UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

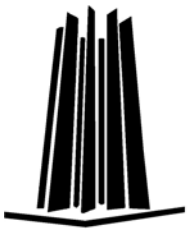
FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN

"INFORME DEL PROYECTO PARA LA ELABORACION DE UNA INTERFAZ WEB DE REGISTRO DE PERSONAL UTILIZANDO SOFTWARE LIBRE"

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

A N D R A D E G Ó M E Z L U I S J A V I E R

ASESOR: ING. SILVIA VEGA MUYTOY



MÉXICO, 2006.

Indice *Página*

INTRODUCCIÓN

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-WWW-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.

Objetivos Específicos:

- El participante identificará los métodos de operación y administración del sistema operativo, utilizará los programas de manejo de usuarios, y llevará a cabo la instalación del dispositivo de respaldo, configuración de la red e instalación del software.
- El participante instalará el sistema operativo Linux en PC's y configurará la tarjeta de red, video y particiones del disco duro; asimismo manejará el ambiente de usuario
- El participante elaborará páginas WWW apoyado con HTML.

• El participante identificará el procedimiento para instalar, configurar y administrar su propio servidor de WWW en un servidor de plataforma Linux.
• Proporcionar al participante los conocimientos necesarios que le permitan crear aplicaciones dinámicas e interactivas para el Web utilizando el lenguaje PHP.
• El alumno conocerá y desarrollará una aplicación de bases de datos que funcione a través del WWW, empleando herramientas de software libre.
• El participante reconocerá la importancia de la seguridad e identificará los elementos que le permitirán proteger el sistema y la información.

Organización:

MÓDULO

1. Sistema operativo Linux
2. Instalación y administración de Linux
3. Editores para la creación de páginas WEB
4. Administración de servidores de WWW con Linux
5. Programación con PHP
6. Interacción de WWW con bases de datos
7. Introducción a la seguridad en cómputo
8. Programación Java *

** Módulo opcional de titulación*

I. HERRAMIENTAS DE SOFTWARE LIBRE PARA EL DESARROLLO DE SISTEMAS

I.1 **Sistema Operativo Linux**

OBJETIVO ESPECÍFICO: El participante conocerá el sistema Linux y utilizará comandos básicos para su operación. TEMARIO: 1. Introducción 2. Requerimientos de instalación 3. Comandos y utilerías básicas 4. Nociones de administración 5. Fuentes de información

La historia de Linux.

Linux nació como un proyecto de Linus Torvalds, inspirado en el MINIX, el sistema operativo desarrollado por Andrew S. Tanenbaum en su obra "*Sistemas Operativos: Diseño e Implementación*". Libro en el cual, tras un estudio general sobre los servicios que debe proporcionar un sistema operativo y algunas formas de proporcionar éstos, introduce su propia implementación del UNIX en forma de código fuente en lenguaje C y ensamblador, además de las instrucciones necesarias para poder instalar y mejorar el mismo.

Fue en Julio de 1991 cuando el estudiante de ciencias computacionales de la Universidad de Helsinki en Finlandia, envió su primer mensaje al grupo de noticias *comp.os.minix*, respecto a un proyecto personal sobre el sistema operativo *Minix*. Este es el primer mensaje:

From:torvalds@klaava.Helsinki.FI (Linus Benedict Torvalds) *Newsgroup: comp.os.minix* *Subject: GCC-1.40 and a posix question* *Message-ID: 1991Jul13, 100050.9886@klaava.Helsinki.FI* *Date: 3 Jul 91 10:00:50 GMT*
Hello netlanders, *Due a project I'm working on (in minix), I'm interested in the posix standard definition. Could somebody please point me to a (preferably) machine-readable format of the latest posix rules? Ftp-sites would be nice.* *....*
Linux Torvalds torvalds@kruuna.helsinki.fi

Al que le siguió este mensaje, que muchos consideran el verdadero inicio del Linux.

From:torvalds@klaava.Helsinki.FI (Linus Benedict Torvalds) *Newsgroup: comp.os.minix* *Subject: What would you like to see most in minix?* *Summary: small poll for my new operating system* *Message-ID: 1991 Aug 25, 20578.9541@klaava.Helsinki.FI* *Date: 25 Aug 91 20:57:08 GMT* *Organization: University of Helsinki.*
Hello everybody out there using minix- *I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones. This has been brewing since april, and is starting to get ready. I'd like any feedback on things people like/dislike in minix; as my OS resembles it somewhat (same physical layout of the file-sytem due to practical reasons) among other things.* *I've currently ported bash (1.08) an gcc (1.40), and things seem to work.* *This implies that I'll get something practical within a few months, and I'd like to know what features most people want. Any suggestions are welcome, but I won't promise I'll implement them ☺*
Linux Torvalds torvalds@kruuna.helsinki.fi

Respecto al inicio, Linus escribiría años después lo siguiente:

> *"Estaba finalizando de escribir mi tesis - Fue el único crédito que he tenido en mi vida académica, cuando escribí Linux. Yo tenía 21 años, y la verdad no tenía idea de lo que estaba haciendo. Me creía el mejor programador del mundo, todo programador de 21 años lo cree,.. Me dije: ¿Cuán difícil puede ser, es sólo un sistema operativo...? "*

Las versiones iniciales fueron distribuidas en código fuente por el propio Linus, para que otras personas puedan entender su proyecto y sobretodo para que lo ayuden.

La primera versión de Linux, enumerada como 0.01 contenía sólo los rudimentos del núcleo y funcionaba sobre una máquina con el MINIX instalado, esto es, para compilar y jugar con Linux era necesario tener instalado el MINIX de Tanembaum. Nunca se hizo un anuncio de la versión 0.01. Las fuentes ni siquiera eran ejecutables.

El 5 de Octubre de 1991, Linus anunció su primera versión 'oficial', la 0.02 con esta versión ya se podía ejecutar el *bash* (*GNU Bourne Shell*) y el *gcc* (*GNU C compiler*).

Después de la versión 0.03, Linus cambió este número por 0.10 y tras las aportaciones de un grupo inicial de usuarios se incrementó de nuevo la denominación a 0.95, reflejando la clara voluntad de poder anunciar en breve una versión 'oficial' (con la denominación 1.0).

Linus trabajó activamente hasta la versión 0.96, pues tras ello, se sumaron al proyecto más programadores y se formó un grupo de desarrollo amplio (*Linux Developers*) que continúa siendo dirigido por él; pero como él mismo lo reconoce, su labor es más la de un *router* del grupo que la del desarrollo en si.

En Diciembre de 1993 el núcleo estaba en su versión 0.99. En la actualidad la última versión estable es la 2.6.16.5 aunque existe ya la versión de desarrollo 2.6.17-rc1.

La enumeración de las versiones de Linux implica a tres números separados por puntos, el primero de ellos es la versión del sistema operativo, es el que distingue unas versiones de otras cuando las diferencias son importantes. El segundo número indica el nivel en que se encuentra dicha versión. Si es un número impar quiere decir que es una versión de desarrollo con lo cual se avisa de que ciertos componentes del núcleo están en fase de prueba, si es par se considera una versión estable. El último número identifica el número de revisión para dicha versión del sistema operativo, suele ser debido a la corrección de pequeños problemas o al añadir algunos detalles que anteriormente no se contemplaba con lo cual no implica un cambio muy grande en el núcleo. Como ejemplo se tiene la versión 2.4.13, esto es, la versión 2 en su nivel 4 (estable) y la revisión número 13 de la misma en este caso fue la última.

Hay que señalar que Linux no sería lo que es sin la aportación de la *Free Software Foundation* y todo el software desarrollado bajo el soporte de esta asociación así como la distribución del UNIX de Berkley (BSD), tanto en programas transportados como en programas diseñados para este que forman parte de algunas distribuciones del Linux.

Casi todos los paquetes de programas UNIX importantes, libremente redistribuibles han sido portados a Linux, y también hay abundante software comercial. La lista de hardware soportado es mayor que la del núcleo original. Mucha gente ha ejecutado pruebas de rendimiento en sistemas Linux 80486 y han encontrado que es comparable a estaciones de trabajo de Sun Microsystems y Digital Equipment Corporation.

Conceptos básicos.

Para entender claramente las capacidades del sistema operativo Linux se empieza por aclarar conceptos básicos.

Tarea:	Actividad que se realiza en cierto tiempo o al instante.
Usuario:	Persona que utiliza un servicio, sistema o instrumento.
Plataforma:	En informática se refiere a la arquitectura del sistema y generalmente de forma indirecta al sistema operativo.

Linux es un sistema operativo multitarea, multiusuario y multiplataforma de código abierto. Es multitarea porque soporta la ejecución de varias tareas en tiempo real. Es multiusuario porque soporta la apertura de varias sesiones de trabajo para varios usuarios al mismo tiempo. Es multiplataforma porque puede implementarse en diferentes arquitecturas de hardware. Los limites de cantidad de tareas y usuarios los determina la capacidad del hardware instalado en el equipo.

Linux se compone de dos componentes esenciales: el *kernel* y el *shell*.

Kernel:	Es el software responsable de facilitar a los distintos programas acceso seguro al hardware del equipo o en forma más básica, es el encargado de gestionar recursos, a través de servicios de llamada al sistema.
Shell:	Es el intérprete de comandos usado para interactuar con el kernel.

El usuario *root*.

Para utilizar Linux es necesario tener una cuenta de usuario en el sistema. Este usuario tiene ciertos privilegios, permisos determinados, acceso particulares y algunas limitantes que varían dependiendo del tipo o nivel de usuario. En el nivel más alto en esta jerarquía de usuarios se encuentra el *superusuario root* que tiene acceso a todo el sistema, tiene todos los permisos sobre todos los archivos y directorios así como la facultad de modificar o eliminar las cuentas de los demás usuarios. La cuenta de *root* se le otorga al administrador del sistema.

Primeros comandos:

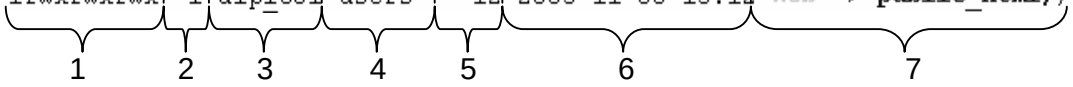
man (opciones) [comando]

Solicita el manual de la instrucción deseada. Ejemplo: *man ls*

ls (opciones) (ruta)

Enlista el contenido de una carpeta. Se le puede pasar la ruta o utilizarlo directamente sin ella.

```
diplo01@atena:~$ ls -l
total 26
-rwxr-xr-x  1 diplo01 users 17290 2005-11-08 15:24 aMSN_installer.bin*
-rw-r--r--  2 diplo01 users   182 2005-11-08 15:35 cheats.html
drwxr-xr-x  2 diplo01 users   208 2005-03-16 16:21 other/
-rw-r--r--  1 diplo01 users     0 2005-11-08 15:09 pacman.swf
drwxr-xr-x  3 diplo01 users   352 2005-11-08 15:31 public_html/
lrwxrwxrwx  1 diplo01 users    12 2005-11-08 15:12 web -> public_html//
```



Esta parte se lee por columnas como sigue:

1a. columna — Tipo y permisos que tiene el archivo, este a su vez se subdivide así:

1er. carácter: tipo de archivo

- d directorio
- \- archivo común
- l liga
- c archivo tipo carácter
- b archivo de acceso por bloques
- s *sockets* de conexión
- p tuberías (*piff*)

Los 9 caracteres siguientes son los permisos para el usuario al que pertenece (3 caracteres), el grupo al que pertenece (siguientes 3 caracteres) y para otros usuarios (últimos 3 caracteres).

2a. columna — Niveles de directorio a partir del actual.

3a. columna — Usuario al que pertenece.

4a. columna — Grupo al que pertenece.

5a. columna — Tamaño en Kb.

6a. columna — Fecha y hora de la última modificación.

7a. columna — Nombre completo del archivo (una "/" al final indica que es un directorio).

cp (opciones) [origen] [destino]

Copia archivos del (los) origen (es) hacia su destino. Es muy útil para hacer un respaldo antes de modificar algún archivo o directorio. Para copiar varios archivos se deja un espacio entre sus nombres.

mkdir (opciones) [nombre_del_directorio]

Crea directorios. Ejemplo: *mkdir temporal*

rmdir (opciones) [nombre_del_directorio]

Elimina el directorio especificado. Ejemplo: *rmdir temporal*

mv (opciones) [origen] [destino]

Mover o renombrar archivos.

hostname (opciones)
Muestra la dirección del equipo. Ejemplo *hostname -i*

touch [nombre_del_archivo]
Crea un archivo vacio. Ejemplo: *touch manual.txt*

adduser [nombre_de_usuario]
Crea un nuevo usuario. Ejemplo: *adduser juan85*

Redireccionamiento.

Cuando se trabaja en modo consola los resultados de una consulta o la respuesta (salidas) de un comando se muestran en la pantalla. Si se desea que dichas salidas se envíen a un archivo se deben **redireccionar**. En Linux existen varios tipos de redireccionamiento.

- Directo. Sustituye o crea el destino. Su sintaxis es:
  cat /etc/lilo.conf **>** *lilo.txt*

- Indirecto. Anexa a partir del final de archivo. Si no existe el destino no lo crea. Su sintaxis es:
  cat /etc/lilo.conf **>>** *lilo.txt*

Redireccionamiento de salidas.

- A una salida de verdad. Dada una instrucción, su resultado es enviado de forma condicionada a un archivo. Cuando el proceso termina sin errores se envía a la salida **1**; en caso contrario el resultado se envía a la salida **2**. En ambos casos no se muestra ningún resultado en pantalla. Su sintaxis es:
  rmdir /home/users/juan85/scripts ***1>****fine.log* ***2>****error.log*

- Pasar la salida a otra instrucción. Cuando se desea procesar la salida de una instrucción mediante otra, se utiliza el carácter "**|**" (*pipe*). Su sintaxis es:
  cat /etc/lilo.conf | *grep -I partition*
  Este tipo de redireccionamiento es muy utilizado para crear **filtros**.

Conexión remota.

El sistema operativo Linux ofrece la posibilidad de iniciar sesiones en el sistema de forma remota. Esto hace posible su administración a través de Internet o alguna red local. Los comandos que se utilizan son los siguientes:

Para iniciar sesión:

ssh -l [nombre_de_usuario] [servidor]

- Se puede utilizar el nombre o la dirección IP para especificar el servidor.
- Si el usuario necesita una contraseña para autentificarse le será requerida.

Para subir archivos:

scp [archivo] [nombre_de_usuario]@[servidor]:[directorio_destino]

- Se puede utilizar el nombre o la dirección IP para especificar el servidor.
- El usuario debe tener permisos de escritura en el directorio destino.

Para descargar archivos:

scp [nombre_de_usuario]@[servidor]:[archivo_origen] [destino]

- Se puede utilizar el nombre o la dirección IP para especificar el servidor.
- El usuario debe tener permiso de lectura del archivo origen.

Permisos de los archivos y directorios.

Existen tres tipos de permisos que se tienen en Linux:

- Lectura: Cuando se permite leer un documento, visualizar imágenes y contenido multimedia.
- Escritura: Cuando se permite eliminar o modificar el contenido, nombre y características, así como la eliminación del archivo.
- Ejecución: Cuando se permite ejecutar algún *script* o programa.

Asimismo existen tres niveles de usuarios a los que se les puede asignar un permiso:

- Usuario: Se refiere al usuario propietario del archivo o directorio, usualmente quien lo creó.
- Grupo: Se refiere a los otros usuarios del mismo grupo al que pertenece al usuario propietario.
- Otros: Se refiere a todos los demás usuarios que no se consideraron en las categorías anteriores.

El comando que se utiliza para cambiar los permisos a un archivo, programa o directorio es **chmod**. Su sintaxis es:

chmod [permisos] [archivo]

Cuando se desea especificar el nivel de usuario se utilizan las siguientes opciones:

- *-u* para el usuario
- *-g* para el grupo
- *-o* para otros
- *-a* para todos

Existen dos modos de declarar los permisos: simbólico y octal. Ambos se asignan de acuerdo a la siguiente tabla.

Octal	R	W	X	Símbolo	Significado
0	-	-	-	-	Ninguno (prohibido)
1	-	-	1	x	Ejecución
2	-	1	-	w	Escritura
3	-	1	1	wx	Escritura y ejecución
4	1	-	-	r	Lectura
5	1	-	1	rx	Lectura y ejecución
6	1	1	-	rw	Lectura y escritura
7	1	1	1	rwx	Lectura, escritura y ejecución

- Modo simbólico: Se utilizan los símbolos de más (+) para agregar permisos y menos (-) para quitarlos. Ejemplos:
  chmod u-w+x, g-r+wx, o+wx ejemplo.txt
  chmod u=x, g=rw, r ejemplos.txt

- Modo octal: Sólo se escribe el decimal que equivale al permiso que se le da al archivo para cada usuario, es decir que deben escribirse tres dígitos. Ejemplos:
  chmod 755 ejemplos.txt
  chmod 777 ejemplos.txt

Monitorear procesos.

Cuando se necesita saber que procesos se están ejecutando en el sistema, se pueden utilizar los comandos:

1) **ps** muestra los procesos activos en el momento de su ejecución. Su sintaxis es:
   ps [opciones]
2) **top** muestra todos los procesos en tiempo real.
   top
3) **jobs** muestra los trabajos activos. Su sintaxis es:
   jobs

Los procesos que están en ejecución en el sistema pueden presentar varios estados, como pueden ser:

Z	*Zombie* (background)
R	Ejecución
T	Detenido
S	Durmiendo, esperando algo
D	Durmiendo

Ligas

En Linux es posible crear ligas (enlaces) de un archivo. Sirven para enlazar archivos de forma suave o directa.

