que apunta el enlace, se utiliza una forma estandarizada que se denomina URL (Uniform Resource Locator, es decir, Localizador Uniforme de Recursos). Un URL está formado de la siguiente manera: esquema://maquina/ruta (en realidad, como se verá dentro de un momento, la barra / puede considerarse parte de la ruta).
El esquema es un nombre que identifica el tipo de servicio que va a proporcionarse en el destino del enlace. La razón de esta aparente complicación es que el WWW pretende unificar el acceso a servicios de información que previamente eran incompatibles entre sí, como ftp, gopher o telnet. El esquema más utilizado es http, correspondiente al propio WWW (es decir, que cualquier referencia a un documento HTML debería comenzar con http://). Otros esquemas muy frecuentes son ftp, telnet, gopher o wais.
La máquina y la ruta sirven para localizar el objeto al que apunta el enlace. La máquina es la identificación del servidor en el cual está situado el objeto al que apunta el enlace.
Puede ser simplemente el nombre de una computadora (como www.etsit.upm.es) o también un nombre y un puerto (por ejemplo www.etsit.upm.es:8000).
La ruta es el nombre del fichero que contiene el documento en concreto, incluyendo el nombre del subdirectorio en el que se encuentra. Los diferentes nombres que constituyan la ruta completa al archivo se deben separar con la barra /, tal y como se hace en el sistema operativo UNIX (y al revés que en MS-DOS). La razón de este convenio es precisamente que la mayor parte de los servidores de WWW que hay en Internet son ordenadores basados en UNIX, debido a la gran superioridad tecnológica de este sistema sobre MS-DOS. Esto se nota también en que por lo general los nombres

de los ficheros no tienen muchas limitaciones: pueden ser casi tan largos como se desee, contener varios puntos, etc. Por ejemplo, el nombre de cierto directorio situado en un servidor podría ser /info/documentos/ciencia/fisica/relatividad.html. Se debe tener en cuenta que en UNIX las mayúsculas y las minúsculas son distintas en los nombres de los ficheros: no es igual DIRECTORIO que directorio.
Conviene que se enfoque momentáneamente en la estructuración habitual de los ficheros en un servidor de WWW. Para empezar, siempre hay una página de bienvenida (homepage) que podría compararse con la portada de un periódico o revista; si no se sabe exactamente qué es lo que se busca, o dónde encontrarlo, la portada es lo primero que se ve. Para acceder a la homepage de cualquier servidor de WWW, basta con escribir una barra en el lugar de la ruta (es decir, se reclama al servidor el directorio raíz). Por ejemplo, para acceder a la página de la NASA habría que contactar con http://www.nasa.gov/.
El resto de la información que se puede encontrar en un servidor de WWW se distribuye a partir de ese directorio raíz en distintos subdirectorios y archivos. Un convenio muy habitual relativo al nombre de los ficheros es hacer que los archivos que contengan documentos HTML terminen en .html.

Con estos conocimientos es posible abordar sin problemas el asunto que originalmente ocupaba: cómo se introducen enlaces en un documento HTML. Para definir un enlace es necesario marcar con la directiva <a> el objeto del cual va a partir dicho enlace. Dicha directiva debe incluir el parámetro href="URL" para especificar el destino del enlace. Es decir, que antes del objeto elegido se debe abrir con , y después cerrar con . Por ejemplo, si se quisiera que el texto pulse aquí para visitar la NASA conduzca a la home page de la NASA, se debe escribir en el texto HTML:

Pulse aquí para visitar a la NASA

Por lo general no preocupa ir tan lejos, sino sencillamente enlazar con otro documento que se encuentra en el mismo servidor, puede que incluso que en el mismo subdirectorio. En este caso no es necesario escribir el camino completo al destino del enlace, sino que basta con dar la mínima información imprescindible. El programa que se use para leer el documento final suele ser lo bastante listo como para deducir el resto. Es decir, que si desde cierto documento se quiere enlazar con otro que se encuentra en el mismo subdirectorio, basta con poner su nombre. También pueden utilizarse rutas relativas.

Para incluir un gráfico en un documento HTML se utiliza la directiva <img>. En dicha directiva debe incluirse un parámetro src="URL", con el cual se indica dónde está el fichero con el gráfico concreto que se quiere para el documento. Esto supone una gran flexibilidad, ya que es posible complementar el contenido del documento tanto con gráficos que se encuentren disponibles en el servidor de WWW como con una foto situada en un servidor de la NASA o de la UNAM, por ejemplo, sin que el lector final tenga por qué apreciar ninguna diferencia.
Existe alguna limitación respecto a los formatos gráficos que los programas lectores de HTML pueden interpretar sin problemas. El formato fundamental es el GIF, que cualquier programa con capacidades gráficas debería poder mostrar directamente en texto (Mosaic y Netscape pueden hacerlo). Si se utiliza otro formato diferente, lo más probable es que cuando un lector esté accediendo al documento, el programa no comprenda ese formato y se tenga que solicitar la ayuda de otro programa, con lo cual al

final el gráfico no se insertará en el lugar estratégico del documento, sino que aparecerá en otra ventana diferente.
Hay un parámetro optativo de la directiva <img> que sirve para proponer un texto alternativo a un gráfico. Este texto aparecerá cuando se esté usando para leer el HTML un programa sin capacidades gráficas (por ejemplo Lynx, que sólo trabaja con texto). Se trata de alt="texto". Conviene utilizarlo cuando los gráficos sirven como origen a hiperenlaces, porque si no los programas sin capacidades gráficas no podrían mostrar los enlaces deseados.
Como ocurría antes con los enlaces, por lo general no es necesario escribir el URL completo, sino que basta con dar la mínima información. Y además se puede hacer que un gráfico sea un enlace, utilizando la directiva <a>. En este caso no se debe olvidar utilizar la opción alt="texto" para que todos los usuarios puedan seguir el enlace.

CARACTERES ESPECIALES

HTML tiene ciertos límites en cuanto a caracteres aceptables se refiere, esto incluye a vocales acentuadas, la letra Ñ, algunos símbolos, etc. Existe una razón evidente que impide que se pueda escribir ciertos símbolos directamente en un texto HTML, por ejemplo el <: dichos símbolos tienen un significado en HTML, y es necesario diferenciar claramente cuándo poseen ese significado y cuándo se quiere que aparezcan literalmente en el documento final. Por ejemplo, como se sabe, < indica el comienzo de una directiva, y, por ello, si se desea que aparezca en el texto como tal tendrá que dar un rodeo escribiendo algo que no de lugar a confusión, en este caso &lt. Los símbolos afectados por esta limitación, y la forma de escribirlos, se detallan a continuación:

< (Menor que): <
> (Mayor que): >
& (símbolo de and, o ampersand): &
" (comillas dobles): "

El otro caso especial se da cuando en un texto HTML se quiere escribir una eñe, por ejemplo. Existen dos formas de hacerlo. La primera, que es a la que obliga el estándar de HTML, consiste en utilizar entidades, es decir, aliases como las que antes se presentaron para escribir ciertos símbolos. Las entidades comienzan siempre con el símbolo &, y terminan con un punto y coma (;). Entre medias va un identificador del carácter a escribir. Las entidades necesarias en español son:
á: á
é: é
í: í
ó: ó
ú: ú
Á: Á
É: É
Í: Í
Ó: Ó
Ú: Ú
ü: ü
Ü: Ü
ñ: ñ
Ñ: Ñ

¿: ¿
¡: ¡

Como puede verse, las vocales acentuadas se identifican añadiendo el sufijo acute a la vocal sin acentuar (puesto que se trata de un acento agudo). Para la u con diéresis y la eñe se usan uml tras una u y tilde detrás una ene, respectivamente. La equivalencia de los signos de abrir interrogación y exclamación es algo más oscura: a falta de una denominación más evidente, se tiene que usar el valor numérico de dichos caracteres en el código estándar latin1 (ISO-8859-1). Esto se puede hacer con cualquier otro carácter del código latin1, que es el código de caracteres básico en HTML, escribiendo &#numero;.

La segunda manera, que sin duda es más cómoda, consiste en no preocuparse por esta limitación y escribir literalmente los caracteres afectados. A pesar de que este método suele funcionar en las conexiones WWW directas (porque el protocolo HTTP, que transporta el HTML por los vericuetos de Internet, requiere un canal de 8 bits), no tiene por qué funcionar bien cuando los documentos HTML se envían por correo electrónico, por ejemplo. Por tanto, y a pesar de los inconvenientes, es absolutamente recomendable respetar la norma especificada en HTML.
En cualquier caso, no resulta muy complicado escribir un programa que traduzca todas las apariciones de los caracteres especiales por sus correspondientes entidades HTML, o viceversa. Con un programa así, uno puede escribir los documentos sin preocuparse por estos problemas, y luego traducir a HTML correcto.

La recolecta de datos forma una parte muy importante para cualquier sitio de Internet y esta también se lleva acabo con elementos HTML denominados formas, los elementos principales de una forma son las marcas <form> que indican como y a donde será enviada la información. Dentro de <form> se declaran dos atributos muy importantes, uno de ellos es method y el otro action.

El atributo method le indica al navegador como debe ser enviada la información al servidor, este valor puede ser post o get. Si se utiliza get, los elementos de la forma serán enviados al Servidor en el URL de action, esto es, si dentro de la forma se solicitan los valores: nombre, edad y nacionalidad, la información sería enviada de la siguiente manera:

http://www.osmosislatina.com/cgi-bin/confirmar.pl?nombre=PedroMtz&edad=25&nacionalidad=mexicana

Después del URL el navegador agrega “?” para indicar que continúan variables solicitadas, a partir de ? continúan las variables definidas por medio de un = y separadas por un &, en cambio si se utiliza post las variables y sus valores se incluirán en el contenido de la requisición, por debajo del URL. Lo anterior garantiza que el usuario final no sea capaz de observar las variables en el URL, fuera de esto, la única implicación en utilizar post o get es que la aplicación de Servidor sea capaz de leer las variables y sus valores.
En el caso anterior la Aplicación de Servidor (Programa) que procesa esta información es aquella definida por el atributo action, la cual es un "script" en Perl llamado: confirmar.pl; en otros casos esta Aplicación de Servidor pudo ser: Un JSP ("Java Server Page"), ASP ("Active Server Page"), PHP ("Hypertext Preprocessor") u otra

metodología de Servidor. Para declarar variables en una forma se pueden utilizar varios métodos o menús, los cuales son descritos a continuación:

Etiquetas input
La etiqueta input permite pasar un conjunto de variables con sus respectivos valores en una forma. Son las siguientes:

Tipo Text
Este tipo de variable permite al usuario final asignar un valor de texto a la variable en cuestión, se expresa de la siguiente manera:

País: <input type="text" name="pais" size="10" maxlength="20" value="mexico">

La declaración indica que será desplegado un recuadro de 10 espacios (size) donde la variable llevará por nombre país, dicha variable puede tener un valor máximo de 20 letras (maxlength) y aparecerá un valor "default" mexico (este valor puede ser modificado por el usuario).

Tipo Password
También permite que el usuario especifique un valor de texto, pero a diferencia del tipo text el usuario observará asteriscos (****) mas no el texto.
NOTA: A pesar que el valor no aparece en pantalla, el valor no es protegido al momento de ser transferido al nivel de Red, para realizar esta protección se requiere utilizar encriptación en el Servidor de Paginas.

Tipo Radio
Este tipo de variable es utilizada para desplegar un menú con botones.

País:<input type="radio" name="pais" value="ar">Argentina
<input type="radio" name="pais" value="br">Brazil
<input type="radio" name="pais" value="co">Colombia
<input type="radio" name="pais" checked value="mx">México
<input type="radio" name="pais" value="ve">Venezuela

Lo anterior indica que el nombre de la variable pais será un menú de selección de botones, donde el valor "default" (checked) será México, los parámetros value indican el valor que tomará la variable en caso de ser seleccionada; value es de gran ayuda cuando la descripción de la variable es extensa y solo se desea enviar un vocablo o iniciales como valor al Servidor de Paginas, si se omite el valor value , el valor de la variable pasa a ser aquel que se encuentra del lado derecho del elemento.

Tipo Checkbox
Este tipo de variable es utilizada para desplegar un menú con cuadros de selección, a diferencia del tipo Radio , Checkbox permite la selección de múltiples valores :

Intereses:<input type="checkbox" name="intereses" value="p">Política
<input type="checkbox" name="intereses" checked value="d">Deportes
<input type="checkbox" name="intereses" value="c">Cine
<input type="checkbox" name="intereses" checked value="t">Tecnología
<input type="checkbox" name="intereses" value="a">Administración

Los parámetros que son utilizados para este tipo de variable tienen el mismo funcionamiento que el tipo radio (name, checked, value), la única diferencia es que si son seleccionados varios valores, la información es pasada al Servidor de Paginas como: intereses="d" intereses="t", por lo tanto debe asegurarse que su Aplicación de Servidor sea capaz de procesar dos o más valores para la misma variable.

Tipo Hidden
Este tipo de elemento es utilizado para esconder el valor de cierta variable, se utiliza generalmente cuando es necesario procesar datos a lo largo de dos o tres páginas, de esta manera la variable sigue existiendo en la forma, pero no es desplegada en pantalla.

```
<input type="hidden" name="intereses">
```

Etiqueta select
La etiqueta select permite generar menús de una manera similar a las mencionadas anteriormente.

```
<select size="3" name="intereses" multiple>
<option  value="p">Política
<option  value="t" selected>Tecnología
<option  value="d">Deportes
<option  value="c">Cine
<option  value="m">Música
</select>
```

Los parámetros del Etiqueta select son:

- name: Indica el nombre de la variable (como las etiquetas input).
- size: Especifica el número de opciones que serán desplegadas en pantalla (las restantes serán incluidas en forma de menú).
- multiple (opcional): Permite al usuario final elegir varios valores (si se omite multiple solo se permite seleccionar un valor, como el comportamiento del input radio); para seleccionar diversos elementos de manera aleatoria, se oprime del teclado Ctrl.
- <option>: Define los valores de la variable en cuestión, dentro de estas etiquetas es posible indicar el parámetro value que tiene el mismo funcionamiento que las etiquetas input, y selected que permite preseleccionar ciertos valores.