- Liga directa (dura). Crea una copia del archivo y la enlaza directamente con el origen. Si el archivo original es modificado, la copia se modificará también; si el archivo es eliminado la copia se conserva. Su sintaxis es:
  ln [archivo_origen] [archivo_destino]

- Liga suave. Crea un acceso directo a un archivo, proceso o tarea. Su sintaxis es:
  ln -s [archivo_origen] [archivo_destino]

Apagar el sistema.

Cuando se necesita apagar el sistema o reiniciarlo se pueden utilizar los siguientes comandos:

1. **shutdown** se utiliza para apagar el sistema dependiendo de las opciones en un tiempo variable. Su sintaxis es:
   shutdown [opciones] [tiempo]
2. **halt** se utiliza para detener todos los procesos del sistema y apagarlo. Su sintaxis es:
   halt
3. **restart** se utiliza para reiniciar el sistema. Su sintaxis es:
   restart

Operaciones con usuarios.

Cuando se necesita crear un nuevo usuario en el sistema se utiliza el comando **adduser**. Su sintaxis es:
adduser [nombre]

A continuación se introducen los datos que se van solicitando, exceptuando los que contienen información personal, esto último por seguridad.

Si un usuario desea cambiar el tipo de *shell* de su sesión puede hacerlo con el comando **chsh**. Su sintaxis es:
chsh [nombre_de_usuario] [archivo_de_shell]

Así mismo es posible cambiar los datos personales de un usuario con el comando **chfn**. Su sintaxis es:
chfn [nombre_de_usuario]

Para eliminar o dar de baja a algún usuario se utiliza el comando **delusser**. Su sintaxis es:

userdel [nombre]

Un consejo práctico respecto a los usuarios sería que se revisen ciertos archivos en el sistema los cuales contienen información respecto a los usuarios y de forma encriptada las contraseñas y datos importantes. Los archivos mencionados son:

/etc/passwd
/etc/shadow
/home/[nombre_de_usuario]
/etc/shells/[shells]

I.2 **Instalación y Administración de Linux**

OBJETIVO ESPECÍFICO:
El participante instalará el sistema operativo Linux en *PCs* y configurará la tarjeta de red, video y particiones del disco duro.

TEMARIO:

1. Perfil y actividades del Administrador.
2. Instalación.
3. Dar de alta y/o de baja el sistema.
4. Mantenimiento de claves de usuarios.

Conceptos básicos.

Administrador de sistemas: Responsable de configurar, mantener y actualizar el sistema o conjunto de sistemas que forman una red.
Cuidando el funcionamiento del software, hardware y periféricos de forma que estén disponibles para ser utilizados por los usuarios.

Importancia de la Administración

- Proporcionar un ambiente seguro, eficiente y confiable
- Brindar un funcionamiento confiable del sistema.
- Se divide el trabajo entre varios administradores, dependiendo del tamaño del sistema.

Planear las actividades.

- Absolutamente todas las actividades de administración se planean.
- Guardar copias de seguridad.
- Jamás modificar sin respaldar previamente.

Tener copias de seguridad

Absolutamente todo lo que se almacene o realice en el sistema debe estar respaldado:

- Información de los usuarios
- Bases de datos
- Correo electrónico
- Archivos originales del sistema

Conocer las utilerías del sistema

- Utilerías del sistema:
  Básicas: *cut*, *sort*, *paste*, *diff*, *comm*, *tail*, *head*, *grep*, *egrep*, *compress*, etc.
  Control de tareas: *at*, *crontab*.
  Respaldos: *dump*, *dd*, *restore*, *tar*, *cpio*.

- Herramientas de programación:
  C.
  shell.
  awk.
  Perl.

- Documentación:
  En línea (*man*, *apropos*, *info*).
  Impresa (libros, manuales).
  Internet.

Conocimientos requeridos

- Técnicas de programación.
- Dominio de al menos un lenguaje de programación.
- Funcionamiento del sistema operativo.
- Técnicas de administración del sistema operativo.
- Conocimientos básicos de hardware y mantenimiento de dispositivos.
- Comprensión profunda sobre redirección, tuberías, procesamiento en segundo plano, etc.
- Manejo de *vi*, pues es el común denominador entre los sistemas UNIX.
- Programación *shell*.

Conocer el hardware

- ✓ Características, modelo, capacidad, etc.
- ✓ Ubicación física.
- ✓ Consultar las listas de hardware soportado publicadas por la distribución que se desea instalar (HCL)
- ✓ Hacer pruebas antes de adquirir el equipo

Establecer políticas de uso y administración

- Apertura de cuentas.
- Horas de mantenimiento.
- Responsabilidad de los respaldos.
- Borrado de archivos temporales.

- Cuotas de disco.
- Seguridad del sistema.

Tareas administrativas comunes:

- Administración usuarios.
- Configuración de dispositivos.
- Programar respaldos periódicamente.
- Capacitar usuarios.
- Asegurar el sistema.
- Registrar los cambios del sistema.
- Asesorar a los usuarios.
- Mantenimiento de claves de usuarios.

Instalación y mantenimiento de dispositivos.

- Impresoras.
- Discos.
- Unidades de respaldo.
- Instalación y actualización de software (comercial y dominio público).
- Configuración de las interfaces de red.
- Administración de los recursos (cpu, memoria y disco).
- Monitoreo del sistema.
- Detección de fallas.
- Auditoria e implantación.

Instalación.

Requerimientos de instalación

Slackware Linux no requiere de un sistema extremadamente potente para ejecutarse. Se puede ejecutar en equipos 386 o superiores. Los requerimientos mínimos para instalar y ejecutar Slackware son:

Hardware	Requiere
Procesador	386
RAM	16 MB –64M
Espacio en Disco	500MB –3G
Floppy Drive	1.44 MB
Unidad CD-ROM	

Preparación de la instalación

Antes de instalar Slackware se deben conocer las series de paquetes que se van a instalar. Linux Slackware es una de las distribuciones de

Linux más antiguas, contiene un conjunto de paquetes que se reparten en series ordenadas alfabéticamente.

Tipos de instalación:

CD-ROM

Si nuestro equipo cuenta con la opción de arranque por medio de un CD-ROM, esta seria la forma más fácil de instalar Linux Slackware.

Discos de arranque

En caso de no contar con la opción de arranque desde un CD-ROM se necesita crear los discos de arranque *booty root*.

Disco de red

Esta opción se emplea cuando se necesita hacer la instalación de varios equipos, se emplea la red para este propósito y se puede hacer mediante NFS, FTP y HTTP.

Disco PCMCIA

Este método se emplea cuando se desea instalar Linux en una computadora portátil, este tipo de equipos cuentan con dispositivos PCMCIA.

Comienzo de la instalación.

Cuando la instalación se realiza por CD-ROM automáticamente se cargan la imagen de los discos *boot* y *root*. Después de arrancar la instalación Linux detectara la mayor parte del hardware que esta instalado en nuestro equipo. El primer paso de la instalación es elegir la configuración de nuestro teclado.

Se debe elegir la opción *es.map* del menú, que es el mapa del teclado para la configuración en español. El termino mapa del teclado se refiere a la ubicación de las teclas en un teclado y dependiendo del idioma estas tendrán una ubicación diferente.

A continuación se debe registrar ante el sistema para que entregue un *shell* y seguir con la instalación, para ello hay que registrarse como *root*

Ahora se tiene que crear las particiones necesarias para la instalación.
El administrador debe tener un esquema de disco pensado para una administración óptima y que sea flexible para facilitar cualquier tarea que se desee realizar, pensando en la funcionalidad del servidor. Tener todo el sistema en una sola partición es mala idea ya que no es flexible y hace difícil de administrar el servidor. Lo más recomendable es hacer particiones pensando en la carga de archivos del sistema, de los

usuarios, servicios y otras consideraciones, logrando con ello facilitar las tareas de respaldo, actualización, administración fácil y eficiente, entre muchas otras ventajas.

Ahora se tienen que guardar los cambios realizados en las particiones y comenzar la instalación con el comando *setup.*

Aquí se debe seguir el procedimiento de arriba hacia abajo, para lo cual hay que desplazarse a KEYMAP para elegir el tipo de teclado como anteriormente se realizó. Una vez que se ha definido el tipo de teclado el sistema automáticamente detecta la partición de *swap*, solicita darle formato y la prepara para la instalación. Una vez que ha preparado el área de *swap* el proceso de instalación reconoce las particiones de Linux, la partición que se elija en este momento es donde se va a instalar el sistema. Ya que se eligió la partición solicita darle formato rápido, formato con revisión de sectores dañados o montar la partición sin formatear.

El siguiente paso es elegir el recurso desde el cual se va a realizar la instalación. Lo más conveniente es decirle que aplique un montaje del CD en forma automática, ya que de otra forma se necesitaría saber en que puerto se encuentra conectada la unidad de CD y si esta como maestro o como esclavo, información que sólo se puede obtener abriendo el CPU o revisando la configuración en el BIOS. Ya que se ha detectado el CD el asistente pedirá elegir las series de paquetes que se van a instalar.

Ahora se tiene que elegir el modo de instalación, el modo Full es automático, los otros modos solicitan la instalación de los paquetes ya sea por series o aplicando algún otro criterio. Después de haber terminado la instalación de paquetes se continua con la configuración del sistema, el siguiente paso es indicarle desde que medio se desea instalar el *Kernel* para el sistema, obviamente en este caso es desde CD. Ahora se debe elegir cual *kernel* es el que se va a instalar, éste depende de la arquitectura y de las características de la computadora donde se está instalando.

Gestor de arranque *LiLo.*

LiLo es el gestor de arranque el cual ayuda a generar un menú de inicio para equipos multisistemas, entre otras cosas. Debido a que la actual instalación esta contemplada para una máquina que tiene dos sistemas operativos de este menú se elige la opción en modo experto. Este menú permite iniciar la configuración de LILO con las propiedades principales.

Enseguida se debe elegir del menú el tipo de resolución que se desea para modo texto o consola, para ello se necesita saber las características de la tarjeta de video y que tipos de resolución soporta.

El siguiente paso es indicarle en donde se desea instalar LILO, lo recomendable es que se instale en el MBR del disco duro. Para lo cual el asistente preguntara en que disco duro se desea que se instale, esto por si se cuenta con más de un disco duro. Después se tiene que elegir el tiempo de espera antes que inicie con el primer sistema que tiene por *default*.

Una vez que se ha terminado de configurar la parte inicial de LILO hay que agregar los sistemas que se desea sean visibles en el menú de *booteo* de los sistemas operativos.

Dar de alta y/o de baja el sistema.

El administrador de sistemas debe conocer las diferentes formas para dar de alta o de baja el sistema, de ello dependerá en gran medida el rendimiento y confiabilidad del sistema incluso después de un reinicio, también pude ser muy útil para rescatar un sistema cuando no inicia apropiadamente.

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

El nivel de inicio por *default* se encuentra definido en el archivo */etc/inittab*, para ello se necesita conocer los diferentes niveles que están predefinidos en el archivo de inicio. Cabe mencionar que dependiendo del nivel de inicio la rutina de encendido apagado puede variar.

 # These are the default run levels in Slackware:
 # 0 = halt
 # 1 = single user mode
 # 2 = unused (but configured the same as run level 3)
 # 3 = multi user mode (default Slackware run level)
 # 4 = X11 with KDM/GDM/XDM (session managers)
 # 5 = unused (but configured the same as runlevel3)
 # 6 = reboot

Configuración del gestor de arranque LiLo.

Lilo por sus siglas en ingles quiere decir *Linux Loader*, el cual permite generar el menú de inicio de sistemas entre otras propiedades. Toda su configuración se encuentra en el archivo */etc/lilo.conf.* Para configurarlo se invoca al comando *liloconfig*, el cual desplegará el asistente de Lilo que se vio durante la instalación, ahora se tiene que realizar todo el procedimiento para configurar su gestor de arranque e instalarlo. Una vez que se terminó de configurarlo, con el comando *lilo -t* aplicara un *test* a su configuración. De no haber errores ejecute el comando *lilo* para instalar su nuevo gestor de arranque.

Mantenimiento de claves de usuarios

Una de las principales tareas es la administración de las cuentas de usuario, para ello se debe tener un plan administrativo de claves contemplando altas, bajas y cambios de las claves, también es importante definir los grupos a los que va a pertenecer uno o más usuarios así como se debe tener un especial cuidado en la capacitación de los usuarios para elegir contraseñas fuertes, asignar permisos adecuados y administrar apropiadamente los recursos para no tener un abuso de ellos.

Comandos para administrar los usuarios:

Adduser	Agrega usuarios modo comando.
Useradd	Agrega usuarios modo comando.
Userdel	Elimina usuario.
Usermod	Modifica las propiedades del usuario.
Groupadd	Agrega grupos.
Groupdel	Elimina grupos.
Groupmod	Modifica grupos.
Groups	Lista los grupos a los que pertenece un usuario.
Passwd	Modifica propiedades de usuarios/grupos.
Kuser	Administra usuarios y grupos en modo grafico.

Uso de recursos:

El comando *du* permite monitorear el estado del sistema de discos. El comando *df* muestra el espacio ocupado por directorios entregando un total de espacio ocupado, este comando se puede aplicar sobre los directorios de cada usuario y saber rápidamente cuanto espacio esta usando cada usuario en total.

Una buena planeación para la implementación de un servidor permite prevenir ciertos inconvenientes al separar los espacios de almacenamiento para ciertas prioridades, de tal forma que si se satura

un área de almacenamiento no se pierda el control de acceso al sistema para que los usuarios puedan depurar sus cuentas; una buena opción es configurar cuotas de disco.

Sistema de archivos

En Linux hay varios tipos de sistema de archivos, cada uno con características variadas las cuales determinan la funcionalidad del sistema, algunas ofrecen ventajas sobre otras teniendo desde el estándar *ext2* hasta sistemas mejorados como *reiser* o *journal.* El tipo de sistema de archivos se puede elegir en el momento de formatear las particiones determinando el tipo de sistema de archivos que se desea tenga dicha partición.

El sistema nombra a todos los dispositivos de tal forma que siempre se tiene control de ellos, se inicia identificando las unidades IDE o SCSI conforme se encuentran conectados dentro del CPU físicamente. Por la forma en que se nombran los discos en Linux es prácticamente imposible cometer errores, ya que así se puede saber directamente con que disco se esta trabajando físicamente.

IDE 0 disco maestro	/dev/had
IDE 0 disco esclavo	/dev/hdb
IDE 1 disco maestro	/dev/hdc
IDE 1 disco esclavo	/dev/hdd

El sistema dispone de un directorio */mnt* el cual forma parte de la estructura estándar de Linux. Dentro de este directorio existen algunos otros que se sugieren para montar las unidades de disco. Cuando se instala Linux éste reconoce todas las unidades de disco y crea un directorio para cada una de ellas en /mnt, si se agregan mas unidades de disco al CPU se podrían crear tantos directorios como se necesiten.

Montaje de dispositivos.

Para montar un disco se cuenta con el comando *mount*. Su sintaxis es:
mount [dispositivo] [punto_de_montaje]

Anteriormente había que indicar el tipo de sistema de archivo que tiene el dispositivo, actualmente Linux hace el reconocimiento en forma automática y lo monta en el directorio indicado; si el dispositivo y el punto de montaje existen, el disco será montado en el punto de montaje indicado, de otro modo el sistema enviará un mensaje de error.

El sistema cuenta con apuntadores que ayudan a ciertas tareas, tal es el caso de la unidad de CD-ROM, en */dev/* existe un apuntador que se dirige al directorio */mnt/cdrom*, el cual esta declarado dentro de un

archivo de configuración. Cabe aclarar que son pocos los dispositivos que cuentan con estos apuntadores, aunque se pueden generar para agilizar diversas tareas.

Para desmontar un dispositivo se deben seguir ciertos lineamientos, primero el dispositivo no debe estar en uso, esto quiere decir que nadie debe estar realizando tareas como lectura o escritura, además una unidad no se puede desmontar si alguien se encuentra colocado en la estructura de directorios de dicha unidad. Se utiliza el comando *umount*, su sintaxis es:

umount [punto_de_montaje]

Los detalles de las unidades montadas en el sistema se encuentran en el archivo *fstab*. Para entender el contenido del archivo */etc/fstab* se presenta la siguiente tabla describiendo los campos:

/dev/hda5 /respaldo ext3 defaults 0 0

/dev/hda5	Partición o dispositivo
/respado	Punto de montaje en el sistema
ext3	Tipo de sistema
defaults	Opciones de montaje asociadas al dispositivo
0	Sistemas que requieren *dump*
0	Orden en que son revisados los sistemas.

Por otro lado el archivo */etc/mtab* describe los sistemas que están actualmente montados, es muy parecido a */etc/fstab.*

Administración del área de *swap.*

En muy raras ocasiones tendrá que hacer modificaciones a la configuración de *swap*, aunque puede encontrarse en el caso de que la memoria física se encuentre saturada y la *swap* no sea lo suficientemente grande como para resolver las necesidades del sistema. Se pueden hacer arreglos de dos formas, uno es hacer un archivo en el sistema el cual se comportará como espacio de *swap*, la otra es hacer una partición en disco duro la cual sustituirá a la que ya se tiene hecha.

Para la primera opción se realizan los siguientes pasos:

1. Determinar el tamaño del archivo que se utilizará multiplicando el tamaño de bloque por el tamaño deseado de la nueva *swap* en *megabytes*. Suponiendo que se va a necesitar un archivo de 64 Mb con bloques de 1024 bites cada uno: 1024 x 64 = 65536 Mb.