Etiqueta textarea
Las marcas textarea son muy similares a input text, la diferencia estriba en que textarea permite definir el área por renglones y columnas. El parámetro name indica el nombre de la variable que llevará el contenido, mientras cols y rows definen el espacio que es visible en pantalla, cualquier texto que aparezca entre los etiquetas textarea será desplegado dentro del recuadro.

HTML provee de la infraestructura básica para poder despegar hacia implementaciones de proyectos y sitios con códigos y lenguajes más poderosos. En las aplicaciones de servidores web y protocolos de procesamiento en servidor como PHP, estos comandos de HTML serán básicos, proveyendo al sistema de la estructura básica en cuanto a formas, texto, y encabezados se refiere.

2.4 - Administración de Servidores WWW con Linux

Objetivo

Ofrecer al estudiante los conocimientos necesarios para instalar, configurar y administrar un servidor www basado en tecnología Apache sobre una plataforma Linux. El estudiante comprenderá al final del curso los métodos y funcionamientos del software de servidor.

Contenido

Un servidor web es un programa que implementa el protocolo HTTP (hypertext transfer protocol). Este protocolo está diseñado para transferir los llamados hipertextos, en otras palabras, páginas web o páginas HTML. 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, aunque más exactamente, se está refiriendo a la aplicación. Un servidor web se encarga de mantenerse a la espera de peticiones HTTP llevada a cabo por un cliente HTTP conocido como 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, de 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.

Los servidores de web más populares en el mercado actualmente son:

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

Con todas estas opciones, los argumentos que se deben considerar para seleccionar uno son:

- Función del Servidor de web
- Expertise de los administradores
- Plataforma disponible
- Numero de conexiones concurrentes
- Numero transacciones por segundo
- Costo computacional por transacción
- Proyección del crecimiento esperado.
- Soporte para la tecnologia utilizada para el desarrollo.
- Analisis del retorno de Inversión.

Para este proyecto, se eligió Apache Web Server, por muchas razones; no sólo es software libre, lo que lo hace completamente gratis, sino que también es el preferido por un amplio margen sobre todos los demás. De hecho, según las ultimas estadísticas, el número de usuarios de Apache excede a todos los demás servidores juntos. Son embargo, más específicamente, las razones para escogerlo sobre IIS por ejemplo, se listan a continuación:

- Robusto, Soporte de un gran numero de transacciones.
- Configurable para diferentes entornos de trabajo.
- Con un alto nivel de seguridad.
- Disponible para una gran variedad de plataformas.
- Soporte para servicio de proxy
- Soporte para granjas de servidores
- Soporte para Scripting languages integrados como modulos (por ejemplo PHP, mod_perl)
- Incluye el codigo fuente del servidor
- Soporte para accesos restringuidos
- Soporte para SSL

Así, se analizará antes que nada, su instalación en una plataforma Linux, aunque bien se pueden seguir los mismos pasos para una plataforma Unix. El servidor, como ya se había mencionado, está disponible de forma gratuita en el sitio de Apache (http://www.apache.org). En este sitio es posible descargar el servidor http, y lo primero que se verá es que está disponible en forma de RPM y como un simple tar.gz. Durante este ensayo no se han analizado las ventajas y desventajas del sistema RPM; en breve, fue desarrollado para la distribución Red Hat, y automatiza la instalación de paquetes en el sistema. Aunque esto lo hace al parecer más sencillo, no ofrece el nivel de control que se va a requerir de Apache, así que lo mejor, inclusive si se está trabajando con una distribución Red Hat o Fedora, es simplemente obtener el tar.gz que no es difícil de instalar. Una vez que se tenga el archivo comprimido en la computadora, se descomprime a su directorio, y desde ahí se utiliza la instrucción "./configure –prefix=[ruta]" en la línea de comandos. Una vez terminado ese proceso, se utilizan las directivas "make", y "make install" secuencialmente. Con esto está el servidor instalado.

Administración en Linux de Apache Web Server

Apache es administrado por más de 200 directivas las cuales permiten que determinada funcionalidad pueda ser incluida. En Linux el administrador controla que directivas estarán disponibles de acuerdo a los módulos con los que se compiló Apache al instalarlo.
La configuración se realiza mediante directivas dispuestas en tres grupos:

- Global Environment
- Main Server
- Virtual Servers

Estos grupos se pueden encontrar en el archivo httpd.conf, que se encuentra en el directorio /usr/local/apache/conf. La sección Global Environment administra las directivas generales de operación para apache. La sección Main Server administra las directivas del servidor principal o estandar de apache. La secciónVirtual Servers, administra las directivas donde los mismos procesos de apache soportan diversas IPs o nombres de dominio.

Algunas de las directivas más importantes en las tres secciones son:

ServerName.- define el nombre para el servidor Apache, el cual será utilizado cuando el cliente solicite otros recursos web.
User.- esta directiva define el user ID por el cual operará Apache. Su valor por default es User#-1
GroupName.- esta directiva define el group ID por el cual operará Apache.
Listen.- antes de la versión 2.0, esta directiva era llamada port. Señala el puerto por donde Apache operará. Su valor por default es 80, que es el puerto estándar que utilizan generalmente todos los servidores.
ServerAdmin.- aquí se define el correo electrónico del administrador, e indica la dirección a incluirse en los mensajes de error que pueda mostrar el servidor. Aunque para que esto funcione, se necesita activar también la directiva ServerSignatureEMail.
DocumentRoot.- define la ruta absoluta donde se encuentran los archivos html que se desean publicar. Sus valores por defecto son /usr/local/apache/htdocs si se compila al instalarlo, y /var/www/html si simplemente se utiliza el RPM.
ServerRoot.- define la ruta absoluta donde se encuentran los archivos conf y las bitácoras. Usualmente el valor por defecto siempre es el directorio de Apache.
MinSpareServers.- esta directiva define el número de procesos mínimo que son mantenidos en spare, es decir, disponibles para ser llamados. Su valor por defecto es de 5.
MaxSpareServers.- a diferencia de la anterior, esta directiva controla el número máximo de procesos de servidor spare. Eliminará los números en exceso siempre y cuando no estén siendo utilizados.
StartServers.- define el número inicial de procesos al arrancar Apache.
MaxClients.- define el número máximo de procesos que podrán ser iniciados, con el objeto de atender la solicitud de los clientes. Su valor defecto es de 20.
ErrorDocument.- en caso de que un error o problema ocurra con Apache cuando se solicite un recurso, este puede ser configurado para que haga una de las siguientes acciones:

1. Enviar un mensaje de error (default).
2. Enviar un mensaje personalizado.
3. Redirigir a un URL local para manejar el problema o error.
4. Redirigir a un URL externo para manejar el error o problema.

PidFile.- la directiva PidFile define el lugar donde se almacenará el archivo Pid, que es un valor numérico que identifica al proceso httpd “padre”. En caso de que se pierda este archivo, se pueden tener problemas al invocar el demonio desde el script en /etc/.
ErrorLog.- Apache cuenta con dos archivos de bitácoras: access.log, que almacena todos los ingresos al sitio, y error.log, que registra los errores que genere un acceso o que las directivas puedan reportar. Esta directiva define el archivo para los errores.
LogLevel.- define el nivel de mensajes de error que entran a la bitácora: emerg, alert, crit, error, warn, notice, info, debug.

CustomLog.- esta directiva señala al servidor el nombre del archivo en el cual registrará los sucesos que se presenten.
LogFormat "format-string" <nickname>.- "Format-string": "%h %l %u %t \"%r\" %>s %b". Donde se especifica el formato para registrar las bitácoras. %h = Registra la IP del cliente, hostname. %l = Si el deamon identd corre en el cliente, reporta la información que el identd devuelva. %u = Si se requiere de un username y password para acceder, en este campo se registra. %t = Fecha y hora del request en el formato [dia/mes/año:hora:minuto:segundo tzofffset]. \"%r\" = Recurso solicitado por el cliente, entre comillas, request. %>s = Un código de tres digitos donde se muestra el valor devuelto al cliente, status. %b = El número de bytes devueltos exceptuando encabezados.
HostNameLookUp.- habilita o deshabilita la resolución de nombres en el DNS cuando se establece una conexión. Por razones de rendimiento se sugiere no habilitarla. Para la resolución de nombres en las bitacoras existe el programa logresolve, tambien de apache, que puede realizar la resolución de nombre fuera de línea.
<Directory>y<DirectoryMatch>.- permite aplicar otras directivas de forma específica a todos los recursos dentro de la ruta absoluta definida.
Options.- controla que características del servidor están disponibles para un directorio en particular. Puede ser colocado a none en caso de que no se requieran funcionalidades especiales. Las opciones a escoger son:

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

ScriptAlias.- la tecnología CGI "Common Gateway Interface" define un modelo de programación que puede ser implementado por múltiples lenguajes. Los CGIs son programas que siguen un estándar definido, corren en el servidor, reciben parámetros desde el cliente y su salida es enviada al navegador. Fueron la primera alternativa para generar dinamismo a un sitio web. Esta directiva convierte las solicitudes via URL a la ruta absoluta donde se encuentran los scripts CGI.
AddHandler.- permite que determinada extensión sea relacionada a un evento en particular, en el caso de los CGIs, lo que se indica es que las extensiones .cgi y .pl quedan identificadas como script.
AddType text/html .shtml.- la opción SSI activa un filtro que permiten incluir etiquetas dentro de un archivo HTML, las cuales son procesadas por Apache, antes de ser enviadas como HTML al cliente. Es necesario habilitar la opción +Includes en el directorio donde se encuentran los archivos que incluyen etiquetas SSI. Esta directiva relaciona los archivos que tienen extensión shtml como aquellos que contienen etiquetas SSI.
AddOutputFilter INCLUDES .shtml.- habilita el filtro para los archivos .shtml. hay que tener en cuenta que SSI representa un riesgo, así que hay que tener cautela al usarlo.

Los servidores virtuales (o virtual hosts) permiten que un mismo servidor Apache pueda responder a diferentes solicitudes, con lo cual es posible mantener múltiples sitios web con diferentes nombres y/o direcciones IPs. Como ejemplo al manejar servidores virtuales en el archivo httpd.conf, están estas líneas:

```
NameVirtualHost 192.168.40.5

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

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

Como se puede ver, la directiva **NameVirtualHost** recibe una dirección IP, y en el campo de virtual servers, se especificarán los servidores virtuales de esa misma conexión, y sus nombres respectivos.

Apache ofrece la funcionalidad al estilo "ACL", con lo cual es factible determinar las direcciones IPs y los usuarios que tendrán acceso sobre recursos especificos del servidor Web. La configuración de acceso vía nombre de dominio o vía IP puede realizarse con ayuda de las siguientes directivas:

Order.- define el orden en que las directivas allow y deny serán definidas.
Allow.- define los hosts que tendrán acceso al recurso.
Deny.- define los hosts que no tendrán acceso al recurso.
AuthUserFile <archivo>.- indica un archivo con una lista de passwords para realizar la autenticación de usuarios.
AuthName "nombre".- define el nombre del recurso a ser usado. Imprime este nombre en la ventana de password.
Require valid-user.- indica que se requiere de un usuario y una clave de acceso para un recurso determinado.

Módulos

En Apache los módulos son los encargados de agregar funcionalidad incrementando el número de directivas disponibles. Existen los módulos estáticos, y los módulos dinámicos; los módulos estáticos son aquellos que se agregan al momento de compilar Apache, y los módulos dinámicos se agregan durante el ambiente productivo, sin necesidad de volver a compilar.

A continuación se presenta un ejemplo de construcción de un sitio basado en Apache. Para mantener una imagen realista, se utilizará el frente de una compañía en la red, Fotos Inc, que crea y envía postales con imágenes. Tiene que tener su propia dirección de red, pero ya que no se desea salir al mundo exterior en este ejemplo, se tiene que

hacerla una simple variante de la ID de red. Esto se logra alterando el archivo hosts (en los folders /Windows/ y /etc/ dependiendo del sistema operativo) en las máquinas visitantes y el servidor:

127.0.0.1 localhost
192.168.123.2 www.fotos.com
192.168.123.2 sales.fotos.com
192.168.123.3 sales-IP.fotos.com

El archivo de configuración httpd.conf ahora contiene lo siguiente:

User webuser
Group webgroup
ServerName localhost
DocumentRoot /usr/www/site.simple/htdocs
TransferLog logs/access_log

En htdocs es posible tener un archivo de texto llamado 1.txt que contenga la simple cadena de “Hola mundo”

Si en este momento se utiliza una máquina cliente para entrar a www.fotos.com desde un navegador, se debería ver:

Index of /
. Parent Directory
. 1.txt

Haciendo clic en 1.txt se puede ver la cadena de bienvenida. Todo esto es normal, pero si se experimenta, se obtiene el mismo resultado si se ingresa a sales.fotos.com. Porque? Ya que no se ha mencionado ninguna URL o sus direcciones IP en el archivo de configuración, no se debería obtener ninguna respuesta. Sin embargo, cuando se configura la máquina en donde corre el servidor, se le dijo a la interfaz de red que respondiera a cualquiera de estas direcciones de IP:

192.168.123.2
192.168.123.3

Por defecto, Apache escucha a todas las direcciones IP que pertenecen a la máquina y responde de la misma manera a todas ellas. Si hay hosts virtuales configurados (en este caso no hay ninguno), Apache corre a través de ellos, buscando una IP que corresponda a la conexión entrante. Apache utiliza esa configuración si la encuentra, pero en caso contrario, utiliza la configuración general.

En este momento, al ingresar al servidor, se están dejando registros en la bitácora, si se examina el archivo access_log, se podrá ver la línea:

192.168.123.1 - - [<date-time>] "GET / HTTP/1.1" 200 177

200 es el código de respuesta, que significa que todo está en orden, y 177 es el número de bytes transferidos. A menos de que haya tenido algún error, error_log debe de estar completamente vacío.

Ahora es el momento de crear los documentos HTML que se vayan a utilizar, esto incluye también todos los recursos gráficos que se quiera insertar en las páginas del sitio. La base de la jerarquía de archivos se debe encontrar en el fólder htdocs de Apache, a menos de que se haya cambiado el DocumentRoot.

Apache proporciona un modelo de servidor sencillo pero poderoso. Se puede personalizar completamente a través del uso de módulos, y es lo suficientemente versátil como para acomodar cualquier tipo de sitio que se requiera. Utilizando simples directivas de texto se puede configurar y manejar el servidor.

2.5 - Programación con PHP

Objetivo

Ofrecer al estudiante los conocimientos necesarios para desarrollar aplicaciones basadas en un servidor web por medio de PHP. Al final del curso, el alumno podrá construir proyectos sobre esta plataforma.

Contenido

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 plugins.
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.
Las aplicaciones de servidor suelen ser la opción por la que se opta en la mayoría de las ocasiones para realizar aplicaciones web. La razón es que, al ejecutarse ésta en el servidor y no en la máquina del cliente, éste no necesita ninguna capacidad adicional, como sí ocurre en el caso de querer ejecutar aplicaciones javascript o java. Así pues, cualquier cliente dotado de un navegador web básico puede utilizar este tipo de aplicaciones. Algunos conceptos relacionados con las aplicaciones web son:
PHP
ASP
Perl
CGI
.NET
JSP (Tecnología Java)

Origen de PHP

En 1995, Rasmus Lerdorf desarrolla a través de Perl un conjunto de scripts para el control de acceso a sus páginas personales, al conjunto de scripts se les denominó PHP (Personal Home Pages). La versión 2.0 se desarrolla en 1997 denominada PHP/FI(Forms Interpreter).
En 1998, Zeev Suraski y Andi Gutmans renuevan el lenguaje creando PHP 3.0 y a partir de aquí se le denomina Hypertext Preprocessor. Zeev y Andi crean un nuevo núcleo para el lenguaje (Zend Engine), y se libera la versión 4.0 en 1999. Actualmente en el 2004 se crea el motor Zend 2.0 y con él, se libera PHP 5.0, que introduce un modelo de orientación a objetos que es muy similar al que tiene Java.

¿Cómo instalar Apache?, hay que recordar que si se está utilizando una versión anterior a la 1.3, se tiene que activar el uso de módulos al configurar su instalación, con el parámetro –enable-module=so en la línea de configure.
PHP puede instalarse de dos maneras: como un CGI o como un módulo del servidor web. La opción de CGI no es recomendable; es el sistema operativo el encargado de administrarlo y procesarlo, mientras que de la forma de módulo, PHP es procesado por el mismo servidor web, lo que lo hace más rapido, y mucho más eficiente.

El proceso de instalación es bastante sencillo, una vez que se ha descargado la versión deseada de su sitio web (http://www.php.net), se descomprime el archivo tarball y se configura la instalación de la siguiente forma (como un ejemplo):

```
./configure --prefix=/usr/local/phpxxx --with-apxs2=/usr/local/apachexxx/bin/apxs
make
makeinstall
cp php.ini-dist /usr/local/phpxxx/lib/php.ini
```

Se agregan las siguientes líneas al archivo httpd.conf:

```
AddType application/x-httpd-php .php .phtml
AddType application/x-httpd-php-source .phps
LoadModule phpx_modulemodules/libphpx.so
```

Cuando reinicie Apache, PHP debería funcionar, es posible comprobarlo creando un pequeño programa PHP en el document root llamado info.php que tenga el contenido “<? Phpinfo(); ?>”. La función Phpinfo imprime una tabla con la información sobre las instalaciones de PHP, servidores web, y en caso de existir, gestiones de bases de datos.

PHP es un lenguaje interpretado, lo que significa que siempre permanece en su forma original, es decir, el programa fuente, es el intérprete el que proporciona su traducción al momento de ejecutar cada una de las instrucciones. Es decir que el programa será ejecutado sin necesidad de ser codificado antes, y de que en caso de error, este se detendrá en la línea errónea.
Si se es observador, en el ejemplo sobre la función Phpinfo, el código de PHP está encerrado en los delimitadores <? ?>, existen varias formas de escribirlos: <?php ?>, o <% %> son utilizadas también. Originalmente la forma era un largo <script language=’php’> </script>, que fue remplazado por nemónicos en la versión 2.0 mediante la directiva short_open_tag.

PHP es un lenguaje débilmente tipado, es decir que sus variables no necesitan que se les asocie un tipo cuando se declaran, y pueden almacenar diferentes tipos de información. La sintaxis requerida para denotar variables es preceder su nombre con un signo $. Por ejemplo: $numero. El nombre de las variables puede contener letras, números y _. A propósito, PHP es un lenguaje sensible a mayúsculas y minúsculas, así que $Numero no es lo mismo que $numero.
PHP soporta tres tipos de datos simples:

1. Integer.- representa números que varían desde -2 billones a 2 billones, y se pueden representar en formato decimal, octal y hexadecimal.
2. Float.- sustituye a double desde la versión 4.2.0, simplemente son números con valor decimal.
3. String.- es una cadena de caracteres encerrados por comillas dobles o simples.

Y dos tipos de datos complejos:

1. Array.- como su nombre lo indica, son estructuras de almacenamiento de datos bajo un mismo nombre. El primer elemento del arreglo siempre tiene el subíndice 0, y se pueden crear a través del constructor array().
2. Object.- es una estructura que define características propias (o propiedades) y sus funcionalidades (métodos).

Existen también los arreglos asociativos, que son arreglos que utilizan cadenas de texto como subíndices para sus elementos, lo que significa que cada uno está identificado por el par (llave, valor). Al igual que los arreglos tradicionales, la función array los puede declarar de la forma siguiente: $array = array(‘dia’ => ‘lunes’, ‘costo’ => 2.34);.

Es posible convertir el tipo de variables a uno específico que se desee, este procedimiento es llamado casting, y se logra escribiendo entre paréntesis el tipo deseado antes del nombre de la variable en cuestión.

PHP también maneja constantes, que conservan su valor durante la ejecución de todo el programa. Las constantes predefinidas son PHP_VERSION, PHP_OS, TRUE, y FALSE. Para definir constantes propias, se utiliza la estructura define (“nombre”, valor).

Estructuras de Control

Las estructuras de control permiten que el código se ejecute de una forma no secuencial, también se llaman sentencias de control porque efectivamente son sentencias que se escriben en el código que hacen que este se bifurque (estructuras condicionales) o se repita (estructuras cíclicas).

En PHP existen dos sentencias de estructura condicional: if y switch.
La sintaxis básica de una sentencia IF es:

```
If (expresión){
Sentencias;
}
```

Es muy importante notar que la expresión debe de ser una expresión lógica, es decir, que tenga que ser falsa o verdadera, de otra forma, el programa no funcionará como es debido. Existen variaciones a esta sintaxis, o también adiciones, como el formato if..else:

```
If (expresión){
Sentencias;
}
Else{
Sentencias;
}
```

Es muy explícito en su sintaxis, si la condición que está contenida en If no es verdadera, entonces se ejecutan las expresiones que contiene la cláusula Else. En caso de que se quieran evaluar varias condiciones diferentes, se tiene la estructura Elseif:

```
If (expresión){
Sentencias;
}
Elseif(expresión){
Sentencias;
}
Else{
Sentencias;
}
```

En donde el código evalúa de una por una varias condiciones y ejecuta las sentencias adecuadas. Se pueden tener tantas cláusulas elseif como se requieran. Por último, la forma abreviada de escribir el If básico:

<expresión1>?<expresión2>:<expresión3>

La condición lógica es la primera, esta se evalúa, en caso de que sea verdadera, el código presenta la expresión 2, y de la misma forma, en caso de ser falsa, presenta la tercera expresión.

La sentencia Switch compara un dato con varios posibles valores, cada uno con una secuencia de expresiones consecuentes. Su sintaxis es:

```
switch($variable){
Case valor1:
Sentencias;
break;
Case valor2:
Sentencias;
break;
Case valorN:
Sentencias;
break;
```

```
Default:
Sentencias;
}
```

En este caso, el código analiza el valor de $variable y lo compara con las posibilidades que obtenga en el switch, en este caso, valor1, valor2 y valorN. Si coincide con alguno, ejecuta las sentencias respectivas hasta encontrar un comando break. Esto significa que si se olvida escribir break en un conjunto de sentencias, el código seguirá ejecutando las líneas siguientes sin importar que no correspondan al campo del valor solicitado.

PHP tiene 4 estructuras cíclicas, que permiten ejecutar una porción de código más de una vez, generalmente para recorrer los elementos de un arreglo. Estas estructuras son:

For
For each
While
Do while

For permite realizar un determinado bloque de líneas de código un específico número de veces. Su sintaxis es:

```
For (inicialización; condición; incremento){
Sentencias
}
```

En el argumento de inicialización, generalmente se va a declarar una variable con valor 1. Suponiendo que se quiera que el bloque de sentencias se repita 10 veces, la condición será que la variable de inicialización sea menor a 10, y el incremento simplemente le suma 1 al valor de la misma.

For each se utiliza para recorrer las estructuras de arreglo, obteniendo en cada iteración uno de sus elementos componentes, tiene dos sintaxis:

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

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

While se ejecuta un número indeterminado de veces, siempre y cuando el resultado de comprobar la condición sea verdadero.

```
While(condición){
Sentencias;
}
```

Mientras condición sea verdadera, el código ejecutará las sentencias mostradas.

Do While, a diferencia de while, se ejecuta cuando menos una vez, ya que revisa la condición hasta que termina de ejecutar el código.

```
Do {
Sentencias;
}
While (condición);
```

Existen dos comandos dentro de las estructuras cíclicas: break y continue. Ya se analizó el funcionamiento de break, que detiene la ejecución de la iteración presente; continue hace algo similar, detiene la iteración presente para pasar a la siguiente sin que se efectúen unas instrucciones del ciclo.

Funciones

Una función es una sección separada de código con un propósito específico, a la cual se le asigna un nombre. Ofrece modularidad, es decir, separar al código de un script en partes menores. Su sintaxis en PHP es:

```
Function nombrefuncion(parámetros){
Sentencias;
Return Valor;
}
```

La función recibe parámetros del resto del código, con los cuales trabaja, y devuelve un valor resultado de su funcionamiento. Existen varias formas de pasar argumentos a una función, la primera y más común es hacerlo por valor; es decir, se le pasa una copia de la variable, por lo que al terminar la función, el valor de la misma no es alterado. Esto se logra pasando el parámetro de la siguiente forma:

```
Function iExponente($a){
Return $a*$a;
}
```

En este ejemplo, el valor de $a no se verá afectado por la función iExponente, que regresará su valor al cuadrado.
Otra forma de pasar parámetros es por referencia, que como su nombre lo indica, le pasa a la función una referencia de la variable, en lugar de una simple copia, por lo que el valor de la misma si se ve alterado después de que la función la utiliza. Para pasar parámetros por referencia, se escribe:

```
Function iExponente(&$a){
Return $a*$a;
}
```

Por último, existe la posibilidad de darle a un parámetro un valor por defecto, que le será otorgado en caso de que se llame a la función sin especificar el parámetro. Para lograr esto, la sintaxis correcta es:

```
Function iExponente($a, $exponente=2){
Return $a*$exponente;
}
```

Inclusión de Archivos

La inclusión de archivos es útil para reutilizar código, esencialmente llama a archivos de código al script en cuestión como una librería. Se puede hacer de las formas:

Include "archivo" - la cual incluye y evalúa al archivo cada vez que es interpretada.

Include_once "archivo" - la cual incluye al archivo solamente una vez en el script.

Lo que lleva a la gestión de archivos con el lenguaje PHP. Cuando se manejan archivos en PHP, se pueden hacer cuatro cosas con ellos: abrirlos, cerrarlos, leerlos y escribirlos. Obviamente, se necesita tener los permisos necesarios para todo esto en el sistema operativo. Utilizando apuntadores, los cuales se moverán y modificarán en el curso del script, y mediante los mismos se accederá al archivo en cuestión.

Para abrir un archivo y comenzar a trabajar en él, se tiene la función fopen(nombre_archivo, modo_apertura). Los valores para indicar el modo de apertura son:

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

Otra función útil para la gestión de archivos es fgetc(apuntador), que devuelve el carácter del archivo al que hace referencia el apuntador; si se ha llegado al final del archivo, este devuelve un FALSE.
Esta función puede ser útil para cumplir una serie de sentencias siempre y cuando el archivo no se termine, por ejemplo:

```
while($caracter=fgetc($apuntador)){
sentencias;
}
```

En caso de que se desee trabajar con cadenas y no con caracteres, existe la función fgets(apuntador, [total_car_a_leer]), que devuelve un número de caracteres igual al segundo argumento menos uno. Sin embargo, puede devolver menos en caso de toparse con un salto de línea, o con el final del archivo. Es importante notar que un espacio no detiene a la función, asi que si lo que se desea es obtener una palabra, es mejor utilizar un fgetc en una estructura cíclica.

Similar a fgets, está fread, que recibe exactamente los mismos parámetros que el anterior, y que no deja de leer si encuentra un salto de línea, y también devuelve un total de caracteres a leer.

La función feof(apuntador) devuelve TRUE si es que se ha llegado al final del archivo, comúnmente se utiliza para mantener un bloque de código en ejecución mientras el archivo esté lleno, como una condición de una estructura cíclica. Por ejemplo:

```
while(!feof($apuntador)){
sentencias;:
}
```

A veces es útil tener el archivo dentro de un arreglo, y eso es precisamente lo que la función file hace, recibe de parámetro el nombre del archivo, lee todo su contenido, y finalmente guarda el mismo en un arreglo, donde cada posición contiene una línea del archivo.

La función readfile es similar, y toma el mismo parámetro que file. Sin embargo, lo que hace es que despliega el contenido del archivo por la salida estándar (usualmente el monitor).

PHP cierra todos los archivos que el script utiliza automáticamente al terminar la ejecución del mismo. Sin embargo, no es recomendable tener archivos abiertos que estén ocupando recursos del sistema, cuando no están siendo utilizados, esto reduce no sólo el rendimiento del mismo script, sino también el de la terminal que lo esté ejecutando. Para cerrar un archivo una vez que se termina de usar, se utiliza la función fclose(apuntador).

Existen otras funciones que ayudan a manejar archivos, aunque no se involucren en la gestión directa de los mismos. Funciones que ayudan a conocer características externas del archivo como son:

La función file_exists(nombre_archivo) revisa si existe efectivamente el archivo que recibe como parámetro, y regresa un valor booleano con su resultado.

Si se desea conocer el tamaño de un archivo, existe utilizar la función filesize(nombre_archivo) para recibir el valor en bytes.

También se puede copiar en el sistema de archivos, utilizando la función copy, que recibe 2 parámetros, el nombre del archivo orígen, y el nombre del archivo destino. Además de copiar, devuelve un valor TRUE si la operación fue completada con éxito.

Por último se analizarán a las funciones y métodos que escriben información en los archivos. Las funciones fwrite y fputs son exactamente iguales, y reciben como parámetros el apuntador que se esté utilizando, y una cadena de caracteres, que intentará escribir en el lugar en el que el apuntador esté haciendo referencia en el archivo. Si la operación tiene éxito, devuelven el total de caracteres insertados en el archivo; por el contrario, si es un fallo, devuelven un valor FALSE.

Para devolver el apuntador al principio del archivo, se utiliza la función rewind, que recibe como parámetro el mismo apuntador. Esto permite obviamente volver a comenzar a leer/escribir el archivo desde el principio en caso de que no se desee cerrarlo.

Es posible desplazar el apuntador con más precisión, y esto se logra con la función fseek. La sintaxis que fseek espera es fseek(apuntador, desp, [desde_pos]). Que recibe una copia del apuntador como primer parámetro, un valor numérico desp que le indica cuantos espacios se va a desplazar, y el tercer parámetro, en caso de que se indique ya que es opcional, puede ser SEEK_SET, SEEK_CUR o SEEK_END; cada uno le indica a la función la posición origen desde donde comenzar a contar. Respectivamente, el inicio, la posición actual, o el final. En caso de comenzar desde el final, se tendría que dar el valor desp en números negativos.

La función ftell recibe una copia del apuntador como único parámetro, y devuelve la posición actual del mismo.

PHP es un lenguaje poderoso y sencillo que permite de una manera muy accesible controlar aplicaciones de forma CGI en el servidor. Las tendencias actuales exigen cada vez más de los servidores, y es con estos lenguajes y tecnologías que se va a lograr esa meta.

2.6 - Interacción de WWW con Bases de Datos

Objetivo

Que el estudiante comprenda el funcionamiento y desarrollo de las bases de datos en relación con tecnologías de WWW, que aplique estos conocimientos para desarrollar aplicaciones en el gestor de bases de datos MySQL junto con Apache.

Contenido

Un sistema gestor de bases de datos o SGBD consiste en una colección de datos interrelacionados y un conjunto de programas para acceder a dichos datos. La colección de datos, normalmente denominada base de datos, contiene información relevante para una empresa. El objetivo principal de un SGBD es proporcionar una forma de almacenar y recuperar la información de una base de datos de manera que sea tanto práctica como eficiente.
Los sistemas de bases de datos se diseñan para gestionar grandes cantidades de información. La gestión de los datos implica tanto la definición de estructuras para almacenar la información como la provisión de mecanismos para la manipulación de la información. Además, los sistemas de bases de datos deben proporcionar la fiabilidad de la información almacenada, a pesar de las caídas del sistema o los intentos de acceso

sin autorización. Si los datos van a ser compartidos entre diversos usuarios, el sistema debe evitar posibles resultados anómalos.
Dado que la información es tan importante en la mayoría de las organizaciones, los científicos informáticos han desarrollado un amplio conjunto de conceptos y técnicas para la gestión de los datos.

A lo largo de las ultimas cuatro décadas, el uso de las bases de datos creció en todas las empresas. En los primeros días, muy pocas personas interactuaron directamente con los sistemas de bases de datos, aunque sin darse cuenta interactuaron con bases de datos indirectamente (con los informes impresos como extractos de tarjetas de crédito, o mediante agentes como cajeros de bancos y agentes de reserva de líneas aéreas). Después vinieron los cajeros automáticos y permitieron a los usuarios interactuar con las bases de datos. Las interfaces telefónicas con las computadoras, que en ese entonces eran sistemas de respuesta vocal interactiva, también permitieron a los usuarios manejar directamente las bases de datos. Un llamador podía marcar un número y pulsar teclas del teléfono para introducir información o para seleccionar opciones alternativas, para determinar las horas de llegada o salida, por ejemplo, o para matricularse en una universidad.

La revolución de Internet a finales de la década de 1990 aumentó significativamente el acceso directo del usuario a las bases de datos. Las organizaciones convirtieron muchas de sus interfaces telefónicas a las bases de datos en interfaces Web, y pusieron disponibles en línea muchos de sus servicios. Cuando se accede a un sitio Web, la información personal puede ser recuperada de una base de datos para seleccionar los anuncios que se deberían mostrar. Más aún, los datos sobre los accesos Web pueden ser almacenados sobre una base de datos.
Así, aunque las interfaces de datos ocultan detalles del acceso a las bases de datos, y la mayoría de la gente ni siquiera es consciente de que están interactuando con una base de datos, el acceso a las bases de datos forma una parte esencial de la vida de casi todas las personas actualmente.

Sistemas de bases de datos frente a sistemas de archivos

Considérese parte de una empresa de cajas de ahorros que mantiene información acerca de todos los clientes y cuentas de ahorros. Una manera de mantener la información en una computadora es almacenarla en archivos del sistema operativo. Para permitir a los usuarios manipular la información, el sistema tiene un conjunto de programas de aplicación que manipula los archivos, incluyendo:

- Un programa para efectuar cargos o abonos en una cuenta
- Un programa para añadir una cuenta nueva
- Un programa para calcular el saldo de una cuenta
- Un programa para generar las operaciones mensuales

Estos programas de aplicación se han escrito por programadores de sistemas en respuesta a las necesidades de la organización bancaria.
Si las necesidades se incrementan, se añaden nuevos programas de aplicación al sistema. Por ejemplo, supóngase que las regulaciones de un nuevo gobierno permiten a las cajas de ahorros ofrecer cuentas corrientes. Como resultado se crean nuevos archivos permanentes que contengan información acerca de todas las cuentas corrientes

mantenidas por el banco, y puede ser necesario escribir nuevos programas de aplicación para tratar situaciones que no existan en las cuentas de ahorro, tales como manejar descubiertos. Así, sobre la marcha, se añaden más archivos y programas de aplicación al sistema.
Este sistema de procesamiento de archivos que se utilizó como ejemplo es típico, y se mantiene mediante un sistema operativo convencional. Los registros permanentes son almacenados en varios archivos y se escriben diferentes programas de aplicación para extraer registros y para añadir registros a los archivos adecuados. Antes de la llegada de los sistemas de gestión de bases de datos, las organizaciones usualmente almacenaban la información de esta manera. Sin embargo, hacerlo presenta una larga serie de problemas:

1. **Redundancia e inconsistencia de datos.-** debido a que los archivos y programas de aplicación son creados por diferentes programadores en un largo periodo de tiempo, los diversos archivos tienen probablemente diferentes formatos y los programas pueden estar escritos en diferentes lenguajes. Más aún, la misma información puede estar duplicada en diferentes archivos. Esta redundancia puede conducir a un almacenamiento y coste de acceso más altos. Además, puede también ocasionar una inconsistencia de datos, es decir, las diversas copias de los mismos datos pueden no coincidir.
2. **Dificultad en el acceso a los datos.-** supóngase que uno de los empleados del banco necesita averiguar los nombres de todos los clientes que viven en el distrito postal 28733 de la ciudad. El empleado pide al departamento de procesamiento que genere dicha lista. Debido a que esta petición no fue prevista cuando el sistema original fue diseñado, no hay un programa de aplicación a la mano para satisfacerla. Hay, sin embargo, un programa de aplicación que genera la lista de todos los clientes. El empleado tiene dos opciones: obtener la información que necesita manualmente de la lista de todos los clientes, o bien, pedir al departamento de procesamiento de datos que haga que un programador de sistemas escriba el programa de aplicación necesario. Ambas alternativas son insatisfactorias. Supóngase que se escribe tal programa y que, varios días más tarde, el mismo empleado necesita arreglar esa lista para incluir sólo aquellos clientes que tienen una cuenta con saldo de 10 000 pesos o más. Como se puede esperar, un programa para generar esta lista no existe. De nuevo se tienen dos opciones, ninguna de las cuales es satisfactoria. La cuestión es que el entorno de procesamiento de archivos convencional no permite que los datos sean obtenidos de una forma práctica y eficiente. Se deben desarrollar sistemas de recuperación de datos más interesantes para un uso general.
3. **Aislamiento de datos.-** debido a que los datos están dispersos en varios archivos, y que los archivos pueden estar en diversos formatos, es difícil escribir nuevos programas de aplicación para recuperar los datos apropiados.
4. **Problemas de integridad.-** los valores de los datos almacenados en la base de datos deben satisfacer ciertos tipos de restricciones de consistencia. Es decir, el saldo de una cuenta bancaria no puede nunca ser más bajo de una cantidad predeterminada. Los desarrolladores hacen cumplir esas restricciones en el sistema añadiendo el código apropiado en los diversos programas de aplicación. Sin embargo, cuando se añaden nuevas restricciones, es difícil cambiar los programas para hacer que se cumplan. El problema es complicado cuando las restricciones implican diferentes elementos de datos de diferentes archivos.

5. **Problemas de atomicidad.-** un sistema de computadoras, como cualquier otro dispositivo mecánico o eléctrico, está sujeto a fallo. En muchas aplicaciones es crucial asegurar que, una vez que un fallo ha ocurrido y se ha detectado, los datos se restauran al estado de consistencia que existía antes del fallo. Considérese un programa para transferir 50 pesos de una cuenta A a una cuenta B. si ocurre un fallo del sistema durante su ejecución, es posible que los 50 fueran eliminados de la cuenta A, pero no abonados a la cuenta B, resultando una base de datos inconsistente. Claramente, es esencial para la consistencia de la base de datos que ambos, el abono y el cargo tengan lugar, o que ninguno lo tenga.
6. **Anomalías en el acceso concurrente.-** conforme se ha ido mejorando el conjunto de ejecución de los sistemas y ha sido posible una respuesta en tiempo más rápida, muchos sistemas han ido permitiendo a múltiples usuarios actualizar los datos simultáneamente. En tales sistemas un entorno de interacción de actualizaciones concurrentes puede dar lugar a datos inconsistentes. Considérese una cuenta bancaria A que contiene 500 pesos. Si dos clientes retiran fondos (por ejemplo, 50 y 100 pesos respectivamente) de la cuenta A en aproximadamente el mismo tiempo, el resultado de las ejecuciones concurrentes puede dejar la cuenta en un estado incorrecto o inconsistente. Supóngase que los programas se ejecutan para cada retirada y escriben el resultado después. Si los dos programas funcionan concurrentemente, pueden leer ambos el valor $500 y escribir después $450 y $400. dependiendo de cuál escriba el último valor, la cuenta puede contener $400 o $450, en lugar del valor correcto de $350. Para protegerse de esta posibilidad, el sistema debe mantener alguna forma de supervisión. Sin embargo, ya que se puede acceder a los datos desde muchos programas de aplicación diferentes que no han sido previamente coordinados, la supervisión es difícil de proporcionar.
7. **Problemas de seguridad.-** no todos los usuarios de un sistema de bases de datos deberían poder acceder a todos los datos. Por ejemplo, en el sistema bancario, el personal de nóminas necesita ver sólo esa parte de la base de datos que tiene información de varios empleados del banco. No necesita acceder a la información acerca de las cuentas de clientes. Como los programas de aplicación se añaden al sistema de forma ad hoc, es difícil asegurar tales restricciones de seguridad.

Estas dificultades, entre otras, han motivado el desarrollo de los sistemas de bases de datos. Un sistema de bases de datos es una colección de archivos interrelacionados y un conjunto de programas que permitan a los usuarios acceder y modificar estos archivos. Uno de los propósitos principales de un sistema de bases de datos es proporcionar a los usuarios una visión abstracta de los datos. Es decir, el sistema esconde ciertos detalles de cómo se almacenan y mantienen los datos.
Para que el sistema sea útil debe recuperar los datos eficientemente. Esta preocupación ha conducido al diseño de estructuras de datos complejas para la representación de los datos en la base de datos. Como muchos usuarios de sistemas de bases de datos no están familiarizados con computadoras, los desarrolladores esconden la complejidad a los mismos a través de varios niveles de abstracción para simplificar la interacción de los usuarios con el sistema:

- **Nivel físico.-** el nivel más bajo de abstracción describe cómo se almacenan realmente los datos. En el nivel físico se describen en detalle las estructuras de datos de bajo nivel.
- **Nivel lógico.-** el siguiente nivel más alto de abstracción describe que datos se almacenan en la base de datos y que relaciones existen entre estos datos. La base de datos completa se describe así en términos de un número pequeño de estructuras relativamente simples. Aunque la implementación de estructuras simples en el nivel lógico puede involucrar estructuras complejas del nivel físico, los usuarios del nivel lógico no necesitan preocuparse de esta complejidad. Los administradores de bases de datos, que deben decidir la información que se mantiene en la base de datos, usan el nivel lógico de abstracción.
- **Nivel de vistas.-** el nivel más alto de abstracción describe solo parte de la base de datos completa. A pesar del uso de estructuras más simples en el nivel lógico, queda algo de complejidad, debido a la variedad de información almacenada en una gran base de datos. Muchos usuarios del sistema de base de datos no necesitan toda esta información. En su lugar, tales usuarios necesitan acceder sólo a una parte de la base de datos. Para que su interacción con el sistema se simplifique, se define la abstracción del nivel de vistas. El sistema puede proporcionar muchas vistas para la misma base de datos.

Una analogía con el concepto de tipos de datos en lenguajes de programación puede clarificar la distinción entre los niveles de abstracción. La mayoría de lenguajes de programación de alto nivel soportan la estructura de tipo registro. Por ejemplo, en un lenguaje estructurado:

```
Type cliente = record
        Nombre-cliente : string;
        Id-cliente : string;
        Calle-cliente : string;
        Ciudad-cliente : string;
End;
```

Este código, presentado en un lenguaje al estilo Pascal, define un nuevo registro llamado cliente con cuatro campos. Cada campo tiene un nombre y un tipo asociado a él. Volviendo al ejemplo del banco, puede tener varios tipos de registros:

Cuenta: con los campos número y saldo.
Empleado: con los campos nombre y sueldo.

En el nivel físico, un registro cliente, cuenta o empleado se puede describir como un bloque de posiciones almacenadas consecutivamente. El compilador del lenguaje esconde este nivel de detalle a los programadores. Análogamente, el sistema de bases de datos esconde muchos de los detalles de almacenamiento de bases de datos. Los administradores de bases de datos pueden ser conscientes de ciertos detalles de la organización física de los datos.

En el nivel lógico, cada registro de este tipo se describe mediante una definición de tipo, y se define la relación entre estos tipos de registros. Los programadores, cuando usan un lenguaje de programación, trabajan en este nivel de abstracción. De forma similar,

los administradores de bases de datos trabajan habitualmente en este nivel de abstracción.

Finalmente, en el nivel de vistas, los usuarios de computadoras ven un conjunto de programas de aplicación que esconden los detalles de los tipos de datos. Análogamente, en el nivel de vistas se definen varias vistas de una base de datos, y los usuarios de la misma ven únicamente estas vistas. Además de esconder detalles del nivel lógico de la base de datos, las vistas también proporcionan un mecanismo de seguridad para evitar que los usuarios accedan a ciertas partes de la base de datos.

Modelos de los datos

Bajo la estructura de la base de datos se encuentra el modelo de datos, una colección de herramientas conceptuales para describir los datos, las relaciones, la semántica y las restricciones de consistencia. Para ilustrar el concepto de un modelo de datos, se describirán dos tipos de ellos: el modelo entidad relación, y el modelo relacional. Los modelos que se han propuesto se clasifican en tres grupos diferentes: modelos lógicos basados en objetos, modelos lógicos basados en registros y modelos físicos.

El modelo de datos entidad relación (E-R) está basado en una percepción del mundo real que consta de una colección de objetos básicos, llamados entidades y de relaciones entre estos objetos. Una entidad es una "cosa" u "objeto" en el mundo real que es distinguible de otros objetos. Por ejemplo, una persona es una entidad, las cuentas bancarias también pueden ser consideradas entidades.
Las entidades se describen en una base de datos mediante un conjunto de atributos. Por ejemplo, los atributos número-cuenta y saldo describen una cuenta particular de un banco y pueden ser atributos del conjunto de entidades cuenta. Análogamente, los atributos nombre-cliente y ciudad-cliente pueden describir una entidad cliente.
Un atributo extra, id-cliente, se usa para identificar unívocamente a los clientes. Se debe asignar un identificador único de cliente a cada uno.
Una relación es una asociación entre varias entidades. Por ejemplo, una relación impositor asocia un cliente con cada cuenta que tiene. El conjunto de todas las entidades del mismo tipo, y el conjunto de todas las relaciones del mismo tipo, se denominan respectivamente conjunto de entidades y conjunto de relaciones.
La estructura lógica general de una base de datos se puede expresar gráficamente mediante un Diagrama Entidad Relación (DER), que consta de los siguientes componentes:

- **Rectángulos.-** que representan conjuntos de entidades
- **Elipses.-** que representan atributos
- **Rombos.-** que representan relaciones entre conjuntos de entidades
- **Líneas.-** que unen los atributos con los conjuntos de entidades y los conjuntos de entidades con las relaciones

Cada componente se etiqueta con la entidad o relación que representa.

En el modelo relacional se utiliza un grupo de tablas para representar los datos y las relaciones entre ellos. Cada tabla esta compuesta por varias columnas, y cada columna tiene un nombre único.

El modelo relacional es un ejemplo de un modelo basado en registros. Los modelos basados en registros se denominan así porque la base de datos se estructura en registros de formato fijo de varios tipos. Cada tabla contiene registros de un tipo particular. Cada tipo de registro define un número fijo de campos, o atributos. Las columnas de la tabla corresponden a los atributos del tipo de registro.
No es difícil ver como se pueden almacenar las tablas en archivos. Por ejemplo, un carácter especial se puede usar para delimitar los diferentes atributos de un registro, y otro carácter especial se puede usar para delimitar registros. El modelo relacional oculta tales detalles de implementación de bajo nivel a desarrolladores de bases de datos y usuarios.
El modelo de datos relacional es el modelo de datos más ampliamente usado, y una amplia mayoría de sistemas de bases de datos actuales se basan en el modelo relacional. Se encuentra a un nivel de abstracción inferior al modelo de datos ER. Los diseños de bases de datos a menudo se realizan en el modelo E-R, y después se traducen al modelo relacional.

Un sistema de bases de datos proporciona un lenguaje de definición de datos para especificar el esquema de la base de datos y un lenguaje de manipulación de datos para expresar las consultas a la base de datos y las modificaciones. En la práctica, los lenguajes de definición y manipulación de datos no son dos lenguajes separados; en su lugar simplemente forman partes de un único lenguaje de bases de datos, tal como SQL, que es ampliamente usado.
Un esquema de base de datos se especifica mediante un conjunto de definiciones expresadas mediante un lenguaje especial llamado lenguaje de definición de datos (LDD). Por ejemplo, la siguiente instrucción en el lenguaje SQL define la tabla cuenta:

Create table cuenta (numero-cuenta char(10), saldo integer)

La ejecución de la instrucción LDD anterior crea la tabla cuenta. Además, actualiza un conjunto especial de tablas denominado diccionario de datos o directorio de datos.
Un diccionario de datos contiene metadatos, es decir, datos acerca de los datos. El esquema de una tabla es un ejemplo de metadatos. Un sistema de base de datos consulta el diccionario de datos antes de leer o modificar los datos reales.

La manipulación de datos es:

- La recuperación de información almacenada en la base de datos.
- La inserción de información nueva en la base de datos.
- El borrado de información de la base de datos.
- La modificación de información almacenada en la base de datos.

Un lenguaje de manipulación de datos, o LMD, es un lenguaje que permite a los usuarios acceder o manipular los datos organizados mediante el modelo de datos apropiado. Hay dos tipos básicamente:

1. LMDs procedimentales.- requieren que el usuario especifique qué datos se necesitan y cómo obtener esos datos.
2. LMDs declarativos.- requieren que el usuario especifique qué datos se necesitan sin especificar cómo obtenerlos.

Los LMDs declarativos son más fáciles de aprender y usar que los procedimentales. Sin embargo, como el usuario no especifica cómo conseguir los datos, el sistema de bases de datos tiene que determinar un medio eficiente de acceder a ellos. El componente LMD de SQL es no procedimental.

Una consulta es una instrucción de solicitud para recuperar información. La parte de un LMD que implica recuperación de información se llama lenguaje de consultas. Aunque técnicamente sea incorrecto, en la práctica se usan los terminos lenguaje de consultas y lenguaje de manipulación de datos como sinónimos.
Esta consulta en el lenguaje SQL encuentra el nombre del cliente cuyo identificador es 19.283.746:

```
Select cliente.nombre-cliente
From cliente
Where cliente.idcliente = '19.283.746'
```

La consulta especifica que las filas de (from) la tabla cliente donde (where) el id-cliente es 19.283.746 se debe recuperar, y que se debe mostrar el atributo nombre-cliente de estas filas. Las consultas pueden involucrar información de más de una tabla.

Los programas de aplicación son programas que se usan para interactuar con las bases de datos. Los programas de aplicación se escriben usualmente en un lenguaje anfitrión, tal como COBOL, C, C++ o Java.
Para acceder a la base de datos, las instrucciones LMD necesitan ser ejecutadas desde el lenguaje anfitrión. Hay dos maneras de hacerlo:

1. proporcionando la interfaz del programa de aplicación (conjunto de procedimientos) que se pueden usar para enviar instrucciones LMD y LDD a la base de datos, y recuperar los resultados. El estándar de conectividad abierta de bases de datos Open Database Connectivity, u ODBC, definido por Microsoft para el uso con el lenguaje C es un estándar de interfaz de programas de aplicación usado comúnmente. El estándar conectividad de Java con bases de datos, o JDBC (Java Database Connectivity) proporciona características correspondientes para el lenguaje Java.
2. extendiendo la sintaxis del lenguaje anfitrión para incorporar llamadas LMD dentro del programa del lenguaje anfitrión. Usualmente, un carácter especial precede a las llamadas LMD, y un preprocesador, denominado precompilador LMD, convierte las instrucciones LMD en llamadas normales a procedimientos en el lenguaje anfitrión.

Un objetivo principal de un sistema de bases de datos es recuperar información y almacenar nueva información en la base de datos. Las personas que trabajan con una base de datos se pueden catalogar como usuarios de bases de datos o como administradores de bases de datos.

Hay cuatro tipos diferentes de un sistema de bases de datos, diferenciados por la forma en que ellos esperan interactuar con el sistema. Se han diseñado diferentes tipos de interfaces de usuario para diferentes tipos de usuarios.

- **Usuarios normales.-** son usuarios no sofisticados que interactúan con el sistema mediante la invocación de alguno de los programas de aplicación permanentes que se ha escrito previamente. Por ejemplo, un cajero bancario que necesita transferir 50 pesos de la cuenta A a la cuenta B invoca un programa llamado transferir. Este programa pide al cajero el importe de dinero a transferir, la cuenta de la que el dinero va a ser transferido y la cuenta a la que será transferido. La interfaz de usuario normal para los usuarios normales es una interfaz de formularios, donde el usuario puede rellenar los campos apropiados del formulario. Los usuarios normales pueden también simplemente leer informes generados de la base de datos.
- **Programadores de aplicaciones.-** son profesionales informáticos que escriben programas de aplicación. Los programadores de aplicaciones pueden elegir entre muchas herramientas para desarrollar interfaces de usuario. Las herramientas de desarrollo rápido de aplicaciones (DRA) son herramientas que permiten al programador construir formularios e informes sin escribir un programa. Hay también tipos especiales de lenguajes de programación que combinan estructuras de control imperativo (por ejemplo, para ciclos for, ciclos while, e instrucciones if-else) con instrucciones del lenguaje de manipulación de datos. Estos lenguajes, llamados a veces lenguajes de cuarta generación, a menudo incluyen características especiales para facilitar la generación de formularios y la presentación de datos en pantalla. La mayoría de los sistemas de bases de datos comerciales incluyen un lenguaje de cuarta generación.
- **Los usuarios sofisticados.-** interactúan con el sistema sin programas escritos. En su lugar, ellos forman sus consultas en un lenguaje de consulta de bases de datos. Cada una de estas consultas se envía al procesador de consultas, cuya función es transformar instrucciones LMD a instrucciones que el gestor de almacenamiento entienda. Los analistas que envían las consultas para explorar los datos en la base de datos entran en esta categoría. Las herramientas de procesamiento analítico en línea (OLAP, Online Analytical Processing) simplifican la labor de los analistas permitiéndoles ver resúmenes de datos de formas diferentes. Por ejemplo, un analista puede ver las ventas totales por región o por producto. Las herramientas también permiten al analista seleccionar regiones específicas, examinar los datos con más detalle o examinar los datos con menos detalle. Otra clase de herramientas para los analistas son las herramientas de recopilación de datos, que les ayudan a encontrar ciertas clases de patrones de datos.
- **Usuarios especializados.-** son usuarios sofisticados que escriben aplicaciones de bases de datos especializadas que no son adecuadas en el marco de procesamiento de datos tradicional. Entre estas paliaciones están los sistemas de diseño asistido por computadora, sistemas de bases de conocimientos y sistemas expertos, sistemas que almacenan los datos con tipos de datos complejos y sistemas de modelado del entorno.

Una de las principales razones de usar un sistema de gestión de bases de datos es tener un control centralizado tanto de los datos como de los programas que acceden a esos datos. La persona que tiene este control central sobre el sistema se llama administrador de la base de datos. Las funciones del ABD incluyen las siguientes:

- **Definición del esquema.-** el ABD crea el esquema original de la base de datos escribiendo un conjunto de instrucciones de definición de datos en el LDD.

- **Definición de la estructura y del método de acceso.**
- **Modificación del esquema y de la organización física.-** los ABD realizan cambios en el esquema y en la organización física para reflejar las necesidades cambiantes de la organización, o para alterar la organización física para mejorar el rendimiento.
- **Concesión de autorización para el acceso a los datos.** La concesión de diferentes tipos de autorización permite al administrador de la base de datos determinar a que partes de la base de datos puede acceder cada usuario. La información de autorización se mantiene en una estructura del sistema especial que el sistema de base de datos consulta cuando se intenta el acceso a los datos en el sistema.
- **Mantenimiento rutinario.**- algunos ejemplos de actividades rutinarias de mantenimiento del administrado de la base de datos son:
    1. Copia de seguridad periódica de la base de datos, bien sobre cinta, o sobre servidores remotos, para prevenir la pérdida de datos en caso de desastres como inundaciones.
    2. asegurarse de que haya suficiente espacio libre para las operaciones normales y aumentar el espacio en disco según sea necesario.
    3. supervisión de los trabajos que se ejecuten en la base de datos y asegurarse de que el rendimiento no se degrada por tareas muy costosas iniciadas por algunos usuarios.

Los sistemas de bases de datos comerciales necesitan de un lenguaje de consultas cómodo para el usuario. SQL utiliza una combinación de álgebra relacional y construcciones del cálculo relacional.
Aunque SQL se considere un lenguaje de consultas, contiene muchas otras capacidades además de la consulta en bases de datos. Incluye características para definir la estructura de los datos, para la modificación de los datos en la base de datos y para la especificación de restricciones de seguridad.

El lenguaje SQL tiene varios componentes:

- **Lenguaje de definición de datos.-** el LDD de SQL proporciona órdenes para la definición de esquemas de relación, borrado de relaciones, creación de índices y modificación de esquemas de relación.
- **Lenguaje interactivo de manipulación de datos.**- el LMD de SQL incluye un lenguaje de consultas, basado tanto en el álgebra relacional como en el cálculo relacional de tuplas. Incluye también órdenes para insertar, borrar y modificar tuplas de la base de datos.
- **Definición de vistas.-** el LDD de SQL incluye órdenes para la definición de vistas.
- **Control de transacciones.**- SQL incluye órdenes para la especificación del comienzo y final de transacciones.
- **SQL incorporado y SQL dinámico.**- SQL dinámico e incorporado define cómo se pueden incorporar las instrucciones SQL en lenguajes de programación de propósito general.
- **Integridad.**- el LDD de SQL incluye órdenes para la especificación de las restricciones de integridad que deben satisfacer los datos almacenados en la base de datos. Las actualizaciones que violen las restricciones de integridad se rechazan.

- **Autorización.**- el LDD de SQL incluye órdenes para especificar derechos de acceso para las relaciones y vistas.

Una base de datos relacional consiste en un conjunto de relaciones, a cada una de las cuales se le asigna un nombre único. SQL permite el uso de valores nulos para indicar que el valor o bien es desconocido, o no existe. Se fijan criterios que permiten al usuario especificar a qué atributos no se puede asignar valor nulo.
La estructura básica de una expresión SQL consiste en tres cláusulas: select, from y where.

La cláusula select corresponde a la operación proyección del álgebra relacional. Se usa para listar los atributos deseados del resultado de una consulta.

La cláusula from corresponde a la operación producto cartesiano del álgebra relacional. Lista las relaciones que deben ser analizadas en la evaluación de la expresión.

La cláusula where corresponde al predicado selección del álgebra relacional. Es un predicado que engloba a los atributos de las relaciones que aparecen en la cláusula from.

En el presente, las bases de datos son completamente indispensables para el desarrollo y funcionamiento no sólo de empresas, sino de servicios públicos y privados. En conjunto con las tecnologías web, y utilizando la capacidad de SQL para integrarse con lenguajes de aplicación, los servicios en Internet pueden fácilmente presentar bases de datos de forma natural. A finales de los 90, la fuerza de los negocios estrictamente basados en sitios de Internet ya era considerable. Esta tendencia se ha balanceado, y ahora es impensable que una empresa importante no tenga servicios web.

2.7 - Introducción a la Seguridad en Cómputo

Objetivo

Conocer las bases en las que se funda la seguridad informática, el funcionamiento principal de las firmas y la criptografía.

Contenido

Se entiende como seguridad una característica de cualquier sistema (informático o no) que indica que ese sistema está libre de peligro, daño o riesgo. Se entiende como peligro o daño todo aquello que pueda afectar su funcionamiento directo o los resultados que se obtienen del mismo. Para la mayoría de los expertos el concepto de seguridad en la informática es utópico porque no existe un sistema 100% seguro. Para que un sistema se pueda definir como seguro se deben de dotar las siguientes características al mismo:

1. Que se opera como se espera que lo haga.
2. Que es ajeno a todo riesgo.
3. Que es ajeno a toda amenaza.
4. Que no posee vulnerabilidades.
5. Que es confiable
6. Que funciona sin fallo.

Estos seis puntos se pueden resumir en tres principales características:

- Confidencialidad
- Integridad
- Disponibilidad

Y dependiendo de las fuentes de amenazas que estén presentes, se puede dividir la seguridad en lógica y física.

Los pasos que se toman tradicionalmente para realizar el objetivo de tener un sistema seguro, es diseñar el mismo para que los agentes, ya sean usuarios o programas sólo puedan tomar acciones que hayan sido explícitamente permitidas. Esto incluye especificar e implementar una póliza de seguridad. Las acciones en cuestión pueden ser reducidas a operaciones de acceso, modificación y borrado. La seguridad de computadoras puede ser vista como un sub-campo de la ingeniería de seguridad, que además de contemplar a la anterior, observa campos más amplios sobre seguridad en general.

En un sistema seguro, los usuarios autorizados del sistema son capaces de hacer lo que deberían hacer. Aunque es posible asegurar un sistema utilizando medidas extremas, como el aislamiento físico de la terminal, por ejemplo; debido al requerimiento anterior, esto no convierte al mismo en un sistema seguro.

Es importante distinguir las técnicas utilizadas para incrementar la seguridad de un sistema del estado de seguridad. Particularmente, sistemas que contengan fallas fundamentales en su diseño de seguridad no pueden ser hechos seguros sin comprometer su funcionalidad. Consecuentemente, muchos sistemas de computadoras no pueden ser seguros incluso después de la aplicación de medidas extensivas; mientras más seguros se hacen, más detrimentos tienen en funcionalidad.

Términos Relacionados

Activo

Un activo es un recurso del sistema de información o algo relacionado con este. Es necesario para que la organización que lo emplea, así como todos sus componentes, funcionen y se desarrollen.

Amenaza

El término se refiere a un evento o circunstancia, ya sea física o lógica, que pueda causar daño al sistema a través de una vulnerabilidad.

Vulnerabilidad

Ausencia de una contramedida, o tal vez una debilidad de la misma. Comúnmente conocida como hoyo o hueco, generalmente se originan en el diseño, la implementación o en los procedimientos para operar y administrar el sistema.
Es la predisposición de un sistema a ser afectado por un agente perturbador.

Riesgo

Es la posibilidad de que una vulnerabilidad sea afectada. El riesgo presente en un sistema es calculado con la simple ecuación riesgo= peligro x exposición x vulnerabilidad.

Exposición

El grado de exposición es el nivel de afectación, o número de personas, bienes o módulos (por ejemplo) que serían afectados en caso de haber una falla en el sistema de seguridad.

Peligro

Es la probabilidad de que se presente una amenaza.

Existen dos diferentes visiones de la seguridad de cómputo. Una se enfoca mayoritariamente en amenazas externas, y generalmente trata al sistema mismo como confiable.
La segunda considera al sistema como in confiable, y lo rediseña para hacerlo más seguro de varias maneras.

Esta última técnica alienta la separación de privilegios, donde una entidad solo tiene los privilegios que se necesitan para su función. De esa forma, aún si un atacante ha subvertido una parte del sistema, la más fina seguridad se encarga de que sea muy difícil subvertir el resto.

Más aún, al romper el sistema en varios componentes más pequeños, la complejidad de los componentes individuales se reduce, abriendo la posibilidad de usar técnicas tales como "prueba de teoremas automática" (ATP) para probar que los sub-sistemas de software cruciales estén en correcto estado. Donde las pruebas formales no son posibles, el uso riguroso de la revisión de código y la prueba de unidades puede ser utilizado para hacer que los módulos sean tan seguros como sea posible.