2. Crear el archivo (en este caso en el directorio raíz y asignándole el nombre *swapfile*) utilizando el comando *dd* de la siguiente forma:
   dd if=/dev/zero of=/swapfile bs=1024 count=65536

3. Indicar al sistema que este archivo será utilizado como *swap:*
   mkswap /swapfile

4. Activar el archivo *swap* para que comience a ser utilizado:
   swapon /swapfile

5. Para que se active al inicio del sistema se debe modificar el archivo */etc/fstab*, agregando la siguiente línea:
   /swapfile swap swap defaults 0 0

6. Si se tiene un área *swap* activada, ésta tiene que desactivarse antes de poner en funcionamiento el archivo:
   swapoff /dev/[partición_swap]

Para tener una *swap* como partición:

1. Crear la partición.
2. Cambiarla a tipo 82 (s*wap*)
3. Darle formato de tipo *swap* con: *mkswap /dev/[partición]*
4. Habilitarla con el comando *swapon /dev/[partición]*
5. Habilitarla en el archivo */etc/fstab*

Ahora ya se tiene una nueva área de *swap* instalada, recuerde que si hay un espacio de *swap* con anterioridad, debe desactivarse para cambiarlo por el nuevo.

Interfaces gráficas

Linux incorpora varias interfaces gráficas, entre las cuales se puede elegir la que se desea utilizar por medio de un menú desplegable al momento de iniciar sesión en la interfaz gráfica, cada una de ellas tiene sus propias cualidades. Sin embargo hay dos que tienen mayor aceptación por parte de los usuarios: KDE y GNOME, destacando KDE por ser la interfaz estándar sobre Linux ya que es muy flexible y de fácil manejo. Existen diferentes comandos para iniciar la sesión gráfica en Linux: *startx*, *xdm*, *kdm* o *gdm.* Cada uno abre un administrador de sesiones gráficas para la interfaz de KDM o GNOME. Otra opción es cambiar el modo de inicio por *default* en el archivo */etc/inittab.*

La versión 10 de Slackware sólo instala el idioma de inglés en su interfaz gráfica por *default*, por lo que toda la interfaz estará en inglés. Si se desea cambiar el idioma a español habrá que instalarlo uno mismo, en

el segundo disco de instalación se encuentra un archivo llamado *kde-i18n-es-3.2.3-noarch-1.tgz* el cual contiene el idioma español para KDE.

Después de descomprimirlo y seguir las instrucciones de instalación se ejecuta el comando *kpersonalize*, el cual abre un asistente de configuración básica para la interfaz gráfica KDE. Dicho asistente muestra los idiomas disponibles de la interfaz, se elige el deseado y a continuación se puede continuar la configuración o aplicar los cambios actuales y cerrar el asistente.

Finalmente es importante mencionar que las interfaces gráficas cuentan con aplicaciones preinstaladas como editores gráficos de texto, reproductores multimedia, navegadores de Internet, etc. También integran versiones gráficas de algunas aplicaciones de administración del sistema como visores de puntos de montaje, programadores de tareas, administradores de cuentas de usuarios, entre otros. Las versiones de estas aplicaciones varían dependiendo de la propia versión y el tipo de interfaz que se utilice.

I.3 **Editores para la creación de páginas web**

OBJETIVO ESPECÍFICO:
El participante elaborará páginas WWW, apoyado con HTML.

TEMARIO:

1. Introducción
2. Etiquetas (*markup tags*)
3. Formato de caracteres
4. Ligas
5. Imágenes
6. Tablas
7. Elementos avanzados

Para el diseño de páginas es necesario establecer algunos de los parámetros más importantes a considerar como son los siguientes:

a) Objetivo principal de la página

Es importante definir el objetivo principal de la página WEB a crear, ya que este parámetro permitirá al diseñador el establecer las prioridades en su desarrollo, así como el enfoque de la información y la presentación de la misma.

b) Información necesaria para el desarrollo de la página

Una vez establecido el objetivo de la página, se procede a hacer una recopilación del material escrito y gráfico que permitirá conformar el contenido de la misma, dicha información puede obtenerse haciendo un estudio del tema o bien apoyándose en información y archivos que pueden ser proporcionados por la empresa o de la propia Internet.

c) Software empleado para el desarrollo de la página

El diseño de una página Web se ve apoyado en gran medida al emplear un editor de HTML, el cual facilita las tareas a realizar ya que asiste al diseñador, construyendo por si sólo, las etiquetas de HTML, además de brindarle una referencia rápida de etiquetas, así como la versatilidad de manipular la información empleando la filosofía de los procesadores de texto.

Editores de HTML.

Los editores de HTML son un software que facilita al usuario la creación de páginas de WEB, empleando para su construcción la filosofía de los procesadores de palabra que consiste en permitir que el usuario vaya capturando el texto y después, mediante las acciones de marcar y seleccionar opciones o botones, se va dando formato y presentación, además de que cuenta con una

serie de asistentes que facilitan al usuario la inserción imágenes y vínculos o ligas de *hypertexto* hacia otras páginas y servicios.
Algunos ejemplos de editores de HTML son:

- Front-page (Microsoft)
- Hot Dog
- Quanta
- Dreamweaver

d) Manipulación y tratamiento de imágenes

Las imágenes, son una parte muy importante de cualquier página WEB por lo que es necesario el dominar algunos aspectos básicos tales como cambios de tamaño, cambio de formato de archivo, modificación de atributos y edición de las mismas, ya que todo esto permitirá dar presentación a la página e incluso generar sus propias animaciones de imágenes.

e) Equipo necesario para el montaje y/o desarrollo de la página

Para desarrollar una página WEB es necesario contar con un equipo de cómputo capaz de manipular texto e imágenes en el cual se puede ir preparando la página y un Servidor de HTML o WEB que permita aceptar y atender las peticiones que llegan a través de Internet para consultar la página. El equipo de cómputo debe ser capaz de ejecutar programas para manipulación y tratamiento de imágenes, además de contar con los recursos necesarios para ejecutar un visualizador de páginas WEB y un editor de HTML. El equipo puede ser del tipo PC compatible con IBM, Mac, Unix u otra plataforma que este disponible y cumpla con los requisitos anteriores.

Un servidor WEB es un programa que atiende los requerimientos de conexión emitidos por un navegador WEB. Obtiene la información acerca de la máquina solicitante, busca en los archivos de configuración del servidor para saber si hay instrucciones referentes a la aceptación o rechazo de solicitudes de la máquina que está llamando y actúa en consecuencia. Si la conexión se acepta, el servidor busca en el equipo y trata de encontrar el archivo o programa requerido para enviarlo al cliente (figura 3.1).

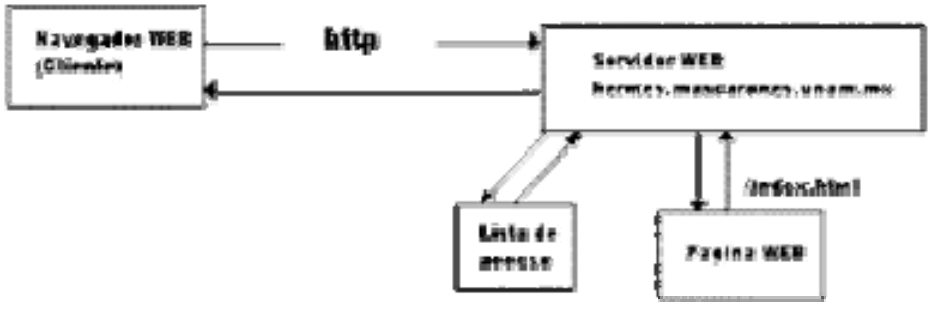


Figura 3.1: Funcionamiento de un servidor.

f) Publicación y difusión de la página.

Una vez que la página (o páginas) están terminadas, el siguiente paso es proceder a montar o publicar la página en el Servidor de WEB; esto puede normalmente resumirse a copiar o transferir todos los archivos necesarios, como las páginas, imágenes y otros hacia un directorio establecido por el administrador del Servidor de WEB, vía copia directa, FTP u otro medio disponible.

Una vez que los archivos se copian en el directorio indicado es necesario probar que todas las páginas y archivos se visualicen tal y como se esperaba para entonces proceder a la fase de difusión.

Una vez terminadas las fases de diseño y pruebas, es necesario el hacer una difusión de la dirección electrónica de la página para que esta sea consultada y cumpla con las expectativas con las que fue creada. Además de hacer difusión a través de la papelería generada por la empresa y los medios tradicionales (periódicos, radio, televisión, etc.) se debe de registrar esta nueva página en los sistemas de búsqueda en Internet más importantes, en los cuales el registro es, generalmente, sin costo y que son consultados diariamente por millones de personas en todo el mundo.

g) Actualización y mantenimiento de la página.

Finalmente, una vez terminada la página, es posible que se tengan que programar una serie de actividades encaminadas a estar actualizando y agregando información a la página según las necesidades de la empresa y la capacidad para actualizar esta información. Además es conveniente considerar que, por lo menos cada año, se debe plantear un nuevo diseño o actualización de la información tomando en cuenta los comentarios y cambios que puedan mejorar la página (hacer que cada año se proponga un nuevo diseño de página y se mejore en cada nueva versión).

Visualizadores de páginas.

Para poder navegar en el WEB y visualizar las páginas es necesario, además de una conexión a Internet, contar con un software conocido como visualizador, navegador o *browser*: el cual permite desplegar las páginas en el monitor de la computadora interpretando el código HTML que compone la página; además de desplegar texto e imágenes y otras aplicaciones que se incorporan a la página. Este software se puede adquirir en discos flexibles o bien a través de la propia Internet se pueden adquirir o emplear las versiones académicas que tienen un

costo menor o en algunos casos son de distribución gratuita y se obtienen vía FTP anónimo.

Visualizadores de páginas.

Algunos de los visualizadores más actuales, compatibles y con mayor número de opciones integradas son:

- Nestcape
- Internet Explorer
- Mozilla Firefox
- Opera

Estos visualizadores y muchos otros se pueden descargar de Internet desde las páginas de sus fabricantes. Asimismo, existen muchos y diversos navegadores más, siempre es recomendable tener más de uno para que en caso de problemas con alguno, se utilice otro.

Etiquetas de HTML.

Las etiquetas de HTML son el conjunto de instrucciones o comandos que después son interpretados por los visualizadores de WEB, desplegando como resultado de esta interpretación las páginas WEB. Las Etiquetas de HTML se interpretan en el orden en que se ponen y van encerradas entre los signos menor y mayor que (<, >). Estas etiquetas no son sensitivas de tal forma que se pueden escribir con mayúsculas o minúsculas indistintamente.

Estructura básica de una página de HTML.

Las páginas de HTML pueden escribirse en cualquier procesador de texto que permita guardar en el formato de sólo texto, o bien apoyándose en un editor de HTML que maneja ya el formato del archivo de texto por *default*. La estructura básica de una página debe llevar el siguiente esqueleto o estructura base:

```
<html>
  <head>
    <title>Espacio para título de la ventana</title>
  </head>
  <body>
    Espacio para escribir todo el contenido de la página.....
  </body>
</html>
```

Etiquetas correctas

Todo texto que se encuentre entre los caracteres “<” y, “>” se considerará una etiqueta. Si la etiqueta no fuera una de las validas del lenguaje HTML no será tenida en cuenta, sin causar ningún tipo de error, dejándose el texto o las etiquetas a las que afectaba como si no existiera la etiqueta extraña. Cuando se comete un error sintáctico al expresar una etiqueta o un atributo no se producirá ningún error, simplemente no de obtendrá el efecto que se desea.

Etiqueta	Función
<*html*>...</*html*>	Es la primera y la última etiqueta en un documento HTML y le indica al navegador que despliegue el documento como Hipertexto y no como texto sencillo.
<head>.. </head>	La etiqueta encabezado identifica la sección inicial del documento HTML. Entre otras cosas escribe el titulo del documento de esa sección
<title>...</title>	La etiqueta titulo contiene el titulo que aparece en la barra de titulo del navegador web.
<body>...</body>	La etiqueta de cuerpo abarca la parte más grande de una página Web. Contiene todo lo que el usuario ve cuando entra a esa página. La mayoría de las etiquetas se colocan dentro de BODY.
<hn>...</hn> para n=1..6	La etiqueta encabezado, crea varios tamaños de encabezados. La n se reemplaza por un número entre 1 y 6 cuanto menor sea el número mayor será el encabezado.
<hr>	La etiqueta de regla horizontal, dibuja una línea horizontal en medio de la página Web , haciendo más fácil su lectura.
 	La etiqueta de interrupción origina un salto de línea, con excepción de que coloca una línea en blanco como separador obligadamente.

Existen muchas etiquetas más para dar formato a textos, crear tablas, insertar hipervínculos e imágenes, así como para realizar muchas otras funciones en páginas HTML.

El lenguaje HTML evoluciona muy rápidamente y cada nueva versión de los programas navegadores presenta etiquetas nuevas que causan efectos más espectaculares o atributos nuevos de las etiquetas ya existentes. Esto causa que los programas más antiguos no entiendan estas nuevas etiquetas y por tanto las considere erróneas y no realice la acción que se desea, dándose el caso de atributos que son validos sólo para un único navegador.

Al escribir código HTML se debe hacer lo más estándar posible para permitir que el documento pueda ser visto de forma efectiva por distintos navegadores en máquinas distintas. Por tanto se debe renunciar a efectos espectaculares que sólo tienen validez en un navegador e intentar comprobar como se ve el documento en una variedad de navegadores, ya que las personas que se conectan a la página no tendrán en la mayoría de los casos la misma configuración o el mismo navegador que nosotros.

También puede resultar interesante como se vería el documento en los distintos tamaños de la ventana del navegador, teniendo en cuenta que todos no tienen un monitor con la misma resolución.

I.4 **Administración de servidores WWW con Linux**

OBJETIVO ESPECÍFICO:
El participante identificará el procedimiento para instalar, configurar y administrar su propio servidor de WVVW en un servidor de plataforma Linux.

TEMARIO:

1. Introducción a los servidores WWW
2. Instalación del servidor WWW
3. Configuración del servidor
4. Ejecución del servidor
5. Incorporación de módulos
6. Accesos restringidos
7. Registro de accesos
8. Manejo de sitios virtuales

Un servidor WWW ofrece servicios dentro de la *Word Wide Web* en Internet. Es un programa encargado de ofrecer comunicación mediante el protocolo HTTP (*Hypertext Transfer Protocol*).

HTTP es el protocolo de red para la WWW. Basa su operación en la arquitectura "Cliente- Servidor". El servidor HTTP es el encargado de publicar "recursos" electrónicos. El cliente HTTP consulta los recursos que el servidor ofrece. Es simple y poderoso.

Funcionamiento de HTTP (figura 4.1)

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

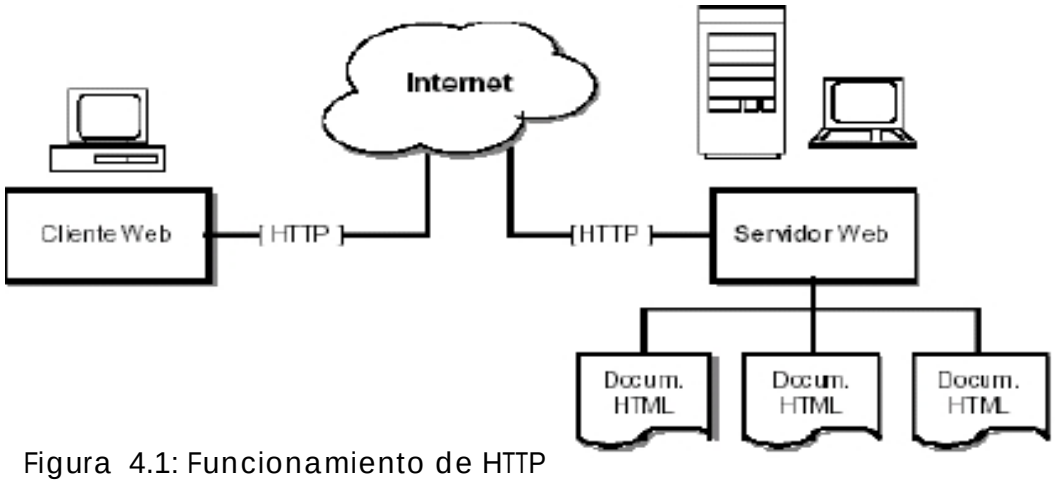


Figura 4.1: Funcionamiento de HTTP

Actualmente existe una gran variedad de servidores WWW que se pueden adquirir y montar en un sistema. Cada uno requiere características diferentes en el equipo donde se desea instalar, así como otras diferencias que se deben tomar en cuenta antes de decidirse por alguno. Entre los servidores de WWW más populares están:

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

Para seleccionar el servidor que cubra mejor nuestras necesidades se deben tomar en cuenta los siguientes criterios de selección:

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

Una vez evaluadas todos los criterios mencionados se procede a la selección del servidor a instalar. En el caso de este informe se eligió utilizar el servidor WWW Apache, el cual es el de mayor difusión actualmente, se encuentra instalado en el 70% de los sitios actuales y sigue creciendo con el número de personas nuevas que entran a cada minuto a la red por primera vez. Las principales ventajas de utilizar Apache:

- Robusto, soporta un gran número 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 módulos.
- Incluye el código fuente del servidor.
- Soporte para accesos restringidos.
- Soporte para SSL.
- Es gratuito.

El segundo servidor WWW actualmente más difundido es IIS de Microsoft. Las principales causas que llevan a no utilizarlo son:

- Gran presencia de virus que lo atacan, como por ejemplo:
  - *Red Code*
  - *Nimda*

- Cuenta con un débil sistema de seguridad, por lo tanto mucha exposición a ataques de *hackers* que atacan problemas como *Buffer Overflows*, entre otros.
- Sólo se puede utilizar en el sistema operativo Windows, el cual es muy conocido en el mundo de las redes por ser inestable e inseguro, dando como resultado un gran problema de seguridad.
- Tiene una tecnología propietaria, es decir que no se puede exportar hacia otra plataforma o en otro entorno de *harware* distinto.

Configuración básica

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 compila Apache.

Grupos de Directivas

En Apache la configuración se realiza mediante directivas organizadas en tres grupos: *Global Environment* que administra las directivas generales de operación, *Main Server* que administra las directivas del servidor principal o estándar y *Virtual Host* que administra las directivas donde los mismos procesos soportan diversas IP's o nombres de dominio.

Directivas *Global Environment*

ServerName

Define el nombre para el servidor Apache, el cual será utilizado cuando se construyan los URL's cuando el cliente solicite otros recursos web (páginas, PHP's, cgis, etc.) del servidor.

User

Esta directiva define el *user ID* mediante el cual Apache operará. Valor por *default*: *User #-1*

ServerAdmin

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

DocumentRoot

Esta directiva define la ruta absoluta donde se almacenarán los archivos HTML que se desean publicar. Valores por *default*: compilado: */usr/local/apache/htdocs.* Instalado: */var/www/html.* Esta etiqueta se utiliza tanto en *ServerConfig* como en *VirtualHost.*

ServerRoot

Esta directiva define la ruta absoluta donde los directorios *conf* y *logs* podrán ser encontrados. Valor por *default*: */usr/local/apache*. Esta etiqueta se utiliza tanto en *ServerConfig* como en *VirtualHost*.

MinSpareServers

Esta directiva define cual es el número mínimo de procesos servidores que son mantenidos en *spare*. Se monitorea el número de conexiones en espera si ésta es menor a *MinSpareServers* se levantan los *deamons* necesarios. Valor por *default*: 5

MaxSpareServers

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

ErrorDocument

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

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

Directivas *Main Server*

Directory y *DirectoryMatch*

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

<Directory /usr/local/Apache/htdocs>...</Directory>

Options

La directiva *options* controla que características del servidor están disponibles para un directorio en particular. Su sintaxis es:

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

A *option* puede ser colocado *none* en caso de que no se desee ninguna funcionalidad adicional. Las demás 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 <alias> <path-filesystem>

Convierte las solicitudes vía URL al *path* absoluto donde residen los programas CGI. *Esta etiqueta se utiliza tanto en ServerConfig como en VirtualHost.* Su sintaxis es:

ScriptAlias /cgi-bin /usr/www/cgi-bin

AddHandler cgi-scripts .cgi .pl

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 identificada como un *script*. Apache las interpretará en lugar de sólo enviar el contenido del archivo al cliente.

Server-Side Includes

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.

CGI

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.

Los sitios web dinámicos de la actualidad cuentan al menos con una de las siguientes tecnologías implementadas para hacer su navegación más dinámica de lo que sería con el lenguaje HTML solamente:

- PHP
- JSP
- Servlets
- ASP
- ColdFusion
- ActiveX
- JavaScript
- Applets
- CGIs

Control de Acceso

Apache ofrece la funcionalidad al estilo "ACL", con lo cual es factible determinar las direcciones IP y los usuarios que tendrán acceso sobre recursos específicos del servidor Web. Permiten definir un conjunto de clientes a los que se les pueden permitir o negar el acceso al servicio de Web. La definición de los nombres se puede realizar mediante el nombre parcial de un dominio parcial, una dirección IP, la pareja Red y mascara y/ó un dirección de red definida por *Classless Inter-Domain Routing* (CIDR). 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* o *deny* serán implementadas

Allow
: Define los *hosts* que tendrán acceso al recurso.

Deny
: Define los *hosts* que no tendrán acceso al recurso.

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

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

Módulos

En Apache los módulos son los encargados de agregar funcionalidad adicional incrementando el número de directivas disponibles.

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

Bitácoras

Apache cuenta con dos archivos donde residen las bitácoras:

- *access.log*
  Registra todos los accesos al sitio.
- *error.log*
  Registra los errores que genere un acceso al sitio o que los procesos de Apache reporten.

I.5 **Programación con PHP**

OBJETIVO ESPECÍFICO:
Proporcionar al participante los conocimientos necesarios que le permitan crear aplicaciones dinámicas e interactivas para el Web utilizando el lenguaje PHP.

TEMARIO:

1. Introducción al lenguaje PHP
2. Herramientas elementales
3. *Common Gateway Interface*
4. Cómo diseñar una aplicación CGI
5. Algunas funciones útiles
6. PHP y la programación orientada a objetos

PHP significa *Hypertext Preprocessor*, aunque originalmente significaba *Personal Home Page Tools*. Los archivos PHP normalmente se denominan con la extensión *php*, *php3* o *phtml*.

El PHP es un lenguaje embebido en páginas HTML y que se ejecutan el servidor. Productos similares y propietarios son *Active Server Pages* (ASP) de Microsoft, *ColdFusion* de Allaire y *Java Server Pages* (JSP) de Sun.

PHP es fácil de aprender comparado con otros mecanismos para obtener la misma funcionalidad. A diferencia de JSP o CGI basados en C, PHP no requiere un conocimiento exhaustivo del lenguaje de programación. A diferencia de Perl, PHP tiene una sintaxis muy fácil de comprender y a diferencia de ASP, no requiere conocer más de un lenguaje de programación o de la instalación de módulos externos o comerciales para realizar tareas más complicadas no previstas en el lenguaje más usado (*Visual Basic Script*).

La mayoría de las funciones más útiles están predefinidas:

- Acceso a bases de datos: *ODBC, Oracle, Postgres, SQL Server, MySQL, Informix, Interbase, SyBase, mSQL, dBase,* entre otras.
- Conectividad: HTTP, FTP, COM, YP/NIS, SNMP, Sockets, CORBA, LDAP
- Servicios Correo y Noticias: POP, IMAP, SMTP, NNTP
- Textos y Gráficos: XML, HTML, PDF, GD, Flash
- Funciones Matemáticas.
- POSIX: semáforos, memoria compartida, acceso a archivos, expresiones regulares, cronómetros...
- Comercio Electrónico: Cybercash, Verisign
- Formularios.
- Encriptación y Compresión: MD5, Gzip, Bzip2, OpenSSL...

Las instrucciones PHP están embebidas en HTML. Una página PHP es una página normal HTML que con unas marcas especiales le indican al servidor que deben interpretarse.

Cuando un cliente solicita una página, el servidor web la procesa en forma secuencial desde el principio al final buscando secciones PHP limitadas por **<?** y **?>**. En caso de encontrarlas, las compila y ejecuta.

El resultado es idéntico a si hubiese sido escrito el texto manualmente.

Esto tiene algunas consecuencias:

- PHP puede ser agregado rápidamente al código HTML producido por editores HTML interactivos.
- PHP facilita la interacción entre diseñadores y programadores.
- No se necesita reescribir cada línea de código HTML en un lenguaje de programación.
- PHP reduce costos y aumenta la eficiencia.

Agregar PHP a HTML

PHP es totalmente compatible con HTML y no tomará en cuenta la inclusión de *applets, Javascript*, etc., simplemente los ignorará. Se puede usar cualquier método para generar HTML y luego se puede agregar PHP en él. Para indicar las secciones PHP se deben usar etiquetas especiales, este proceso es llamado "escape del HTML". Las etiquetas válidas son:

1. Etiquetas canónicas PHP: *<?php ?>*
2. Etiquetas cortas (tipo SGML): *<? ?>*
3. Estilo ASP: <% %> (se debe verificar que esté habilitada esta opción en el *php.ini*)
4. Etiquetas estilo HTML: <SCRIPT LANGUAGE="PHP"> </SCRIPT>

Entrar y Salir del Modo PHP

En cualquier momento se puede entrar y salir del modo PHP. Todo lo que esté entre las etiquetas de escape es considerado PHP, todo lo que esté afuera es ignorados, no hay término medio.

Inclusión de archivos PHP

Otra forma de agregar código PHP al HTML es poniendo el código PHP en otro archivo e invocarlo mediante la función *include*.

También se suele usar la función *require* en vez de *include*. Aunque *include* es más flexible, *require* es más rápido. *Include* también permite la inclusión de HTML debido a que se interpreta en modo HTML.

PHP es bastante flexible, más que intentar ser estricto y forzar una disciplina en la programación, enfatiza conveniencia para el programador más que la corrección. El PHP tiene un conjunto mínimo de reglas que hay que seguir, en caso contrario se pueden ver los mensajes de error tipo "*Parser error in line XXX*".

Sintaxis

La sintaxis del PHP es similar a la del lenguaje C, muy sencilla. Si no se sabe como escribir una instrucción, se prueba primero como lo haría en C, y si no funciona se acude al manual. PHP ignora los espacios en blanco.

En algunos casos PHP no es sensitivo a las mayúsculas o minúsculas, como en el caso de nombres de función o construcciones del lenguaje (*if, then. else, while, for,* etc.), pero para otros sí que lo es, por ejemplo para nombres de variables.

Las sentencias son expresiones que terminan en punto y coma. La típica sentencia en PHP es una asignación.

Los bloques de construcción más pequeños son las palabras indivisibles, tal como números (2.718281), cadenas ("uno, dos, tres"), variables ($saludos), constantes (*TRUE, FALSE*) y las construcciones que definen el lenguaje (*if, then* ...). Estas palabras se separan de las demás por espacios en blanco o caracteres especiales que hacen de separador, como paréntesis, operadores, llaves, etc. Los siguientes elementos en complejidad son las expresiones, que son una combinación de palabras que tienen un valor. Las expresiones más simples están formadas por una sola palabra, como un número o una variable. Expresiones simples pueden ser combinadas con operadores para formar expresiones más complejas.

Evaluación de expresiones

Cada vez que el intérprete encuentra una expresión, la expresión es inmediatamente evaluada. Esto significa que el PHP primera evalúa los elementos más pequeños y luego las combina y obtiene su resultado.

El tipo más común de expresión en la asignación, donde el resultado de una expresión es almacenado en una variable. La forma es el nombre de la variable, que comienza siempre con el símbolo *$*, seguido de un símbolo = y a continuación la expresión a evaluar. Una cosa importante a recordar es que las asignaciones son expresiones y por la tanto también tienen un resultado, el mismo que se asigna a la variable:
Normalmente hay dos razones para usar una expresión en PHP: por su valor o por su efecto secundario. El valor de una expresión es aquello

que se pasa para la evaluación de expresiones más complejas, los efectos secundarios son aquellos que ocurren como resultado de una evaluación. Los casos más típicos de efectos son la asignación de resultados a una variable, imprimir algo en la pantalla del usuario o cambiando valores en una base de datos.

Aunque las sentencias son expresiones, no están incluidas en expresiones más complejas, lo que significa que la única razón para usar una sentencia es el efecto secundario.

Construcción de Bloques

Aunque las sentencias no pueden ser combinadas como las expresiones, se puede poner una serie de sentencias en cualquier lugar permitido agrupándolas con "{" y "}".
Por ejemplo, la construcción *if* está formada por una verificación (entre paréntesis) seguido de una sentencia que es ejecutada si la condición es verdadera. Si se quiere que se ejecuten varias sentencias en vez de una sola, lo que se hace es agrupar aquellas sentencias.

El sangrado, aunque ignorado por el intérprete del PHP, es muy importante para la comprensión de los programas.

Comentarios

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

- Comentarios tipo C:

  Hay que tener en cuenta que los comentarios de este tipo no pueden estar anidados:

  ```
  /* Este es un comentario
  similar al del lenguaje C */
  ```

- *Comentarios de una línea: // y #*

  Estos tipos de comentarios están heredados del C++, Java, *Shell* y *Perl*.

  ```
  // esta es una línea
  # esto también es un comentario
  ```

Variables

La forma principal de almacenar valores en el medio de un programa son las variables. Tiene las siguientes características:

- Todas las variables en PHP comienzan con el símbolo dólar $ (heredado del *Shell* y *Perl*)

- ♣ El valor de una variable es igual al valor más recientemente asignado.
- ♣ Las variables son asignadas con el operado '=', con la variable a la izquierda del operador y la expresión a evaluar a la derecha.
- ♣ Las variables no necesitan ser declaradas antes de ser usadas.
- ♣ Las variables no tienen un valor intrínseco, sino que toman el tipo del último valor asignado.
- ♣ Las variables que se usan antes de ser asignadas tienen un valor por defecto.

Ámbito de las variables

El ámbito (*scope*) es el término técnico para definir el comportamiento de un nombre, función o variable, dentro de un programa. Puede darse el caso que el nombre tenga el mismo comportamiento y significado en todo el programa (*global*) o que el mismo nombre se comporte de manera diferente en distintas partes del programa. En este último caso se dice que el ámbito de esas variables es local.

En PHP, cualquier variable que no esté dentro de una función tiene ámbito global y su valor se encuentra disponible en toda la extensión de la ejecución del programa. En otras palabras, si se asigna un valor a una variable al principio del programa, el nombre de la variable tiene el mismo significado para el resto del programa y tendrá siempre el mismo valor.

La asignación de un valor a una variable no afectará a las variables con el mismo nombre en la ejecución del PHP de otras páginas ni tampoco a la ejecución del mismo programa PHP cuando es invocado por conexiones diferentes. La pregunta de si una variable persiste a través de las diferentes etiquetas PHP, es decir si se puede salir del modo PHP, volver a entrar luego y usar las mismas variables con los mismos valores de una asignación previa: si, no hay problemas.

La mayoría de las construcciones del PHP se ejecutan silenciosamente, es decir no producen ninguna salida hacia la pantalla del usuario. Si se desea enviar texto al navegador del usuario se deben usar las funciones de salida.

Echo y *print*
Las dos construcciones básicas para generar salida son *echo* y *print*. Aunque al principio su sintaxis puede parecer un poco confusa, esto se debe a que son construcciones básicas del lenguaje y no funciones, por lo tanto se puede o no usar paréntesis. Su sintaxis es:

echo "cadena";	*print "cadena";*
echo ("cadena");	*print ("cadena");*

Se pueden usar *echo* y *printf* para imprimir de forma combinada cadenas y variables.

Es uso de las comillas simples o dobles cambia el comportamiento. Si se usan comillas dobles, el PHP interpolará (cambiará) los nombres de variables que encuentre por su valor. En cambio si se usan comillas simples, el PHP lo tomará con una cadena que debe imprimirse tal cual han sido escritas.

Es posible generar una salida con formatos más complejos mediante el uso de la función *printf*, con una sintaxis similar al lenguaje C o Perl.

Tipos de variables

Todos los lenguajes de programación tienen un sistema que especifican los diferentes tipos de datos que pueden aparecer en los programas. Normalmente los diferentes tipos corresponden a la forma en que representan los valores en la memoria mediante una serie de bits. El sistema de tipos en PHP es extremadamente sencillo y flexible lo que facilita la tarea de los programadores.

Los tipos básicos de PHP son enteros (*integers*), flotantes (*doubles*), lógicos (*booleans*), cadenas (*strings*), vectores (*arrays*) y objetos.

- *Integers* son números enteros, sin punto decimal.
- *Doubles* son números de punto flotante.
- *Booleans* son valores lógicos y sólo permiten TRUE o FALSE.
- *Strings* son secuencias de caracteres.
- *Arrays* son una colección de datos indexados por alguna clave.

A excepción de unos pocos casos, las cadenas con comillas simples se almacenan y leen literalmente a como fueron escritas. Las comillas dobles dentro de una cadena limitada por comillas simples no "corta" la cadena, sino que son almacenadas como un carácter norma.

Si se quiere almacenar el carácter ' dentro de la cadena, no es tan complicado, sólo hay que *escaparlo*, anteponiendo el carácter “\”.

Las cadenas limitadas por comillas dobles son procesadas de dos maneras por el PHP:

1. Algunos caracteres que comienzan con barra invertida ('\') son reemplazados por caracteres especiales.
2. Los nombres de variables (que comienzan con $) son reemplazados por una representación de cadena del valor que almacenan.

Los caracteres especiales son:

- \n se reemplaza por un carácter de salto de línea.
- \r se reemplaza por un "retorno de carro" (<ENTER>)
- \t se reemplaza por una tabulación
- \$ se reemplaza por un símbolo dólar ($)
- \" se reemplaza por comillas dobles
- \ \ se reemplaza por una barra invertida ('\').

Interpolación de variables

Siempre que aparezca un símbolo $ que no haya sido *escapado*, el PHP intenta interpretarlo como un nombre de variable e inserta el valor de la variable en la cadena. Las reglas son:

- Si la variable tiene como valor una cadena, dicha cadena es insertada en la cadena actual (si está limitada por comillas dobles).
- Si la variable tiene un valor que no sea del tipo *String*, primero se convierte a un *String* y luego es insertado.
- Si la variable no tiene ningún valor asignado, no se inserta nada.

Arreglos *(Arrays)*

El PHP ofrece la posibilidad de agrupar un conjunto de valores para almacenarlos juntos y referenciarlos por un índice. Los índices puedes ser del tipo numérico (entero) o una cadena de forma indistinta.

Verificación de tipos.

Cuando se necesita saber el tipo de variable que se esta manejando se utilizan las siguientes funciones:

- *gettype(arg)*
  Retorna un string representando el tipo de argumento: *integer*, *double*, *string, array, object* o *unknown type*.
- *is_int(arg), is_integer(arg), is_long(arg)*
  Retorna verdadero si *arg* es de tipo entero, falso en caso contrario.
- *is_double(arg), is_float(arg), is_real(arg)*
  Retorna verdadero si *arg* es un *double*, falso en caso contrario.
- *is_bool(arg)*
  Retorna verdadero si *arg* es del tipo *Boolean* (TRUE o FALSE) y falso si no lo es.
- *is_string(arg)*
  Retorna verdadero si *arg* es un *string*.
- *is_array(arg)*
  Retorna verdadero si *arg* es un *array*.
- *is_object(arg)*
  Retorna verdadero si *arg* es un objeto.

Control de Flujo

Es casi imposible hacer programas útiles si no se pudiese hacer que la ejecución del programa dependiese de determinados valores. Este tipo de ejecución requiere de estructuras de control que indican que partes del código deben ejecutarse en distintas situaciones.