El diseño debería usar "defensa profunda", un paradigma en donde más de un sub-sistema necesita ser derrotado para comprometer la seguridad del sistema, y de la información que contiene. Los sub-sistemas deberían configurarse con miras a la seguridad por defecto, y donde sea posible, ser designados a un fallo seguro, más que a un fallo inseguro. Idealmente, un sistema seguro requiere una decisión deliberada, consciente y educada de parte de las autoridades legítimas para hacerse inseguro. Lo que constituye dicha decisión, y lo que hace legítimas a las autoridades es campo obviamente controversial.

Además, la seguridad no debería ser analizada como un campo absoluto de todo o nada. Los diseñadores y operadores de sistemas deben asumir que a largo plazo, es inevitable que suceda una falla de seguridad. Bitácoras de audición completas deben ser llevadas sobre el funcionamiento del sistema, para que cuando el fallo ocurra, el mecanismo y la extensión de la falla puedan ser determinadas. Guardar estas bitácoras remotamente, donde puedan ser leídas, puede mantener a los intrusos lejos de tapar sus huellas.

Finalmente, una apertura total ayuda a asegurar que cuando se encuentren bugs, la ventana de vulnerabilidad quede tan pequeña como sea posible.

Una política de seguridad generalmente se ocupa exclusivamente a asegurar los derechos de acceso a los datos y recursos con las herramientas de control y mecanismos de identificación. Estos mecanismos permiten saber que los operadores tienen sólo los permisos que se les dio.
La seguridad informática debe ser estudiada para que no impida el trabajo de los operadores en lo que les es necesario y que puedan utilizar el sistema informático con toda confianza. Por eso en lo referente a elaborar una política de seguridad, conviene:

- Elaborar reglas y procedimientos para cada servicio de la organización
- Definir las acciones a emprender y elegir las personas a contactar en caso de Detectar una posible intrusión
- Sensibilizar los operadores con los problemas ligados con la seguridad de los sistemas informáticos

Los derechos de acceso de los operadores deben ser definidos por los responsables jerárquicos y no por los administradores informáticos, los cuales tienen que conseguir que los recursos y derechos de acceso sean coherentes con la política de seguridad definida. Además, como el administrador suele ser el único en conocer perfectamente el sistema, tiene que derivar a la directiva cualquier problema e información relevante sobre la seguridad, y eventualmente aconsejar estrategias a poner en marcha, así como ser el punto de entrada de la comunicación a los trabajadores sobre problemas y recomendaciones en término de seguridad.

Una vez que la programación y el funcionamiento de un dispositivo de almacenamiento (o transmisión) de la información se consideran seguras, todavía deben ser tenidos en cuenta los circunstancias "no informáticas" que pueden afectar a los datos, las cuales son a menudo imprevisibles o inevitables, de modo que la única protección posible es la redundancia (en el caso de los datos) y la descentralización -por ejemplo mediante estructura de redes- (en el caso de las comunicaciones). Estos fenómenos pueden ser causados por:

1. Un operador: causa del mayor problema ligado a la seguridad de un sistema informático (por que no le importa, no se da cuenta o a propósito).
2. Programas maliciosos: programas destinados a perjudicar o a hacer un uso ilícito de los recursos del sistema es instalado (por inatención o maldad) en el ordenador abriendo una puerta a intrusos o bien modificando los datos. Estos programas pueden ser un virus informático, un gusano informático, un troyano, una bomba lógica o un programa espía o Spyware.
3. Un intruso: persona que consigue acceder a los datos o programas de los cuales no tiene acceso permitido (cracker, defacer, script kiddie o Script boy, viruxer, etc.)
4. Un siniestro (robo, incendio, por agua): una mala manipulación o una mala intención derivan a la pérdida del material o de los archivos.
5. El personal interno de Sistemas: Las pujas de poder que llevan a disociaciones entre los sectores y soluciones incompatibles para la seguridad informática.

La primera instancia en que se consideró importante la seguridad informática fue en el desarrollo de un sistema operativo llamado Multics. Multics fue notable por muchas cosas, fue uno de los primeros en considerar a los archivos y procesos como similares, y posiblemente el primer sistema operativo que fue diseñado como un sistema seguro desde el comienzo. A pesar de ello, su seguridad fue vencida, no una, sino varias veces. Sin embargo esto originó más estudios sobre la seguridad informática; estudios que fueron los precursores de las técnicas de ingeniería de seguridad modernas.

Técnicas Modernas de Seguridad

Las siguientes son algunas de las técnicas que pueden ser usadas en la creación de sistemas seguros. Estas técnicas, aunque útiles, no aseguran la seguridad por si mismas. Una máxima popular es "Un sistema de seguridad es tan fuerte como su elemento más débil".

- **Prueba Automatizada de Teoremas**.- Junto a otras herramientas de verificación, puede permitir que se pruebe matemáticamente a los algoritmos y el código ejecutado para alcanzar sus expectativas. De esta forma, "microkernels" pueden ser escritos para asegurarse de que no tienen ningún bug. Dos ejemplos de esta tecnología son EROS y Coyotos.

Un sistema operativo mayor, capaz de proveer una interfaz de procesamiento de aplicación (API por sus siglas en inglés), como POSIX, puede ser construido en un microkernel utilizando pequeños servidores API que corran programas pequeños. Si uno de estos servidores tiene un bug, el kernel y los otros servidores no son afectados.

- **Técnicas Criptográficas.-** pueden ser usadas para defender información que esté en tránsito entre sistemas, reduciendo así la probabilidad de que se intercepte o se modifiquen los datos intercambiados entre sistemas.
- **Técnicas de Autenticación Fuertes.-** pueden ser usadas para asegurar que los nodos de una red de comunicación sean quien dicen que son.
- **Criptoprocesadores Seguros.-** implementan técnicas de seguridad física al sistema de seguridad en la computadora.
- **Cadena de Confianza.-** este método se puede usar para asegurar que todo el software haya sido certificado como auténtico por sus propios diseñadores.
- **Control de Acceso Mandatario.-** asegura que el acceso privilegiado sea retirado cuando los mismos privilegios sean revocados. Por ejemplo, borrar una cuenta de usuario debería también detener los procesos asociados con los privilegios del mismo.
- **Lista de Capacidades y Accesos.-** pueden ser usadas para asegurar separación de privilegios y acceso de control mandatario.
- **Abstinencia de Uso de Fallas.-** en un sistema de producción donde una aplicación no tiene manera de arreglar fallas de seguridad conocidas, es recomendable dejar de usarlo hasta que el arreglo esté disponible. Fallas que sean públicas son el principal punto de entrada para gusanos que automáticamente ingresan al sistema y lo utilizan para propagarse por la red.
- **Backups.-** los respaldos son una manera de asegurar la información; básicamente son una copia de todos los archivos de la computadora mantenidos en algún otro lugar. Estos archivos se guardan en discos duros, o medios extraíbles. Pueden ser guardados en una multitud de lugares, generalmente cajas

fuertes que sean resistentes a fuego, agua y calor, o en una localidad diferente a los originales. Los respaldos además son muy importantes por otras razones además de la seguridad; desastres naturales o externos al sistema.

- **Software Antivirus.-** consiste en programas que tratan de identificar, bloquear y eliminar virus de computadoras y otros tipos de software malicioso.
- **Firewalls.-** son sistemas que ayudan a proteger las computadoras y las redes de computadoras de ataques e intrusiones subsecuentes restringiendo el tráfico de la red que puede pasar a través de ellos. Se basan en reglas que el administrador del sistema les define.
- **Autorización de acceso.-** restringe el acceso al sistema a un grupo de usuarios a través del uso de programas de autenticación. Estos sistemas pueden proteger ya sea a la computadora en su totalidad – mediante el uso de una pantalla de login – o servicios individuales, tales como un servidor FTP. Hay muchas formas de identificar y autenticar usuarios, tales como contraseñas, tarjetas de identificación, o sistemas biométricos, en resumen, existen los siguientes mecanismos de autenticación:
    - **Algo que se sabe.-** se basan en claves de acceso, son fáciles de usar y no requieren de hardware especial; como contraseñas, nombres de usuario, etc.
    - **Algo que se tiene.-** necesitan un hardware especial que identifique un objeto que tenga el usuario, generalmente tarjetas o gafetes con la clave de usuario.
    - **Algo que se es.-** necesitan un hardware biométrico. Identifican al usuario basándose en sus características físicas; su rostro, su huella digital, su iris, etc.
- **Encriptación.-** se usa para proteger un mensaje de los ojos de otros. Puede realizarse en un número de formas; cambiando de lugar los caracteres, reemplazándolos con otros, y aún quitando algunos del mensaje. Esto tiene que hacerse en combinación para asegurar realmente la encriptación. Más adelante se analizarán algunos métodos de encriptación.
- **Sistemas de Detección de Intrusos.-** pueden escanear una red para encontrar personas que no estén en un lugar autorizado, o que estén haciendo cosas que no deberían, por ejemplo, tratar muchas contraseñas diferentes para acceder a la red.
- **Percepción de Ingeniería Social.-** mantener a los empleados en perspectiva de los peligros de la ingeniería social y/o mantener una política que prevenga la misma puede reducir considerablemente los ataques a la red y los servidores.

Firmas Digitales

La firma digital de un documento es el resultado de aplicar cierto algoritmo matemático, denominado función hash, a su contenido. Esta función asocia un valor dentro de un conjunto finito (generalmente los números naturales) a su entrada. Cuando la entrada es un documento, el resultado de la función es un número que identifica casi unívocamente al texto. Si se adjunta este número al texto, el destinatario puede aplicar de nuevo la función y comprobar su resultado con el que ha recibido. No obstante esto presenta algunas dificultades.

Para que sea de utilidad, la función hash debe satisfacer dos importantes requisitos. Primero, debe ser difícil encontrar dos documentos cuyo valor para la función "hash"

sea idéntico. Segundo, dado uno de estos valores, debería ser difícil recuperar el documento que lo produjo.
Algunos sistemas de cifrado de clave pública se pueden usar para firmar documentos. El firmante cifra el documento 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.
Existen funciones "hash" específicamente designadas para satisfacer estas dos importantes propiedades. SHA y MD5 son dos ejemplos de este tipo de algoritmos. Para usarlos un documento se firma con una función "hash", cuyo resultado es la firma. Otra persona puede comprobar la firma aplicando la misma función a su copia del documento y comparando el resultado con el del documento original. Si concuerdan, es casi seguro que los documentos son idénticos.
Claro que el problema está en usar una función "hash" para firmas digitales que no permita que un "atacante" interfiera en la comprobación de la firma. Si el documento y la firma se enviaran descifrados, este individuo podría modificar el documento y generar una firma correspondiente sin que lo supiera el destinatario. Si sólo se cifrara el documento, un atacante podría manipular la firma y hacer que la comprobación de ésta fallara. Una tercera opción es usar un sistema de cifrado híbrido para cifrar tanto la firma como el documento. El firmante usa su clave privada, y cualquiera puede usar su clave pública para comprobar la firma y el documento. Esto suena bien, pero en realidad no tiene sentido. Si este algoritmo hiciera el documento seguro también lo aseguraría de manipulaciones, y no habría necesidad de firmarlo. El problema más serio es que esto no protege de manipulaciones ni a la firma, ni al documento. Con este método, sólo la clave de sesión del sistema de cifrado simétrico es cifrada usando la clave privada del firmante. Cualquiera puede usar la clave pública y recuperar la clave de sesión. Por lo tanto, resulta obvio usarla para cifrar documentos substitutos y firmas para enviarlas a terceros en nombre del remitente.

Un algoritmo efectivo debe hacer uso de un sistema de clave pública para cifrar sólo la firma. En particular, el valor "hash" se cifra mediante el uso de la clave privada del firmante, de modo que cualquiera pueda comprobar la firma usando la clave pública correspondiente. El documento firmado se puede enviar usando cualquier otro algoritmo de cifrado, o incluso ninguno si es un documento público. Si el documento se modifica, la comprobación de la firma fallará, pero esto es precisamente lo que la verificación se supone que debe descubrir.
El Digital Signature Algorithm es un algoritmo de firmado de clave pública que funciona como se ha descrito. DSA es el algoritmo principal de firmado que se usa en GnuPG

Criptografía

La criptografía es un campo de las matemáticas y de las ciencias computacionales que se encarga de asegurar la información.
Es un tema interdisciplinario, que toma conceptos de diversos campos de estudio. Las formas de criptografía más antiguas se encargaban primariamente de patrones en el lenguaje. Más recientemente, el énfasis ha cambiado, y la criptografía hace uso extensivo de las matemáticas, particularmente las matemáticas discretas, incluyendo temas de teoría de números, de información, complexión computacional, estadística y combinaciones. También se considera una rama de la ingeniería, aunque una muy inusual, ya que trata con oposición inteligente, activa y maliciosa. Un área activa de

investigación estudia la relación entre problemas de criptografía y la física cuántica (como la computación cuántica y la criptografía cuántica).

Existen dos tipos generales de criptografía: la criptografía simétrica y la asimétrica, que se analizan más a fondo a continuación:

Criptografía Simétrica

Esta se refiere a los métodos de encriptación en los cuales tanto el emisor como el receptor comparten la misma llave (o en que sus llaves son distintas, pero relacionadas de una manera fácilmente calculable). Otros términos pueden ser *llave secreta, privada, una llave* o *llave solitaria*. Durante mucho tiempo, esta era la única forma de encriptación públicamente conocida hasta 1976.