Hay dos tipos básicos de estructuras de control:

1. Ramificaciones (*branches*): *if-else-elseif, switch-case*
2. Ciclos (*loops*): *while, do-while, for*

Todas las estructuras de control tienen dos partes:

- Evaluación, que determina que parte del resto de la estructura se ejecuta. Funciona evaluando una expresión booleana.
- Código, como una rama separada de código o el cuerpo de un ciclo.

La forma más sencilla de expresiones booleana son las constantes TRUE y FALSE.

Operadores Lógicos

Los operadores lógicos combinan otras expresiones lógicas para producir un nuevo valor lógico (o booleano).

- *and, &&:* Verdadero si ambas condiciones son verdaderas
- *or*, | |: Verdadero si una de las condiciones es verdadera.
- *!:* Verdadero si la expresión de la derecha es falsa, falso si la expresión de la derecha es verdadera.
- *xor*: Verdadero si alguna, pero no ambas, de las condiciones es verdadera.

Operadores de Comparación

- == (*igual*): Verdadero si ambos argumentos son iguales.
- != (*distinto*): Verdadero si ambos argumentos son distintos.
- < (*menor que*): Verdadero si el argumento de la izquierda es menor que el de la derecha.
- (*mayor que*) Verdadero si el argumento de la izquierda es menor que el de la derecha.
- <= (*menor o igual*): Verdadero si el argumento de la izquierda es menor o igual que el de la derecha.
- >= (*mayor o igual*) Verdadero si el argumento de la izquierda es mayor o igual que el de la derecha.
- === (*idéntico*): Verdadero si ambos argumentos son del mismo tipo y tienen el mismo valor.

Ramificación

Las dos estructuras principales son *if* y *switch*. *If* es muy utilizada y es la primera estructura que se aprende. *Switch* es útil para los casos en que se tengan múltiples ramas que dependen de la evaluación de un sólo valor. *Elseif* se suele usar cuando se tienen comparaciones en cascada.

El *do-while* es similar al *while* excepto que la verificación se realiza al final del ciclo, es decir que el ciclo se ejecuta por lo menos una vez.

Funciones

Todos los lenguajes de programación modernos proveen capacidades de abstracción de procedimientos que facilita enormemente la programación y el mantenimiento del código. El mecanismo de PHP (al igual que C) es la función. Hay dos clases de funciones:

1. Provistas por el propio lenguaje: *phpinf*, *strtok*, *exit*…
2. Definidas por el propio programador.

La sintaxis básica para el uso (llamado) de funciones es el nombre de la función seguida, entre paréntesis, de una lista de expresiones separadas por coma:

nombre_funcion(expresion_1, expresion_2, ..., expresion_N)

Cuando el PHP encuentra una llamada a una función, primero evalúa las expresiones especificadas como argumentos y usa sus resultados como valores de entrada a la función. Después de la ejecución de la función, si hay algún valor de retorno, es el resultado de la expresión de llamada a función.

Todas las llamadas a funciones son expresiones de PHP, y como cualquier otra expresión, hay dos razones por las que puede interesar llamar una función: por el valor que devuelve o por su efecto secundario.
El valor devuelto por una función es el valor de la expresión, se puede hacer lo mismo que se hace con una expresión normal.

Las funciones de usuario no son obligatorias en PHP, sin embargo facilitan la programación en caso que el código se vuelva extenso o requiera la ejecución de tareas complejas. Un función es un trozo de código al cuál se le da un nombre y que puede ser llamada una o varias veces desde distintas partes del programa. La sintaxis es:

```
function nombre_de_funcion ($argumento_1, ..., $argumento_N)
{
    sentencia_1;
    ...
}
```

La palabra reservada *function* indica el inicio de la definición de la función. El nombre de la función debe ser construido de forma similar a las variables, puede estar formado de letras, números y guión bajo (_) y no debe comenzar con un número.

Los argumentos son variables de ámbito local en la función. Las operaciones que hace el intérprete cuando encuentra una llamada a función son:

1. El PHP busca la función por el nombre, si no está definida genera un mensaje de error.
2. Sustituye los valores de las expresiones en las llamadas en las variables indicadas como argumentos de la función. Los valores son pasados por copia, no por referencia.
3. Las sentencias en el cuerpo de la función son ejecutadas. Si alguna de las sentencias ejecutadas es un *return*, devuelve ese valor, caso contrario la función finaliza en la última sentencia ejecutada y no devuelve ningún valor.

I.6 **Interacción de WWW con bases de datos**

OBJETIVO ESPECÍFICO:
El alumno conocerá y desarrollará una aplicación de bases de datos que funcione a través del WWW empleando herramientas de software libre.

TEMARIO:
1. Introducción
2. Manejo de formularios como *front-end.*
3. Instalación y configuración de la base de datos en Linux
4. Introducción al desarrollo de CGIS
5. Desarrollo del *back-end* con MySQL
6. Desarrollo de una aplicación personalizada

MySQL fue desarrollado por la compañía MySQL AB. MySQL es uno de los DBMS libre más usado para los sitios Web. Antes, MySQL se consideraba como la opción ideal para sitios Web, sin embargo, ahora incorpora muchas de las funciones necesarias para otros entornos y conserva su velocidad.

Algunas de las ventajas de MySQL son:

- Costo: MySQL es gratuito para la mayor parte de los usos y su servicio de asistencia resulta económico.
- Asistencia: MySQL AB ofrece contratos de asistencia a precios razonables y existe una activa comunidad MySQL.
- Velocidad: MySQL es mucho más rápido que la mayor parte de sus rivales.
- Funcionalidad: MySQL dispone de muchas de las funciones que exigen los desarrolladores profesiones, como compatibilidad para la mayor parte de SQL ANSI, duplicación, funciones SSL e integración con la mayor parte de los entornos de programación.
- Portabilidad: MySQL se ejecuta en la inmensa mayoría de sistemas operativos y, la mayor parte de los casos, los datos se pueden transferir de un sistema a otro sin dificultad.
- Facilidad de uso: MySQL resulta fácil de utilizar y administrar. Las herramientas de MySQL son potentes y flexibles, sin sacrificar su capacidad de uso.

La base de datos MySQL

Cuando se instala MySQL, la base de datos *mysql* es una de las que se crea automáticamente. El conocimiento de las tablas de esta base de datos resulta fundamental para poder administrar de forma eficaz la seguridad en el sistema.

Hay cinco tablas principales en la base de datos "*mysql'* que afectan el acceso al sistema: *user, db, host, tables_priv, columns_priv.*

Tabla	Descripción
user	Enumera los usuarios y los equipos y contraseñas asociadas que pueden acceder al servidor, así como los permisos globales que tienen.
db	Enumera las bases de datos a las que pueden acceder los usuarios. Los permisos otorgados se aplican a todas las tablas de la base de datos.
host	Junto con la tabla *db*, permite un acceso más controlado en función de un determinado equipo.
tables_priv	Enumera el acceso a tablas concretas. Los permisos otorgados se aplican a todas las columnas de la tabla.
columns_priv	Enumera el acceso a columnas específicas

Los comandos básicos de MySQL se pueden clasificar en categorías. Los más utilizados son:

1) Iniciar y cerrar MySQL
   * Levantar el servicio
     mysqld_safe --user=mysql &
   * Cerrar el servicio
     mysqladmin shutdown -u root -p
   * Para determinar los directorios de datos utilizados en una instalación existente se puede utilizar la opción de variables de *mysqladmin*:
     mysqladmin -u root -pmipassword variables | less
   * Establecer una conexión a la base de datos
     mysql -u root -p
   * Establecer *password* cuando el usuario *root* no tiene uno
     mysqladmin -u root password 'root_password'

2) Respaldos y restauración de la base de datos
   * Respaldo de una base de datos
     mysqldump base_datos -u root -p > /respaldos/base_resp.sql
   * Respaldo de una tabla de la base de datos
     mysqldump base_datos tabla -u root -p > /respaldos/tabla.sql
   * Restauración de la base de datos
     mysql base_datos -u root -p < /respaldos/base.sql

3) Comandos útiles para proporcionar información acerca de la base de datos
   * Usar una base de datos en particular. El nombre de la base de datos es sensible a mayúsculas y minúsculas.
     USE base_datos

* Base de datos actual
  SELECT database();
* Usuario actual
  SELECT user();
* Versión del servidor
  SELECT VERSION();
* Mostrar las tablas de la base de datos
  SHOW TABLES;
* Mostrar el estado de las tablas
  SHOW TABLE STATUS;
* Mostrar las columnas de una tabla en particular
  SHOW COLUMNS FROM NOMBRE_TABLA;
  DESCRIBE NOMBRE_TABLA;

Agregar usuarios remotos y locales a la base de datos

GRANT otorga a un usuario determinados permisos sobre los componentes de la base de datos, incluyendo a la base de datos. La sintaxis es la siguiente:
GRANT tipo_privilegio1, tipo_privilegio2 ON base_datos.tabla IDENTIFIED BY 'contraseña'
Revocar permisos
REVOKE SELECT ON mysql. FROM regular_user@localhost;*
REVOKE SELECT, INSERT ON FROM regular_user©localhost;

Cambiar el *password* del usuario en sesión:
SET PASSWORD = PASSWORD ('new_password');

Los siguientes son algunos de los tipos de privilegios disponibles que se pueden otorgar o revocar:

Privilegio	*Descripción*
ALL	Concede todos los permisos básicos (excepto GRANT)
ALL PRIVILEGES	Igual que ALL
ALTER	Permiso para cambiar la estructura de una tabla a excepción de los índices.
CREATE	Permiso para crear bases de datos y tablas, a excepción de los índices.
DELETE	Permiso para eliminar registros de una tabla.
DROP	Permiso para eliminar bases de datos o tablas, a excepción de los índices.
INDEX	Permiso para crear, modificar o barrar índices.
INSERT	Permiso para añadir nuevos registros a una tabla.
SHOW DATABASES	Permiso para ver todas las bases de datos.
SELECT	Permiso para devolver datos de una tabla.
SHUTDOWN	Permiso para cerrar el servidor.
UPDATE	Permiso para modificar datos de una tabla.

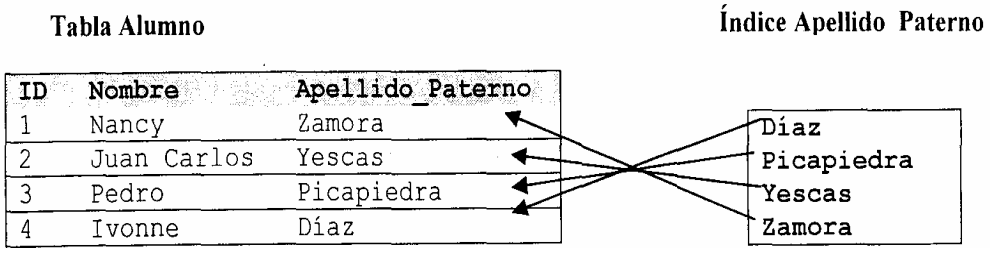
Los cambios realizados en los permisos no tienen efecto inmediato cuando se realizan directamente en las tablas MySQL. MySQL tiene que volver a leer las tablas modificadas. Para ello, se pueden emitir los siguientes comandos: *FLUSH PRIVILEGES, mysqladmin flush-privileges o mysqladmin reload*.

A continuación se muestran las expresiones utilizadas para otorgar y revocar permisos:

Tabla	Descripción
.	Todas las tablas de todas las bases de datos.
*	Todas las tablas de la base de datos actual.
nombreBD.*	Todas las tablas de la base de datos "nombreBD".
nombreBD.nombreTabla	La tabla "nombreTabla" de la base de datos "nombreDB".

INDICES

Cuando se necesitan localizar datos en una tabla, es necesario recorrer la tabla para buscar los patrones que se necesitan. La operación de recorrer la tabla de principio a fin examinando todos los registros se conoce como examen completo de la tabla. Cuando las tablas son de gran tamaño, esta operación resulta poco eficiente ya que la labor de examinar tablas compuestas de varios cientos de miles de registros puede resultar lenta.
Por tanto, cuando se necesita buscar ciertos campos en la tabla, la solución consiste en crear listas separadas para cada campo que se necesite ordenar. No contendrán todos los campos, sólo aquellos que se necesite ordenar y un puntero a un registro completo de la tabla. Estas tabla se denominan índices y son uno de los elementos menos y peor usados de las bases de datos relacionales. Los índices se almacenan en archivos separados en algunos casos (tablas *MyISAM*) o como parte de algún espacio de tabla en otros (tablas *InnoDB*).



Existen cuatro tipos de índice en MySQL: una clave primaria, un índice exclusivo, un índice de texto completo (tablas *MyISAM*) y un índice ordinario.

Creación de una clave primaria

Una clave primaria es un índice establecido sobre un campo en el que cada valor es exclusivo y ninguno de los valores es *NULL*. Para establecer una clave primaria al crear una tabla, se utiliza la instrucción *PRIMARY KEY*, cuya sintaxis es:

```
CREATE TABLE nombre_tabla(
campo1 tipo_dato,
campo2 tipo_dato,
CONSTRAINT pk_nombre_tabla PRIMARY KEY (campo1)
);
```

El término llave primaria es, estrictamente hablando, un término lógico, pero MySQL lo utiliza para denotar un índice físico. Cuando MySQL indica que existe una clave primaria, siempre existe un índice asociado.

Creación de índices ordinarios

Estos índices permiten el uso de valores duplicados, y se pueden crear de varias formas, utilizando la siguiente sintaxis:

```
CREATE TABLE nombre_de_tabla (
nombre_de_campo1 tipo_columna,
nombre_de_campo2 tipo_columna,
INDEX nombre_indice (nombre_de_campo1),
INDEX nombre_indice (nombre_de_campo2)
);
```

También se puede crear un índice en un momento posterior con el siguiente código:

```
ALTER TABLE nombre_tabla ADD INDEX nombre_indice (nombre_campo);
ó
CREATE INDEX nombre_indice ON nombre_tabla (nombre_campo);
```

Creación de un índice único

Un índice único es lo mismo que un índice ordinario con la salvedad de que no se permiten duplicaciones. Los índices únicos permiten campos nulos. Su sintaxis es:

```
CREATE TABLE nombre_tabla (
nombre_campo1 tipo_columna,
nombre_campo2 tipo_columna,
UNIQUE (nombre_campo)
);
```

Ó si la tabla ya existe, se puede utilizar la siguiente sintaxis:

ALTER TABLE nombre_tabla ADD UNIQUE nombre_indice (nombre_campo);
ó
CREATE UNIQUE INDEX nombre_indice ON nombre_tabla (nombre_campo);

Uso eficaz de los índices.

Las tablas con pocos índices devolverán los resultados muy despacio. Pero la inclusión de demasiados índices, aunque no suele ser normal, también ocasiona problemas. Los índices ocupan espacio de disco y, como están ordenados, cada vez que se realice una operación de inserción o de actualización, es necesario volver a organizar el índice para incluir los cambios, lo que da como resultado una carga de trabajo adicional significativa.

Es conveniente utilizar índices en los siguientes casos:

- El uso más común de un índice consiste en recuperar filas que cumplan una condición incluida en una cláusula *WHERE*.
- Si se solicitan valores máximos o mínimos con frecuencia, resultara extremadamente útil crear un índice sobre el campo pertinente.
- Otro caso en el que los índices resultan útiles es cuando se utiliza la instrucción *ORDER BY* para ordenar un campo.

Campo de Incremente Automático

Los campos de incremento automático resultan útiles porque permiten incrementar el valor de un campo automáticamente cada vez que se inserta un nuevo registro. Sólo se puede incrementar automáticamente un campo de un registro y dicho campo debe ser una clave primaria numérica o un índice exclusivo numérico. La sintaxis para crear un campo de incremento automático es la siguiente:

```
CREATE TABLE nombre_tabla (
nombre_campo1 INT AUTO_INCREMENT PRIMARY KEY,
nombre_campo2 tipo_columna
);
```

Para crear un campo de incremento automático en una tabla ya existente, se utiliza a siguiente sintaxis:

ALTER TABLE nombre_tabla MODIFY nombre_campo tipo_columna AUTO_INCREMENT;

Evaluación del rendimiento de las funciones y de equipos

La función *BENCHMARK ()* indica cuánto tiempo necesita MySQL para realizar una tarea un número dado de veces. Esta función brinda una idea general sobre la diferencia de potencia entre dos equipos, aunque su principal objetivo es ayudar a optimizar las funciones. Su sintaxis es la siguiente:

SELECT BENCHMARK (número_de_repeticiones, expresión);

Transacciones

Un conjunto de instrucciones que se deben realizar como una sola unidad, es llamado transacción. El ejemplo común es el de transferir una cantidad de efectivo de una cuenta *A* a otra cuenta *B*. Es claro que se necesitan dos actualizaciones, una para retirar el efectivo de la cuenta *A* y la otra para depositario en la cuenta *B*. Si el usuario declara que las dos actualizaciones son parte de la misma transacción, entonces el sistema puede en efecto garantizar que se hagan ya sea ambas o ninguna de ellas, aun cuando el sistema fallara a la mitad del proceso.

Las tablas *InnoDB* son tablas de transacción segura (lo que significa que disponen de las funciones *COMMIT* y *ROLLBACK*). En una tabla *MyISAM*, la tabla entera se bloquea al realizar funciones de inserción. Durante esa fracción de segundo, no se puede ejecutar ninguna otra instrucción sobre la tabla. *InnoDB* utiliza funciones de bloqueo en el nivel de fila de manera que sólo se bloquee dicha fila y no toda la tabla, y se pueden seguir aplicando instrucciones sobre otras filas.

```
BEGIN;
UPDATE nombre_tabla SET campo1=valor_campo1;
UPDATE nombre_tabla SET campo2=valor_campo2;
COMMIT;
```

BEGIN: Indica que va iniciar la transacción
COMMIT: Que la transacción ha terminado
ROLLBACK: Deshacer la transacción en caso de error.

I.7 **Introducción a la seguridad en cómputo.**

OBJETIVO ESPECÍFICO:
El participante reconocerá la importancia de la seguridad e identificará los elementos que le permitirán proteger el sistema y la información.

TEMARIO:

1. Introducción
2. Control de acceso
3. Sistema de archivos
4. Administración básica de la seguridad
5. Seguridad en red
6. Políticas de seguridad

Las redes fueron diseñadas para el intercambio de información y compartir recursos. La seguridad no era un factor tomado en cuenta en el diseño de las redes.

Desafortunadamente, el gran crecimiento reciente de las redes incluirá (incluye) individuos deshonestos que se introducen a los sistemas de información. Ninguna institución está libre del asecho de estos individuos. Todo el mundo alguna vez ha sido afectado por algún problema de seguridad.

Conceptos básicos

Amenaza — Una amenaza es una circunstancia o evento que puede causar daño violando la confidencialidad, integridad o disponibilidad. El término se refiere a un evento. Frecuentemente aprovecha una vulnerabilidad

Vulnerabilidad — Ausencia de una contramedida, o debilidad de la misma, de un sistema informático que permite que sus propiedades de sistema seguro sean violadas. Condición que tiene potencial para permitir que ocurra una amenaza, con mayor frecuencia e impacto. La debilidad puede originarse en el diseño, la implementación o en los procedimientos para operar y administrar el sistema. En el argot de la seguridad computacional una vulnerabilidad también es conocida como un *hoyo*.

Riesgo — El potencial para perdida o falla, como respuesta a las siguientes preguntas:
¿Qué podría pasar (o cual es la amenaza)?
¿Qué tan malo puede ser (impacto o consecuencia)?

¿Qué tan frecuente puede ocurrir (frecuencia)?
¿Qué tanta certidumbre se tiene en las primeras 3 respuestas (grado de confianza)?
Nótese que si no hay incertidumbre, no hay un riesgo *per se*.

Exploit — Se refiere a la forma de explotar una vulnerabilidad. Término muy enfocado a herramientas de ataque, sobre equipos de cómputo. Aprovechamiento automático de una vulnerabilidad. Generalmente se presenta en forma de un programa que realiza de forma automática un ataque aprovechándose de una vulnerabilidad.

Ataques — Acción o acciones que tienen por objetivo el que cualquier parte de un sistema de información automatizado, deje de funcionar de acuerdo con su propósito definido. Esto incluye cualquier acción que causa la destrucción, modificación o retraso del servicio no autorizado.

- ♦ No es un ataque físico (aunque puede ser).
- ♦ Un ataque no se realiza en un sólo paso.
- ♦ Depende de los objetivos del atacante.
- ♦ Puede consistir de varios pasos antes de llegar a su objetivo.

Principales Ataques: Virus, Caballo de Troya, Gusanos (*Worms*), *Backdoors*, *Stack Overflow*, Formatos de cadena, Bombas lógicas, Falsificación, Usurpación, *Sniffers*, *Spoofing*, *Spam*, *Grafiti*, Ingeniería Social, Negación de servicio, etc.

Seguridad Computacional

El conjunto de políticas y mecanismos que permiten garantizar la confidencialidad, la integridad y la disponibilidad de los recursos de un sistema. En la actualidad, el activo más importante en una organización es la información.

Un sistema posee la propiedad de confidencialidad si la información manipulada por éste no es disponible ni puesta en descubierto para usuarios, entidades o procesos no autorizados. Posee la propiedad de integridad si los datos manipulados por éste no son alterados o destruidos por usuarios, entidades o procesos no autorizados. Y posee la propiedad de disponibilidad si la información es accesible (está disponible) en el momento en que así lo deseen los usuarios, entidades o procesos autorizados. El objetivo es minimizar los riesgos potenciales de seguridad.

- Análisis de riesgos
  - Análisis amenazas potenciales que se pueden sufrir,
  - Las pérdidas que se pueden generar,
  - Y la probabilidad de su ocurrencia
- Diseño política de seguridad
  - Definir responsabilidades y reglas a seguir para evitar tales amenazas, ó
  - Minimizar sus efectos en caso de que se produzcan
- Implementación
  - Usar mecanismos de seguridad para implementar lo anterior

El plan estratégico es el proceso de identificación y evaluación del riesgo a sufrir un ataque y perder datos, tiempo y horas de trabajo, comparándolo con el costo que significaría la prevención de este suceso. Su análisis no sólo lleva a establecer un nivel adecuado de seguridad, sino que permite conocer mejor el sistema que se va a proteger.

La información del análisis de riesgo se obtiene en un análisis de riesgo:

- Determinación precisa de los recursos sensibles de la organización.
- Identificación de las amenazas del sistema.
- Identificación de las vulnerabilidades específicas del sistema.
- Identificación de posibles pérdidas.
- Identificación de la probabilidad de ocurrencia de una pérdida.
- Derivación de contramedidas efectivas.
- Identificación de herramientas de seguridad.
- Implementación de un sistema de seguridad eficiente en costes y tiempo.

Las políticas de seguridad bien estructuradas deben especificar las características de seguridad que un sistema debe observar y proveer mediante un conjunto de reglas que deben respetarse para mantener la seguridad de la información. Especifican las amenazas contra las que la organización debe protegerse y cómo debe protegerse. Dependen de los objetivos y metas de la organización. Generalmente son expresadas en un lenguaje o idioma, así como establecidas en términos de sujetos y objetos.

Mecanismos de seguridad

Son la parte más visible de un sistema de seguridad. Se convierten en la herramienta básica para garantizar la protección de los sistemas o de la propia red. Se dividen en:

a) Detección

b) Prevención
c) Recuperación

La prevención incluye funciones de seguridad en *hardware* y *software*. Debe incluir la definición y observancia de políticas de seguridad. Incluye controles administrativos. Es la estrategia más ampliamente usada.

Mecanismos Prevención

Aumentan la seguridad de un sistema durante el funcionamiento normal de éste. Previenen la ocurrencia de violaciones a la seguridad. Algunos ejemplos de mecanismos son:

- Encriptación durante la transmisión de datos
- *Passwords* difíciles
- *Firewalls*
- Biométricos

Los mecanismos de prevención más habituales en la actualidad son:

Mecanismos de autenticación: Hacen posible identificar entidades del sistema de una forma única. Posteriormente, una vez identificadas, son autenticadas (comprobar que la entidad es quién dice ser)

- Basados en algo que se sabe:
  - Los primeros sistemas de autenticación se basaron en claves de acceso: nombre usuario y una clave de acceso. Son fáciles de usar y no requieren de un hardware especial. Siguen siendo el sistema de autenticación más usado hoy en día (*Passwords*, frases y números de identificación personal, NIPs).
- Basados en algo que se es:
  - Biométricas y comportamiento: Se realiza una medición física y se compara con un perfil almacenado con anterioridad,
- Basadas en algo que se tiene
  - Usar un objeto físico que llevan consigo y que de alguna forma comprueba la identidad del portador (*tokens*, tarjetas inteligentes y pases).

Mecanismos de control de acceso: Usados para proteger objetos del sistema (archivos, recursos, etc.). Controlan todos los tipos de acceso sobre el objeto por parte de cualquier entidad del sistema. La autenticación pretende establecer quién eres. La autorización (o control de accesos) establece qué puedes hacer con el sistema. Existen dos modelos: discrecional (DAC) y obligatorio (MAC)

- Control de acceso discrecional (DAC),
  - Un usuario bien identificado (típicamente, el creador o 'propietario' del recurso) decide cómo protegerlo

estableciendo cómo compartirlo, mediante controles de acceso impuestos por el sistema.

- Control acceso obligatorio (MAC)
  - Es el sistema quién protege los recursos. Todo recurso del sistema, y todo usuario tiene una etiqueta de seguridad.

Mecanismos de separación: Permiten separar los objetos dentro de cada nivel para evitar el flujo de información entre objetos y entidades de diferentes niveles siempre que no exista una autorización expresa del mecanismo de control de acceso.

- Definición de un perímetro de seguridad
- Definir las zonas "abiertas" y las zonas cerradas.
- DMZ: Zona desmilitarizada

Mecanismos de seguridad en las comunicaciones: Protegen la integridad y privacidad de los datos cuando se transmiten a través de la red. La mayor parte se basan en la criptografía. Utilizan protocolos seguros.

- Filtros de paquetes
- Ruteadores
- Switch
- Proxies

Mecanismos de detección: Busca descubrir incidentes al momento en que ocurren o lo antes posible. Debe permitir detectar eventos para reducir el daño. Permite identificar y perseguir culpables. También revelan vulnerabilidades. Se utilizan para detectar violaciones de la seguridad o intentos de violación.

- Detectores de violaciones
  - IDS
  - Tripwire
  - Proxies
- Detectores de vulnerabilidades
  - Nessus
  - Nikto
- Pen-test
  - CORE IMPACT
  - CANVAS
  - TWISTER

Mecanismos de recuperación: Se pone en marcha después de un incidente de seguridad. Incluyen la restauración de los recursos dañados. Debe remediar las vulnerabilidades que permitieron el incidente. Se aplican cuando una violación del sistema se ha detectado, para retornar a éste su funcionamiento correcto.

- Respaldos
  - Son una copia de los datos escrita en cinta u otro medio de almacenamiento duradero. De manera rutinaria se recuerda a los usuarios de computadoras que respalden su trabajo con frecuencia. Los administradores de sitios pueden tener la responsabilidad de respaldar docenas o incluso cientos de máquinas
- Redundancia
- Bitácoras
  - Se refiere al procedimiento a través del cual un sistema operativo registra eventos conforme van ocurriendo y los preserva para un uso posterior. Es posible configurar los sistemas de tal forma que los eventos se escriban en uno o en distintos archivos, se envíen a través de la red a otra computadora o se transmitan a algún dispositivo.
- BCP
- DRP

Planes de contingencia

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

Las empresas dependen hoy en día de los equipos informáticos y de todos los datos que hay allí almacenados (nóminas, clientes, facturas, etc.). Dependen también cada vez más de las comunicaciones a través de redes de datos. Si falla el sistema informático y éste no puede recuperarse, la empresa puede desaparecer porque no tiene tiempo de salir nuevamente al mercado con ciertas expectativas de éxito, aunque conserve a todo su personal.

Las pérdidas pueden ser diversas:

- Perdidas por no contar un plan
- Pérdida de clientes.
- Pérdida de imagen.
- Pérdida de ingresos por beneficios.
- Pérdida de ingresos por ventas y cobros.
- Pérdida de ingresos por producción.
- Pérdida de competitividad.
- Pérdida de credibilidad en el sector.

Servicios propuestos por OSI

Norma 7498-2:

a) Autenticación. La autentificación se refiere a demostrar la identidad de las entidades involucradas en la transacción. Evita que alguien tome la identidad de otro. Generalmente toma dos formas:
   i. Autentificación del proveedor de bienes o servicios.
   ii. Autentificación del cliente.

b) Control de acceso. Permite definir quién puede tener acceso a ciertos recursos, dependiendo de los privilegios o atributos que posea. Permite proteger los recursos del sistema contra el uso no autorizado. Se basa en credenciales o atributos. Se aplica a los usuarios y procesos que ya han sido autentificados.

c) Confidencialidad. Servicio que garantiza la privacidad de los datos:
   i. En local.
   ii. En conexiones.
   iii. En modo no conectado.
   iv. En campos selectos.
   v. En flujo de datos.

El principal mecanismo para implementar este servicio es la criptología.

d) Integridad de datos. Permite proteger los datos contra ataques activos.
   i. Con recuperación de datos.
   ii. Sin recuperación de datos.
   iii. En campos selectos. Los mecanismos principales son:
      → CRCs
      → Huellas digitales.

e) No repudiación. Permite comprobar las acciones realizadas por el origen o destino de los datos, con prueba de origen y de entrega. Los mecanismos principales son los certificados, la notarización y las firmas digitales.
   i. Es necesario garantizar que alguien que haya recibido un pago no pueda negar este hecho.
   ii. Es necesario garantizar que alguien que haya efectuado un pago no pueda negar haberlo hecho.

Existen otros modelos de seguridad, como por ejemplo:

- Modelo de Bell LaPadula (BLP)
  - Rígido. Confidencialidad y con autoridad.
- Modelo de Take-Grant (TG)
  - Derechos especiales: tomar y otorgar.
- Modelo de Clark-Wilson (CW)
  - Orientación comercial: integridad.

- Modelo de Goguen-Meseguer (GM)
  - No interferencia entre usuarios.
- Modelo de Matriz de Accesos (MA)
  - Estados y transiciones entre estados
- Tipo Graham-Dennig (GD)
- Tipo Harrison-Ruzzo-Ullman (HRU)

Las amenazas que mas abundan en las redes son:

- *Script Kiddies* (gente con la capacidad de buscar un programa en la red y ejecutarlo).
- Gente con necesidad de pertenencia al llamado *underground*, aunque sean de grupos *lammers*
- Cuando no hay preocupación por las consecuencias reales de sus actos.
- Grafiteros por excelencia.
- Generalmente utilizan tipografía "*leet*" o "7337" (3ST0 S3RI4 UN 3J3MPLO D3 DICH0 L3NGU4J3)

Criptología

Ciencia que estudia los aspectos y contenidos de información en condiciones de secrecía. Del griego: *criptos* oculto y *logos* tratado. Es el arte de construir códigos secretos. La criptología se divide en:

Criptografía.
Es el conjunto de técnicas o procedimientos que alteran los símbolos de información sin alterar el contenido, convirtiendo a la información modificada en un conjunto de símbolos sin contenido para las partes que no disponen de las técnicas.

Criptoanálisis.
Criptoanálisis: Metodologías y técnicas que permiten recuperar la información que ha sido previamente tratada por un procedimiento criptográfico, sin conocer *a priori* la técnica utilizada para la criptografía.

Cifrado
Se define como cifrado a un algoritmo criptográfico empleado para ocultar el mensaje en texto plano. Es la función matemática usada para la encriptación y decodificación del mensaje original

Criptosistemas
Es el conjunto de procedimiento que garantizan la seguridad de la información y utilizan técnicas criptográficas. El elemento fundamental de un Criptosistema es la *"llave".* En algunas referencias a la llave se le conoce como *clave.*

Objetivos de la criptografía

- Mantener la confidencialidad del mensaje
- La información contenida en el mensaje permanezca secreta
- Garantizar la autenticidad tanto del mensaje como del par remitente/destinatario
- El mensaje recibido ha de ser realmente el enviado
- El remitente y destinatario han de ser realmente quienes dicen ser y no remitentes y/o destinatarios fraudulentos

Métodos de Cifrado

Métodos Simétricos: La llave de encriptado coincide con la de descifrado, tiene que permanecer secreta. Emisor y receptor se han puesto de acuerdo previamente o existe un centro de distribución de llaves. Los métodos simétricos son propios de la criptografía clásica o criptografía de llave secreta.

Métodos Asimétricos: La llave de encriptado es conocida por el público y es diferente a la de descriptazo la cual es conocida sólo por el usuario. Los métodos asimétricos corresponden a la criptografía de la llave pública, introducida por Diffie y Hellman en 1976.

Clasificación de los métodos simétricos

Encriptación en flujo.
Usa la llave como semilla de un generador de números pseudo-aleatorio. El resultado del generador es una secuencia de bits. La secuencia se suma módulo 2 con el texto claro (emisión) o con el criptograma (recepción).

Encriptación en bloques.
Se cifra el mensaje original agrupando los símbolos en grupos (bloques) de dos o más elementos. Modos operación de encriptación en bloque:

ECB: Electronic Code Book
CBC: Cipher Block Chaining

Llave Pública

El concepto de llave pública fue inventado en 1976 por Whitfield Diffie y Martin Hellman e independientemente por Ralph Merkle. Más adelante se define que las llaves pueden presentarse en pares. Desde 1976 varios algoritmos han sido propuestos, muchos de estos son considerados seguros, pero son imprácticos. Algunos sólo son buenos para distribución de llaves. Otros sólo son buenos para encriptación. Algunos más sólo son buenos para firmas digitales. Sólo tres algoritmos son buenos para encriptación y firmas digitales:

- RSA
- ElGamal
- Rabin.

Los tres algoritmos son más lentos que los algoritmos simétricos.

Llave privada: Se genera en el sistema cuando arranca por primera vez.
Llave pública: Se genera en cada usuario para autentificar la conexión.

Funciones de autenticación

La función *hash* MD5

MD5 toma como entrada un mensaje de longitud arbitraria y regresa como salida una huella digital de 128 bits del mensaje (llamado *message-digest* o compendio del mensaje). Se estima que es imposible obtener dos mensajes que produzcan la misma huella digital. También es imposible producir un mensaje que arroje una huella predefinida

Huella digital

La salida producida por una función *hash* aplicada a un documento, es conocida con el nombre de huella digital de dicho documento. Cualquier cambio en el documento produce una huella diferente. También es conocida como compendio de mensaje (cuando el documento es un mensaje).

Los certificados digitales

Es un documento que asienta que una llave pública y su correspondiente llave privada pertenecen a un individuo en particular, certificando de esta manera la identidad de dicho individuo. Tiene el propósito de hacer disponible a otras personas una llave pública personal.

Los certificados son expedidos por autoridades confiables conocidas como Autoridades Certificadoras, (CA por sus siglas en inglés). Estas entidades son responsables de certificar la identidad de un individuo y su posesión de una llave pública. Generan y administran certificados y los publican en repositorios. Partes importantes de un certificado:
Nombre del usuario y otra información adicional, tal como su e-mail.
Llave pública del usuario.
Nombre del emisor (CA).
Número serial.
Periodo de validez.
Firma que enlaza todas las partes del certificado.

Políticas de cuentas y contraseñas

- La política define la seguridad de un sistema de cómputo.