El estudio de la criptografía simétrica moderno se remonta principalmente al estudio de bloques de cifrado y de corrientes de cifrado. Un bloque de cifrado es la forma moderna de una cifra polialfabética, es decir, que toman un bloque de texto plano y una llave, y su salida es un bloque de datos del mismo tamaño. Estos no son cripto-sistemas seguros (para los niveles modernos, es inaceptable que la encriptación de un texto simple sea siempre la misma), pero puede ser usada en un modo de operación tal como EAX para implementar encriptación auténtica y segura. Otros estándares para la encriptación de bloques son DES y AES.

Las corrientes de cifrado, en contraste, operan sobre una cadena continua de texto plano, y producen una salida encriptada basada en un estado interno que cambia mientras el cifrado está operando. La evolución del estado puede ser controlada tanto por la llave y la corriente de texto plano.

Sin embargo, la criptografía simétrica conlleva algunos problemas aparte de la encriptación misma; principalmente aquellos que pueden ser descubiertos con cifrado por bloques. Por ejemplo:

Las funciones hash criptográficas toman una entrada muy grande (frecuentemente un mensaje) y entregan un hash muy corto de la misma. Aunque deben existir innumerables colisiones hash (dos entradas que obtengan la misma salida), deben ser muy difíciles de encontrar para un algoritmo eficiente. Los dos ejemplos clásicos de funciones hash criptográficas son MD5 y SHA-1.

Los códigos de autenticación del mensaje (o MACs) se parecen mucho a las funciones hash, excepto que una llave secreta se necesita para computar el valor. Como el nombre lo indica, se utiliza principalmente para autenticar mensajes.

Criptografía Asimétrica

Los sistemas de criptografía simétrica utilizan la misma llave para su cifrado y descifrado, o la llave de descifrado puede ser fácilmente calculada de la llave de cifrado. La principal desventaja de este sistema es que las dos partes comunicándose deben compartir una llave secreta; podría ser difícil establecer el secreto inicialmente, ya que la llave es muy vulnerable mientras es transportada.

En 1976, un impactante artículo escrito por Whitfield Diffie y Martin Hellman propuso la noción de criptografía de llave pública, en la cual dos llaves diferentes pero relacionadas son utilizadas; una para encriptación, y otra para desencriptación. Sin embargo, aunque estén relacionadas, un sistema de llave pública está construido de forma tal que la posesión de una llave no permite el cálculo práctico de la otra llave; en vez de eso, las llaves se producen como un par.

En un sistema así, la llave de cifrado puede ser distribuida libremente, siempre y cuando la llave de descifrado permanezca secreta, por lo tanto, la llave de cifrado es una llave pública, y su pareja es una llave privada o secreta. Diffie y Hellman demostraron que la criptografía era posible presentando el protocolo de intercambio de llaves Diffie-Hellman.
En 1978, Ronald Rivest, Adi Shamir y Len Adleman inventaron el RSA, el primer cifrado de llaves públicas publicado. Sin embargo, en 1997, se supo que la criptografía asimétrica había sido inventada secretamente en el GCHQ, una organización de inteligencia británica, en los primeros años de la década de los 70s, y que tanto Diffie-Hellman y RSA habían sido descubiertas en secreto (por Malcolm J. Williamson y Clifford Cocks respectivamente).

RSA, además de ser el primer ejemplo conocido de un sistema de llave pública, también es uno de los más populares. Otros sistemas asimétricos incluyen el Cramer-Shoup y varias técnicas de curvas elípticas.

Además del cifrado, la criptografía asimétrica cubre las firmas digitales. Una firma digital debe ser el equivalente electrónico de una firma, es decir, que sea fácil de producir para el usuario correcto, pero muy difícil para cualquier otro de falsificar. Sin embargo, las firmas digitales sobrepasan esta noción incorporando el mensaje a firmar en la computación de la misma firma; de esta forma, las firmas digitales no pueden ser simplemente movidas de un documento a otro. En un esquema de firma digital, hay dos algoritmos: uno para firmar, en el cual la llave secreta se combina con el mensaje, y uno para verificar, en el cual la llave pública se utiliza para comparar la firma digital con el mensaje. El RSA se puede usar para la creación de firmas digitales, y algunos esquemas tales como DSA y ElGamal están diseñados especialmente para las firmas. Las firmas digitales son centrales para la operación de la infraestructura de llaves públicas, y varios esquemas de seguridad en red (como Kerberos, muchos VPNs, etc).

Los algoritmos de llaves públicas son utilizados frecuentemente en la complejidad computacional o problemas de teoría de números. Por esto, muchos algoritmos de llave pública incluyen operaciones como multiplicación modular, y exponenciación, los cuales son mucho más caros que las técnicas usadas para la creación de cifras de bloques. Como tal, los sistemas de llaves públicas son usualmente usados en sistemas híbridos, en los que la encriptación simétrica (que es mucho más rápida) se utiliza para el cuerpo del mensaje, mientras que la llave simétrica se manda con el mensaje, encriptada utilizando la llave pública. Similarmente, los esquemas híbridos de firmas que se utilizan frecuentemente computan una función hash, y solamente el hash resultante es cifrado digitalmente.

La seguridad en cómputo es absolutamente indispensable desde el comienzo de las computadoras mismas. Es vital que los diseñadores de sistemas, desde el más básico al

más complejo sepan la importancia de un sistema seguro, y la implementación de servicios y políticas para garantizar que las soluciones informáticas que se desarrollen sean seguras.

CAPÍTULO III

PROPUESTA DE PROYECTO DE UN SISTEMA DE CARRITO DE COMPRAS SEGURO BASADO EN WEB

Introducción

Durante los últimos años, la rentabilidad de un negocio basado exclusivamente en forma electrónica a través de Internet ha ido en auge. No es una proposición arriesgada la de establecer un comercio en línea donde se puedan vender diversos artículos, utilizando tecnologías diversas para ofrecer a la empresa y al cliente seguridad, mercancía y un servicio de calidad, y bajos costos.

Se pretende desarrollar una solución al problema de un negocio basado en Internet. La empresa se encarga de vender artículos de cómputo por medio de transacciones electrónicas; ya tiene resueltos los problemas fuera del sistema, como la provisión de la mercancía y las empresas de envío.

Objetivo

Se pretende desarrollar un sistema capaz de recibir, validar y procesar ordenes de compra en una infraestructura cliente-servidor
Se requiere un sistema capaz de trabajar en un servidor, que tenga la capacidad de mostrar el inventario de una forma atractiva al posible cliente, que provea de información sobre el mismo, y que maneje de manera transparente y segura los procesos de transacción.

Alcance

La aplicación ofrece funcionalidad para:

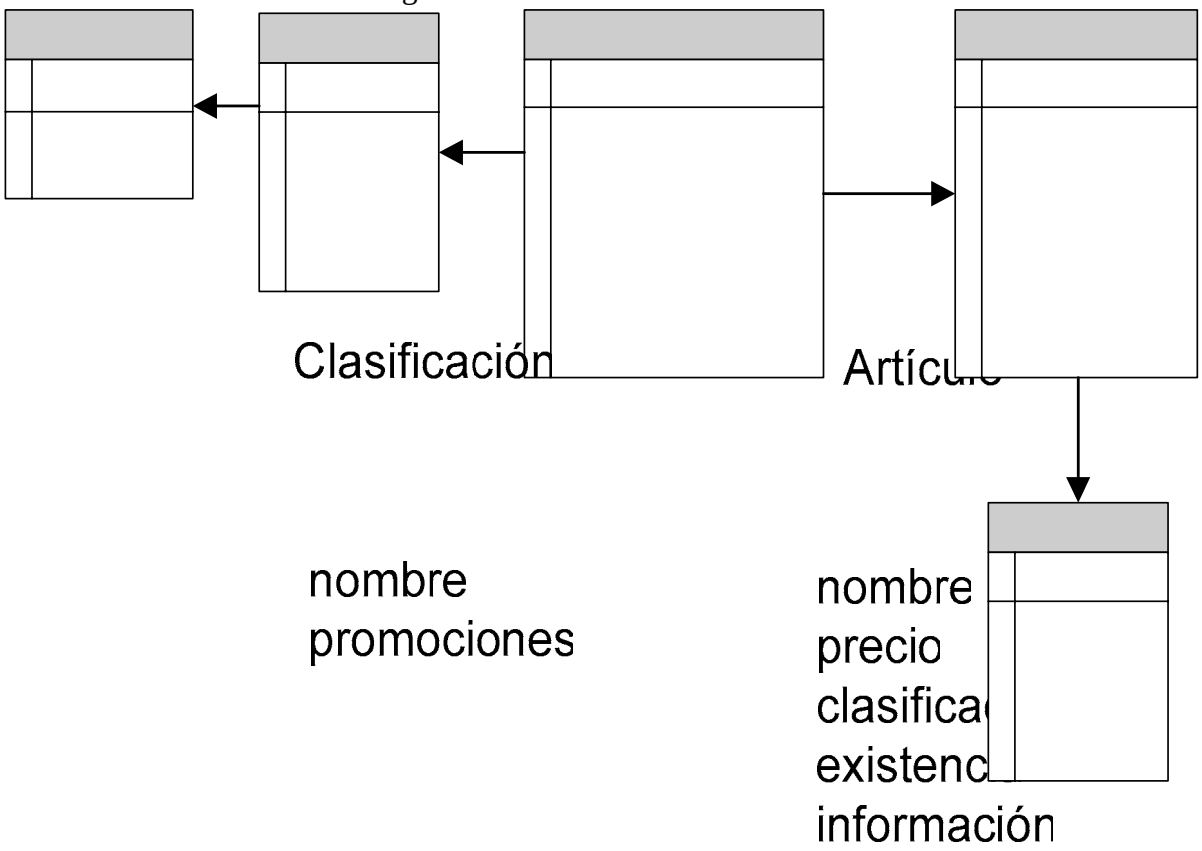
- Presentación de la mercancía en un entorno amigable y atractivo
- Recepción de la orden de compra, y de los datos del cliente
- Procesamiento de la orden.

Lo primero que se tiene que considerar es la plataforma donde se desee basar el sistema, se enfrenta a las opciones de Windows Server o Linux Slackware:

Windows Server.- es un sistema relativamente estable, pero difícil de asegurar, es muy caro. Por otra parte es bastante fácil de usar.

Linux Slackware.- como se analizó, es un sistema operativo libre, lo que lo hace muy barato inicialmente; sin embargo, existe el problema del mantenimiento, el cual es muy caro, también es necesario analizar el entrenamiento que requieran las personas que lo vayan a utilizar. Sin embargo, Linux es un sistema muy seguro, confiable y estable, lo que lo convierte en la mejor de las dos opciones.

Se utilizará MySQL para la creación de la base de datos del inventario y de la información relevante de cada transacción. Para tal fin, se desarrollará un diagrama entidad relación basado en el siguiente modelo:



Para asegurar el buen funcionamiento del sistema, se erigirá una infraestructura PHP sobre un servidor Apache Web Server.

Es necesario activar tanto soporte para PHP como para SSL en el momento de instalación de Apache. Se usará el puerto 8180 como puerto de entrada para evitar la posibilidad de algunos ataques simples que quieran entrar por el puerto 80, que es el tradicional. También sería recomendable que Apache diera privilegios a un usuario dedicado al servidor, y no a "Nobody", ya que muchas veces se sobrecarga de responsabilidades este último, y puede ser vulnerable.

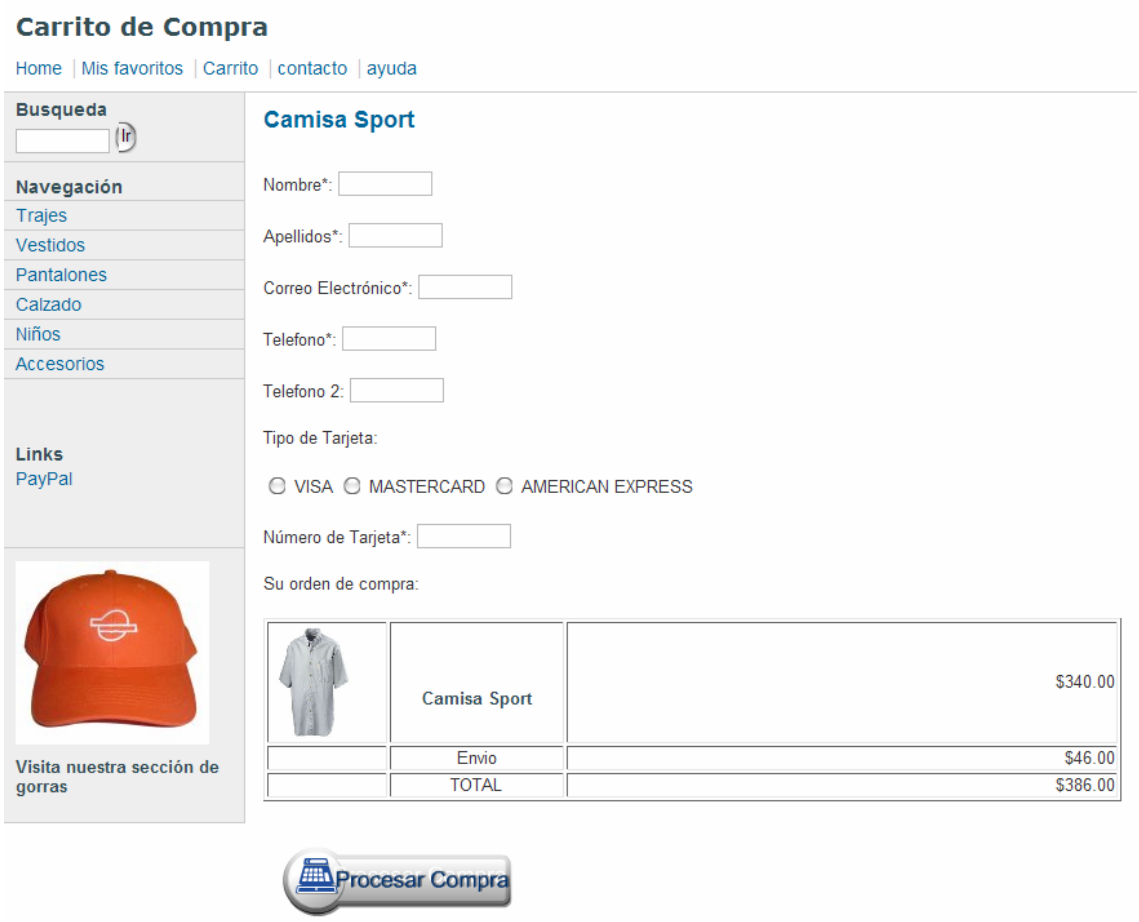
Una vez configurado Apache, creado el dominio del servidor, y el dominio de mail, es posible adentrarse a la creación del sistema en sí, basándose en HTML en conjunto con PHP y MySQL para darle tanto funcionalidad como una imagen atractiva al usuario.

Modulos

Presentación de la mercancía en un ambiente amigable y atractivo.

Como se puede ver en este boceto preliminar, el sitio en su forma básica de compra le ofrece al usuario la posibilidad de examinar una descripción del artículo, junto con una imagen. Sin nunca perder de vista las diferentes secciones de la tienda. El cliente puede navegar por la misma, e ir añadiendo artículos al carrito de compras sin tener que preocuparse por perderlos después. Esto gracias al uso de cookies y de registros en PHP para mantener guardada la información personal de cada cliente. Inclusive se ofrece la opción de ofrecerle productos a los que el usuario haya accesado con frecuencia.

Recepción de la orden de compra, y de los datos del cliente



Una vez que el cliente ingresa los artículos deseados al carrito de compras y está listo para finalizar su compra, el sistema le ofrece esta pantalla; donde se puede ver una tabla con los artículos en el carrito, con su precio, y el costo del envío.

En la parte superior de la composición, se encuentran los campos a llenar con los datos personales necesarios del cliente para llevar a cabo su orden. Esta parte en particular del sistema será desarrollada con la tecnología HTTPs para garantizar la completa seguridad de estos datos y la información relacionada con cada compra. Adicionalmente, será encriptada con una llave generada por GnuPG.

Procesamiento de la orden.

Después de procesada la orden, el sistema le ofrece al usuario una llave generada al instante para identificar su orden, y poder hacer cambios o devoluciones con ella. Detrás de esta interfaz, el sistema enviará la información encriptada de la compra a un servidor de transacciones bancarias para hacer el intercambio de dinero. El cliente recibe también un correo electrónico con el resumen de su compra, gracias a un programa PHP que genera el mismo. Y puede imprimir una copia de su orden para su uso personal.

Conclusiones

Idealmente, se desea tener un sitio Web que le permita al usuario ingresar a una página principal con pequeñas presentaciones de artículos al azar, a menos que el usuario ya haya ingresado al sitio con anterioridad, en tal caso se le presentan con los artículos de la misma clasificación que sus favoritos.
Debe de ser posible buscar por nombre un artículo, comparar precios, y navegar por las clasificaciones de los mismos, poder agregar mercancía al carrito, y seguir comprando. El sistema debe de ser capaz de poder guardar las preferencias de compra y la información del cliente relevante.

Para garantizar que el sistema es seguro, se utilizará el programa de software libre mod_security, que optimiza sistemas basados en Apache; además de que filtra y analiza los requests que llegan al servidor, lo que más interesa es su capacidad de protección contra ataques SQL injection, que puedan afectar la base de datos.

Además, se cifrará por medio de GnuPG los números de las tarjetas de crédito y los datos relevantes de los usuarios. Todos los datos de transacción se tienen que hacer bajo un protocolo seguro de http (https), y finalmente, es importante que se consiga un certificado para el sitio.

CONCLUSIONES

Es muy importante el cambio de paradigma que ha ocasionado la incursión del software libre al mercado. La información y las herramientas necesarias para realizar cualquier tipo de proyecto que se requiera esta disponible para la mayoría de todos. Sin embargo, es necesario ser discriminantes cuando se diseña una nueva solución. El software libre tiene muchos beneficios, tantos, que a veces es muy fácil perder de vista las desventajas; la mayor podría ser los costos de reparación o mantenimiento.

Las herramientas libres son capaces de construir un sistema completo, seguro y confiable. Apache es en la actualidad el programa servidor más utilizado en todo el mundo, superando con creces a la solución de Microsoft: Internet Information Server. PHP es el lenguaje para desarrollo de aplicaciones en red por excelencia.

Un sistema necesita ser capaz de garantizarle al usuario una completa seguridad al utilizarlo, sobre todo cuando se involucra información personal y/o valiosa. La seguridad es un campo delicado en la informática; depende mucho de los errores humanos, de la implementación del sistema. Es imposible garantizar al 100% la seguridad de un sistema, por eso es necesario estar analizando posibles formas de ataque constantemente, y también informarse desde un principio sobre las más comunes, para evitar caer en agujeros de seguridad.

BIBLIOGRAFÍA

Actualización y Mantenimiento del PC
Madrid: Anaya, 2001

Handbook of Theoretical Computer Science
J. van Leeuwen, Ronald Rivest
Elsevier Science Publishers, 1990

Foundations of Cryptography Vol. 1: Basic Tools
Oded Goldreich
Cambridge University Press, 2001

Security Engineering: A Guide to Building Dependable Distribution Systems
Ross J. Anderson
Cambridge University Press, 2000

Mantenimiento Correctivo y Preventivo para PCs
Alonso Molina Gutierrez
DGSCA, UNAM, 2000

Edición Especial Linux, Máxima Seguridad
Prentice Hall, 1999

Linux, Network Administrador Guide
Olaf Kirch
Olaf Kirch, 1995

Linux, Guía de Administración e Instalación
Vicente López Camacho
McGraw Hill, 2000

Linux System Administrator Survival Guide
Thimoty Parker
Sams, 2000

Apache Cookbook
Ken Coar
O Reilly, 2003

Linux, Recursos para el Usuario
James Mohr
Prentice Hall, 1999

Internet y Seguridad en Redes
Karanjit Siyan
Prentice Hall, 1995

Computer Security Basics
Deborah Rusell
O Reilly & Associates, 2002

Seguridad y Comercio en el Web
Simson Garfinkel
McGraw-Hill Interamericana, 1999

UNIX Security
Kansas Lawrence
R & D Books, 1997

Web Hacking: Attacks and Defense
Stuart McLure
Addison-Wesley, 2002