- Un sistema es seguro si vive dentro sus políticas de seguridad.
- La política especifica qué propiedades de seguridad el sistema debe proporcionar.
- La política define la seguridad de cómputo para una organización, especificando:
- Propiedades del sistema
- Responsabilidades de seguridad de la gente

Monitoreo del sistema

Siempre se debe monitorear los procesos que están ejecutándose de preferencia con *scripts* externos. Se deben evitar accesos no autorizados al sistema por medio de detectores de intrusos a nivel *host* como TRIPWIRE, así como deshabilitar los servicios que no se estén usando. La sintaxis de algunos comandos para el monitoreo básico es:

netstat –an	Muestra las conexiones punto a punto y los servicios abiertos.
ps –x	Muestra los procesos que están corriendo en el sistema.
top	Nos muestra el uso del cpu, y que procesos están consumiendo mas recursos.
md5sum	Permite comprobar la integridad de archivos y saber si fueron o no alterados.

Sniffers

Un *sniffer* es un proceso que "olfatea" el tráfico que se genera en la red a nivel de enlace. Puede leer toda la información que circule por el tramo de red en el que se encuentre. Se pueden capturar claves de acceso, datos que se transmiten, números de secuencia, etc.

Un analizador de protocolos es un *sniffer* al que se le ha añadido funcionalidad suficiente como para entender y traducir los protocolos que se están hablando en la red. Debe tener suficiente funcionalidad como para entender las tramas de nivel de enlace, y los paquetes que transporten. Normalmente la diferencia entre un *sniffer* y un analizador de protocolos, es que el segundo está a la venta en las tiendas y no muestra claves de acceso.

El sniffer se dedica a leer TRAMAS de red. Los datos que se obtendrán de él serán tramas que transportarán paquetes (IP, IPX, etc...). En estos paquetes se incluyen los datos de capas superiores entre ellos los de la capa de aplicación (posiblemente claves de acceso). Hay dos tipos de *Sniffers:* pasivos que no realizan actividad alguna sólo capturan paquetes y activos que intentan apoderarse de las sesiones mediante algunas técnicas como el *spoofing* (envenenamiento de la tabla de *arp*).

Spoofing

Spoofing es la creación de paquetes de comunicación TCP/IP usando una dirección IP de alguien más. Lo anterior permite entrar en un sistema haciéndose pasar por un usuario autorizado. Una vez dentro del sistema, el atacante puede servirse de éste como plataforma para introducirse en otro y así sucesivamente.

Virus

Un virus se define como una porción de código de programación cuyo objetivo es implementarse a si mismo en un archivo ejecutable y multiplicarse sistemáticamente de un archivo a otro. Además de esta función primaria de "invasión" o "reproducción", los virus están diseñados para realizar una acción concreta en los sistemas informáticos. En ocasiones de habla de estas variantes como si de virus se tratara, cuando en realidad son conceptualmente diferentes. Algunos antivirus pueden detectarlos. Estas variantes son:

- Troyanos
- Gusanos
- Bomba lógica

Secure Socket Layer

Es una propuesta de estándar para encriptado y autenticación en el Web. Fue diseñado en 1993 por Netscape. Es un esquema de encriptado de bajo nivel usado para encriptar transacciones en protocolos de nivel aplicación como HTTP, FTP, etc.

Con SSL puede autentificarse un servidor con respecto a su cliente y viceversa. Se basa en un esquema de llave pública para el intercambio de llaves de sesión. Las llaves de sesión son usadas para encriptar las transacciones sobre HTTP. Cada transacción usa una llave de sesión, esto dificulta al "atacante" el comprometer toda una sesión.

Políticas de Seguridad

A las *reglas* en la que se incluyan las propiedades de confidencialidad, integridad *y* disponibilidad, en la medida requerida por una organización, se les conoce como Política de Seguridad. Por lo anterior, las políticas deben ser sucintas, claras y entendibles, prácticas, cooperativas y dinámicas.

La correcta implementación de estas reglas, mediante servicios o mecanismos de seguridad, garantizará la seguridad del sistema.

Si algún usuario incurre en dar mal uso a su cuenta, y por ende a los recursos que están asociados a ella, podrá en un momento dado ser sancionado por el Consejo de Seguridad de su organización.

I.8 **Programación con JAVA**

OBJETIVO ESPECÍFICO:
El participante identificará los principales componentes del lenguaje Java, que le permitan ir incorporando los conocimientos adquiridos sobre el lenguaje, durante e curso.

TEMARIO:
1. La tecnología Java
2. Estructura del lenguaje
3. Conceptos de la programación orientada a objetos
4. Trabajando con objetos en Java
5. Excepciones
6. AWT creación de APPLETS y aplicaciones

Java es un lenguaje sencillo y orientado a objetos que permite desarrollar aplicaciones en muy diversas áreas tales como: seguridad animación, acceso a bases de datos, aplicaciones cliente-servidor, interfaces gráficas, páginas de Web interactivas, entre otras. Desarrollado en la empresa *Sun Microsystems* a principios de la década del 90. El lenguaje, que fue diseñado para ser independiente de la plataforma, es una derivación del C++ con una sintaxis más simple, un entorno de tiempo de ejecución más robusto y un manejo de la memoria simplificado. Java no está relacionado con *JavaScript*, aunque tengan nombres similares y compartan una sintaxis parecida al C.

Una de sus principales características es la creación de módulos reutilizables que funcionan sin necesidad de conocer su estructura interna, permitiendo al usuario añadir nuevos módulos. Así como la obtención de programas independientes de la plataforma en la cual fueron desarrollados.

La programación en Java, permite el desarrollo de aplicaciones bajo el esquema de Cliente - Servidor, como de aplicaciones distribuidas, lo que lo hace capaz de conectar dos o más computadoras u ordenadores, ejecutando tareas simultáneamente, y de esta forma logra distribuir el trabajo a realizar.

La *plataforma Java* consta de las siguientes partes:
- El lenguaje de programación, mismo.
- La máquina virtual de Java o JRE, que permite la portabilidad en ejecución.
- El API Java, una biblioteca estándar para el lenguaje.

Historia

Originalmente llamado OAK por los ingenieros de *First Person Inc*. (filial creada por *Sun Microsystems* para llevar a cabo el desarrollo del lenguaje que inicialmente sería utilizado para máquinas domésticas), Java fue diseñado para correr en computadoras incrustadas. Sin embargo, en 1995, dada la atención que estaba produciendo la web, *Sun Microsystems* la distribuyó para sistemas operativos tales como Microsoft Windows.
El lenguaje mismo se inspira en la sintaxis de C++, pero su funcionamiento es más similar al de *Smalltalk* que a éste. Incorpora sincronización y manejo de tareas en el lenguaje mismo (similar a *Ada*) e incorpora interfaces como un mecanismo alternativo a la herencia múltiple de C++.
A fines del siglo XX, Java llegó a ser el lenguaje de mayor acogida para programas de servidor. Utilizando una tecnología llamada JSP (similar a otras tecnologías del lado del servidor como ASP de Microsoft o PHP), se hizo muy fácil escribir páginas dinámicas para sitios de Internet. Sumado a JSP la tecnología de *JavaBeans*, permitía adaptar al mundo web el patrón MVC (modelo-vista-controlador) que ya se había aplicado con éxito a interfaces gráficas.

Java llegó a ser extremadamente popular cuando *Sun Microsystems* introdujo la especificación J2EE (Java 2 *Enterprise Edition*). Este modelo permite, entre otras cosas, lograr una separación entre la presentación de los datos al usuario (JSP o Applets), el modelo de datos (EJB), y el control (Servlets). *Enterprise Java Beans* (EJB) es una tecnología de objetos distribuidos que pudo lograr el sueño de muchas empresas como Microsoft e IBM de crear una plataforma de objetos distribuidos con un monitor de transacciones. Con este nuevo estándar, empresas como BEA, IBM, *Sun Microsystems*, Oracle y otros crearon nuevos "servidores de aplicaciones" que tuvieron gran acogida en el mercado.

Además de programas del servidor, Java permite escribir programas de interfaz gráfica o textual. También se pueden correr programas de manera incorporada o incrustada en los navegadores web de Internet en forma de Java *applets*, aunque no llegó a popularizarse como se esperaba en un principio.

Los programas en Java generalmente son compilados a un lenguaje intermedio llamado *bytecode*, que luego son interpretados por una máquina virtual (JVM). Esta última sirve como una plataforma de abstracción entre la máquina y el lenguaje permitiendo que se pueda "escribir el programa una vez, y correrlo en cualquier lado". También existen compiladores nativos de Java, tanto software libre como no libre. El compilador GCC de GNU compila Java a código de máquina con algunas limitaciones al año 2002.

Con la evolución de las diferentes versiones, no sólo se han producido cambios en el lenguaje, sino que se han producido cambios mucho más importantes en sus bibliotecas asociadas, que han pasado de unos pocos cientos en Java 1.0, a más de tres mil en Java 5.0. En particular, se han añadido APIs completamente nuevas, tales como Swing y Java2D.

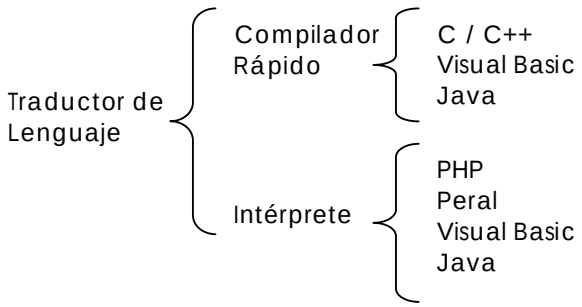
IDE (*Integrated Development Environment)*

Un entorno (o ambiente) integrado de desarrollo o por sus siglas en inglés, IDE, es un programa compuesto por un conjunto de herramientas para un programador. Puede dedicarse en exclusiva a un sólo lenguaje de programación o bien, poder utilizarse para varios. Los principales IDEs utilizados para Java son:

- Eclipse - Software libre
- NetBeans - Software libre
- IntelliJ IDEA - Software no libre
- Rational Application Developer (antes WebSphere) - Software No libre
- BlueJ - Software no libre, gratuito
- jCreator - Software no libre, versión gratuita disponible.
- Sun Java Studio - Software no libre
- JBuilder - Software no libre
- JDeveloper - Software no libre

Conceptos básicos

Programa	Conjunto de instrucciones que realiza una tarea.
Clase	Es un molde que sirve para definir los atributos y el comportamiento de los objetos.
Objeto	Instancia de una clase.



Ventajas de utilizar Java

El lenguaje de programación Java ha sido totalmente mejorado, ampliado y probado por una comunidad activa de unos cuatro millones de desarrolladores de software.

La tecnología Java, una tecnología madura, extremadamente eficaz y sorprendentemente versátil, se ha convertido en un recurso inestimable ya que permite a los desarrolladores:

- Desarrollar software en una plataforma y ejecutarlo en prácticamente cualquier otra plataforma
- Crear programas para que funcionen en un navegador Web y en servicios Web
- Desarrollar aplicaciones para servidores como foros en línea, tiendas, encuestas, procesamiento de formularios HTML, etc.
- Combinar aplicaciones o servicios basados en la tecnología Java para crear servicios o aplicaciones totalmente personalizados
- Desarrollar potentes y eficientes aplicaciones para teléfonos móviles, procesadores remotos, productos de consumo de bajo coste y prácticamente cualquier dispositivo digital

Programando

Cuando se programa en Java, se coloca todo el código en métodos, de la misma forma que se escriben funciones en lenguajes como C.

Comentarios
En Java hay tres tipos de comentarios:

//	Comentarios para una sola línea
/* ... */	Comentarios de una o más líneas
/** ... */	Comentario de documentación, de una o más líneas

Los dos primeros tipos de comentarios son los que todo programador conoce y se utilizan del mismo modo. Los comentarios de documentación, colocados inmediatamente antes de una declaración (de variable o función), indican que ese comentario ha de ser colocado en la documentación que se genera automáticamente cuando se utiliza la herramienta de Java, *javadoc*. Dichos comentarios sirven como descripción del elemento declarado permitiendo generar una documentación de nuestras clases escrita al mismo tiempo que se genera el código.

En este tipo de comentario para documentación, se permite la introducción de algunos *tokens* o palabras clave, que harán que la información que les sigue aparezca de forma diferente al resto en la documentación.

Identificadores
Los identificadores nombran variables, funciones, clases y objetos; cualquier cosa que el programador necesite identificar o usar.
En Java, un identificador comienza con una letra, un subrayado (_) o un símbolo de dólar ($). Los siguientes caracteres pueden ser letras o

dígitos. Se distinguen las mayúsculas de las minúsculas y no hay longitud máxima.

Palabras clave

Las siguientes son las palabras clave que están definidas en Java y que no se pueden utilizar como indentificadores:

abstract	*continue*	*for*	*new*	*switch*
boolean	*default*	*goto*	*null*	*synchronized*
break	*public*	*throw*	*this*	*package*
byte	*double*	*try*	*if*	*threadsafe*
byvalue	*import*	*do*	*else*	*protected*
case	*while*	*transient*	*private*	*true*
catch	*false*	*return*	*int*	*instanceof*
char	*final*	*void*	*implements*	*short*
class	*finally*	*long*	*static*	*interface*
const	*float*	*native*	*super*	*extends*

Palabras Reservadas

Además, el lenguaje se reserva unas cuantas palabras más, pero que hasta ahora no tienen un cometido específico. Son:

cast	future	generic	inner
operator	outer	rest	var

Literales

Un valor constante en Java se crea utilizando una representación literal de él. Java utiliza cinco tipos de elementos: enteros, reales en coma flotante, booleanos, caracteres y cadenas, que se pueden poner en cualquier lugar del código fuente de Java. Cada uno de estos literales tiene un tipo correspondiente asociado con él.

Enteros:

- byte
- short
- int
- long

Reales en coma flotante:

- float
- double

Booleanos:

- true
- false

Caracteres

Cadenas

Arreglos (*Arrays*)

Se pueden declarar en Java *arrays* de cualquier tipo. Incluso se pueden construir *arrays* de *arrays*. Los límites de los *arrays* se comprueban en

tiempo de ejecución para evitar desbordamientos y la corrupción de memoria. En Java un *array* es realmente un objeto, porque tiene redefinido el operador [ ]. Tiene una función miembro: *length*. Se puede utilizar este método para conocer la longitud de cualquier *array*.

Para crear un *array* en Java hay dos métodos básicos. Crear un *array* vacío o se puede crear ya el *array* con sus valores iniciales. No se pueden crear *arrays* estáticos en tiempo de compilación. Tampoco se puede rellenar un *array* sin declarar el tamaño con el operador *new*.

Es decir, todos los *arrays* en Java son estáticos. Para convertir un *array* en el equivalente a un *array* dinámico en C/C++, se usa la clase *vector*, que permite operaciones de inserción, borrado, etc. en el *array*.

Operadores

Los operadores de Java son muy parecidos en estilo y funcionamiento a los de C. Los operadores numéricos se comportan como se espera:

int + *int* = *int*

Los operadores relacionales devuelven un valor booleano. Para las cadenas, se pueden utilizar los operadores relacionales para comparaciones además de **+** y **+=** para la concatenación. El operador **=** siempre hace copias de objetos, marcando los antiguos para borrarlos, y ya se encargará el *garbage collector* de devolver al sistema la memoria ocupada por el objeto eliminado.

Separadores

Sólo hay un par de secuencias con otros caracteres que pueden aparecer en el código Java; son los separadores simples, que van a definir la forma y función del código. Los separadores admitidos en Java son:

( ) - *paréntesis*. Para contener listas de parámetros en la definición y llamada a métodos. También se utiliza para definir precedencia en expresiones, contener expresiones para control de flujo y rodear las conversiones de tipo.

{ } - *llaves*. Para contener los valores de matrices inicializadas automáticamente. También se utiliza para definir un bloque de código, para clases, métodos y ámbitos locales.

[ ] - *corchetes*. Para declarar tipos matriz. También se utiliza cuando se referencian valores de matriz.

; - *punto y coma*. Separa sentencias.

, - *coma*. Separa identificadores consecutivos en una declaración de variables. También se utiliza para encadenar sentencias dentro de una sentencia *for*.

. - *punto*. Para separar nombres de paquete de subpaquetes y clases. También se utiliza para separar una variable o método de una variable de referencia.

Clases

El elemento básico de la programación orientada a objetos en Java es una clase. Una clase define la forma y el comportamiento de un objeto. Cualquier concepto que desee representar en su programa en Java está encapsulado en una clase.

Las clases de Java típicas incluirán variables y métodos de instancia. Los programas en Java completos constarán por lo general de varias clases de Java de distintos archivos fuente. Una clase define la estructura de un objeto y su interfaz funcional, conocida como métodos. Cuando se ejecuta un programa en Java, el sistema utiliza definiciones de clase para crear instancias de las clases, que son objetos reales. La forma general de una clase se muestra a continuación.

```
class nombre_de_clase extends nombre_de_superclase {
        type variable_de_instanciaX;
        type variable_de_instanciaY;;
        type nombre_de_métodoX (lista_de_parámetros) {
                cuerpo_del_método;
        }
        type nombre_de_métodoY (lista_de_parámetros) {
                cuerpo_del_método;
        }
}
```

Aquí, *nombre_de_clase* y *nombre_de_superclase* son identificadores. La palabra clave *extends* se utiliza para indicar que *nombre_de_clase* será una subclase de *nombre_de_superclase*. Hay una clase incorporada, llamada *Object* (objeto), que está en la raíz de la jerarquía de clases de Java. Si se desea realizar una subclase de *Object* directamente, se puede omitir la cláusula *extends*.

Referencias a objeto

Cada nueva clase que se crea añade otro tipo que se puede utilizar igual que los tipos simples. Por lo tanto, cuando se declara una nueva variable, se puede utilizar un nombre de clase como tipo. A estas variables se las conoce como referencias a objeto.

A cada instancia se le puede llamar también objeto. Cuando se declara que el tipo de una variable es una clase, tiene como valor por omisión el *null*, que es una referencia al tipo *Object*, y, por lo tanto, es compatible en tipo con todas las otras clases. El objeto *null* no tiene valor; es distinto del entero 0, igual que el *false* de *boolean*. Este ejemplo declara una variable p cuyo tipo es de la clase *Point*.

Variables de instancia

Los datos se encapsulan dentro de una clase declarando las variables dentro de las llaves de apertura y cierre de la declaración de la clase. A las variables que se declaran en este ámbito y fuera del ámbito de un método concreto se las conoce como variables de instancia.

El operador *new*

El operador *new* crea una única instancia de una clase y devuelve una referencia a ese objeto.

El operador punto (.)

El operador punto se utiliza para acceder a las variables de instancia y los métodos contenidos en un objeto. Esta es la forma general de acceder a las variables de instancia utilizando el operador punto.

```
referencia_a_objeto.nombre_de_variable
```

Aquí *referencia_a_objeto* es una referencia a un objeto y *nombre_de_variable* es el nombre de la variable de instancia contenida en el objeto al que se desea acceder.

Declaración de método

Los métodos son subrutinas unidas a una definición de una clase específica. Se declaran dentro de una definición de clase al mismo nivel que las variables de instancia. Se debe llamar a los métodos en el contexto de una instancia concreta de esa clase. En la declaración de los métodos se define que devuelve un valor de un tipo concreto y que tiene un conjunto de parámetros de entrada.

```
tipo nombre_de_método ( lista_formal_de_parámetros ) {
        cuerpo_del_método;
}
```

El *nombre_de_método* es cualquier identificador legal distinto de los ya utilizados en el ámbito actual. La *lista_forma_de_parámetros* es una secuencia de parejas de tipo e identificador separadas por comas. Si no se desean parámetros, la declaración del método deberá incluir un par de paréntesis vacío.

this

En Java es ilegal declarar dos variables locales con el mismo nombre dentro del mismo ámbito o uno que lo incluya. Se observará que se ha utilizado *x* e *y* como parámetros para el método *init* y en el interior del método se ha utilizado un valor de referencia especial llamado *this* para referirse directamente a las variables de instancia. Si no se hubiera utilizado *this* entonces *x* e *y* se hubieran referido al parámetro formal y

no a las variables de instancia lo que se conoce como ocultar variables de instancia**.**

El método *main( )*
Dado que no hay funciones globales en Java, se debía idear alguna manera de iniciar un programa, de ahí el sentido del método *main*. Puesto que en otros lenguajes (p.e. C y C++) *main* se utilizaba a menudo para pasar parámetros desde la línea de órdenes, un concepto perdido en los usuarios de interfaces de usuario gráficas, el *main* de Java también pasa esos argumentos.

El compilador de Java compilará clases que no tengan el método *main*. El intérprete Java, sin embargo, no tiene ningún modo de ejecutar esas clases. El método *main* es simplemente un lugar de inicio para que el intérprete comience. Un programa complejo tendrá docenas de clases, y sólo una de ellas necesitará tener un método *main*. Para los *applets* (programas Java que están incrustados en los visualizadores de red) no se utiliza el método *main*, ya que los visualizadores (o navegadores) de red siguen un convenio distinto para inicializar *applets*.

Llamada a método
Se llama a los métodos dentro de una instancia de un clase utilizando el operador punto (.). La forma general de una llamada:
referencia_a_objeto . nombre_de_método (lista_de_parámetros);
Aquí, *referencia_a_objeto* es cualquier variable que se refiere a un objeto, *nombre_de_método* es el nombre de un método de la clase con la que se declaró *referencia_a_objeto* y *lista_de_parámetros* es una lista de valores o expresiones separados por comas que coinciden en número y tipo con cualquiera de los métodos declarados como *nombre_de_método* en la clase.

Constructores
Las clases pueden implementar un método especial llamado constructor. Un constructor es un método que inicializa un objeto inmediatamente después de su creación. Tienen exactamente el mismo nombre de la clase en la que residen; de hecho no se puede tener ningún otro método que comparta su nombre con su clase. Una vez definido, se llama automáticamente al constructor después de crear el objeto, antes de que termine el operador *new*.

Sobrecarga de método

Es posible y a menudo deseable crear más de un método con el mismo nombre, pero con listas de parámetros distintas. A esto se le llama sobrecarga de método. Se sobrecarga un método siempre que se crea un método en una clase que ya tiene un método con el mismo nombre.

static

A veces se desea crear un método que se utiliza fuera del contexto de cualquier instancia. Todo lo que se tiene que hacer es declarar estos métodos como *static* (estático). Los métodos estáticos sólo pueden llamar a otros métodos *static* directamente, y no se pueden referir a *this* o *super* de ninguna manera. Las variables también se pueden declarar como *static*, pero debe ser consciente que es equivalente a declararlas como variables globales, que son accesibles desde cualquier fragmento de código. Se puede declarar un bloque *static* que se ejecuta una sola vez si se necesitan realizar cálculos para inicializar las variables *static*.

abstract

Hay situaciones que se necesita definir una clase que declara la estructura de una abstracción dada sin promocionar una implementación completa de cada método. Se puede indicar que se necesita que ciertos métodos se sobrescriban en subclases utilizando el modificador *abstract*. A estos métodos se les llama a veces responsabilidad de subclase. Cualquier clase que contenga métodos declarados como *abstract* también se tiene que declarar como *abstract*. No se pueden crear instancias de dichas clases directamente con el operador *new*, dado que su implementación completa no está definida. No se pueden declarar constructores *abstract* o métodos *abstract static*. Cualquier subclase de una clase *abstract* debe implementar todos los métodos *abstract* de la superclase o ser declarada también como *abstract*.

Control de flujo

El control del flujo es la manera que tiene un lenguaje de programación de provocar que el flujo de la ejecución avance y se ramifique en función de los cambios de estado de los datos. La ramificación, iteración, selección y llamadas a subrutina son formas de control de flujo.

Ramificación

En los programas se necesitará que algunas sentencias se ejecuten condicionalmente y se omitan otras. Java proporciona varios mecanismos para conseguir este control y decidir qué partes del código ejecutar, mediante sus sentencias de ramificación.

if-else

La construcción *if-else* provoca que la ejecución atraviese un conjunto de estados *boolean* que determinan que se ejecuten distintos fragmentos de código. Su sintaxis es:

if (expresión-booleana) sentencia1; [else sentencia2;]

La cláusula *else* es opcional. Cada una de las sentencias puede ser una sentencia compuesta encerrada entre llaves, { }. Una expresión-booleana es cualquier expresión que devuelve un valor *boolean*. Podría ser una variable simple declarada como *boolean*.

break

La sentencia *break* de Java está diseñada para cubrir aquellos casos en los que saltar arbitrariamente a una porción de código es una construcción valiosa y legítima para el control del flujo. El término *break* se refiere al acto de salirse de un bloque de código. Le dice al intérprete que retome la ejecución pasado el final del bloque.

switch

La sentencia *switch* proporciona una forma limpia de dirigir la ejecución a partes diferentes del código en base al valor de una variable o expresión. Esta es la forma general de la sentencia *switch*:

```
switch ( expresión ) {
        case valorX:
                break;
        case valorY:
                break;
        default:
}
```

El valor de expresión se compara con cada uno de los valores literales de las sentencias *case*. Si coincide con alguno, se ejecuta el código que sigue a la sentencia *case*. Si no coincide con ninguno de ellos, entonces se ejecuta la sentencia *default* (por defecto). La sentencia *default* es opcional. La sentencia *break*, comentada anteriormente, hace, en este caso, que la ejecución salte al final del *switch*. Si no se pone el *break*, la ejecución continuará en el siguiente *case*.

return

Como se explicará en el siguiente capítulo, Java utiliza una forma de subrutina llamada método para implementar una interfaz de procedimiento a las clases de objetos. En cualquier momento dentro de un método, se puede utilizar la sentencia *return* para que la ejecución salte y vuelva al punto donde se llamó al método.

Bucles

Se llama bucle a ejecutar repetidamente el mismo bloque de código hasta que cumpla una condición de terminación. Hay cuatro partes en cualquier bucle, inicialización, cuerpo, iteración y terminación. La inicialización es el código que establece las condiciones iniciales de un

bucle. El cuerpo es la sentencia que se quiere repetir. La iteración es el código que se quiere ejecutar después de cuerpo, pero antes de entrar de nuevo en el bucle. Se utiliza a menudo para incrementar o decrementar contadores e índices. La terminación es la expresión booleana que comprueba cada vez a lo largo de un bucle para ver si ha llegado el momento de parar de iterar. Java tiene tres construcciones para bucles: *while, do-while* y *for*.

while
Ejecuta una sentencia repetidamente mientras una expresión booleana sea verdadera. Esta es su forma general:

```
[ inicialización; ]
while ( terminación ) {
        cuerpo;
        [ iteración; ]
}
```

Las partes inicialización e iteración son opcionales, y mientras la expresión terminación devuelva un valor true, la sentencia cuerpo continuará ejecutándose.

do-while
La construcción *do-while* se utiliza cuando se desea ejecutar el cuerpo de un bucle *while* al menos una vez, incluso si la expresión booleana tiene el valor *false* la primera vez. Es decir si se desea evaluar la expresión de terminación al final del bucle en vez de al principio como en el *while*.

```
[ inicialización; ]
do { cuerpo; [ iteración; ] } while ( terminación );
```

for
La sentencia *for* es una forma compacta de expresar un bucle.

```
for ( inicalización; terminación; iteración ) cuerpo;
```

Si las condiciones iniciales no provocan que la terminación devuelva *true* la primera vez, entonces la sentencia cuerpo e iteración no se ejecutarán nunca. A veces se podría desear incluir más de un sentencia en las sentencias de inicialización y de iteración. Java proporciona una manera de expresar sentencias múltiples mediante la utilización de comas (,) en el interior de los paréntesis de un bucle *for*.

Por lo tanto puede ponerse tantas sentencias como se desee, siempre y cuando estén separadas por comas.

continue

Del mismo modo que se desea salir prematuramente de un bucle, se podría desear continuar en el bucle, pero dejar de procesar el resto de código en esta iteración en concreto. La sentencia *continue* de Java salta del cuerpo de bucle, pero permaneciendo en el bucle.

Excepciones

La última manera de provocar un salto en el flujo de un código es utilizando el mecanismo de gestión de excepciones de Java. Las sentencias *try*, *catch*, *throw* y *finally* se utilizan para expresar este modelo potente para ramificación no local. La gestión de excepciones se utiliza muy a menudo en las clases de Java fundamentales.

Una excepción es una condición anormal que surge en una secuencia de código durante la ejecución. La gestión de excepciones de Java lleva la gestión del error en tiempo de ejecución al mundo orientado a objetos. Una excepción de Java es un objeto que describe una condición excepcional que se ha producido en un fragmento de código.

Excepciones no capturadas

Los objetos de excepción los crea automáticamente el intérprete de Java como respuesta a alguna condición excepcional. Como ejemplo se tiene una división entre cero. Cuando el intérprete de Java intenta ejecutar la división, observa que el denominador es cero y construye un nuevo objeto de excepción para que se detenga este código y se trate esta condición de error. Una vez detenido el flujo del código en el operador de división, se buscará en la pila de llamadas actual cualquier gestor de excepciones (pila que contiene un registro de las llamadas a método).

Un gestor de excepciones es algo establecido para tratar inmediatamente la condición excepcional. Si no se codifica un gestor de excepciones, se ejecutara el gestor en tiempo de ejecución por defecto. El gestor por defecto imprime el valor *String* de la excepción y el trazado de la pila del lugar donde se produjo la excepción.

try y *catch*

A menudo es más elegante y practico manejar uno mismo la excepción. Se puede utilizar la palabra clave *try* para especificar un bloque de código que se debería proteger frente a todas las excepciones. A continuación inmediatamente del bloque *try*, se incluye la cláusula *catch* que especifica el tipo de excepción que se desea captar.

Cláusulas *catch* múltiples

En algunos casos, la misma secuencia de código puede activar más de una condición excepcional. Se pueden tener varias cláusulas *catch* en una fila. Se inspecciona cada uno de estos tipos de excepción en el orden en que están y el primero que coincida se ejecuta. Las clases de excepción más específicas se colocaran primero, dado que no se alcanzarán las subclases si están después de unas superclase.

throw
La sentencia *throw* se utiliza para lanzar explícitamente una excepción. En primer lugar, debe obtener un descriptor de una instancia de *Throwable*, mediante un parámetro en una cláusula *catch*, o cerrar una utilizando el operador *new*. El flujo de la ejecución se detiene inmediatamente después de la sentencia *throw*, y no se llega a la sentencia siguiente. Se inspecciona el bloque *try* que la engloba más cercano para ver si tiene una cláusula *catch* cuyo tipo coincida con el de la instancia *Throwable*. Si la encuentra, el control se transfiere a esa sentencia. Si no, se inspecciona el siguiente bloque *try* que la engloba, y así sucesivamente, hasta que le gestor de excepción más externo detiene el programa e imprime el trazado de la pila hasta la sentencia *throw*.

finally
A veces es necesario estar seguro de que se ejecutará un fragmento de código dado independientemente de que excepciones se provocan y capturan. Se puede utilizar la palabra clave *finally* par identificar dicho bloque de código. Incluso aunque so coincida ninguna de las cláusulas *catch*, se ejecutará el bloque *finally* antes del código que está después del final del bloque *try* completo.

Interfaces

Una interfaz es una lista de acciones que puede llevar a cabo un determinado objeto. En una interfaz sólo existe el prototipo de una función, no su código. En java un interfaz define la lista de métodos, pero para que una clase posea un interfaz hay que indicar explícitamente que lo implementa mediante la cláusula *implements*. La estructura de un interfaz es:

```
[modif.visibilidad] interface nombreInterfaz [extends listaInterfaces]
{
prototipo método1;
.....
prototipo método1;
}
```

Donde *modif.visibilidad* puede ser *public* o bien sin especificar, es decir visibilidad pública (desde cualquier clase se puede emplear el interfaz) o de paquete (sólo se puede emplear desde clases del mismo paquete); *nombreInterfaz* por convenio, sigue las mismas reglas de nomenclatura que las clases.

La cláusula opcional *extends*, se emplea para conseguir que un interfaz herede las funciones de otro/s interfaces, simplemente *listaInterfaces* es una lista separada por coma de interfaces de los que se desea heredar.

En muchas ocasiones un interfaz es empleado para definir un comportamiento, que posteriormente será implementado por diversas clases, que podrán no tener nada que ver entre ellas, pero que todas se comportarán igual de cara al interfaz. Es decir, todas tendrán las funciones indicadas por el interfaz.

Cuando varios objetos de distintas clases pueden responder al mismo mensaje (función), aún realizando cosas distintas se denomina **polimorfismo**.

II. PROYECTO PARA LA ELABORACION DE UNA INTERFAZ WEB DE REGISTRO DE PERSONAL UTILIZANDO SOFTWARE LIBRE

OBJETIVO ESPECÍFICO:

Implementar una interfaz web estándar que se pueda instalar y administrar en cualquier equipo que cuente con herramientas de software libre tales como servidor Apache con PHP y MySQL instalado.

Justificación:

En la actualidad el crecimiento de usuarios de Internet así como el incremento en el nivel técnico de conocimientos del usuario promedio ha generado una nueva cultura de estandarización. Existen muchos paquetes de software que integran aplicaciones enfocadas a un sólo fin, por ejemplo los antivirus. Actualmente un sistema de protección para PCs integra el antivirus tradicional en conjunto con alguna aplicación *antispyware*, un *firewall* e incluso utilerías para el mantenimiento del sistema. Lo mismo pasa con muchas páginas de Internet: recientemente se han popularizado paquetes que instalan un foro, un *blog* o un álbum multimedia, siempre y cuando se tengan algunos requisitos mínimos en el servidor.

Hablando de los *blogs* y foros son muchos los paquetes que hay en Internet que se pueden instalar en un servidor que cuente por lo menos con Apache, PHP y MySQL. La gran mayoría son *Open Source* o código abierto, vienen comprimidos en un sólo fichero y para instalarlos sólo es necesario modificar uno o muy pocos archivos y descomprimirlos en el servidor, correr o abrir algún archivo .*php* de instalación y listo.

También la instalación de servidores ha tenido esta tendencia. Se estima que el 90% de las PCs actuales utilizan el sistema operativo Windows en alguna de sus versiones y tomando en cuenta lo anterior se han creado paquetes de instalación para esta plataforma que incluyen herramientas de software libre para montar un servidor web muy completo de manera muy sencilla. Los más conocidos son:

APPSERV

AppServ es un paquete de software que permite instalar en nuestro equipo, bajo el sistema operativo Windows, en pocos minutos y sin dificultad, los siguientes programas:

a) Apache *WebServer*
El servidor HTTP Apache es un servidor HTTP de código abierto, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual. Es el de mayor uso actualmente.

b) Apache *Monitor*
Monitorea la actividad del servidor Apache y guarda bitácoras de entradas y salidas de usuarios, descargas, sesiones, errores, etc.

c) PHP *Script Language*
Herramienta de programación web, muy utilizada para crear sitios de Internet dinámicos, así como para desarrollar aplicaciones más avanzadas.

d) MySQL *Database*
Manejador de bases de datos muy estable y seguro.

e) phpMyAdmin *Database Manager*
Administrador web de bases de datos que utiliza PHP para operar en las mismas.

f) PHP-Nuke *Web Portal System*
Sistema para la elaboración de un portal o CMS con el cual se tiene la posibilidad de diseñar, implementar y modificar un sitio en Internet, pero lo que es más importante, incluye múltiples áreas y módulos que le permitirán administrar usuarios, descargas, novedades, encuestas y estadísticas del sitio.

Todas las aplicaciones anteriores son software libre. AppServ se encuentra disponible en su página oficial desde donde te puede descargar la última versión. Este programa esta protegido por la licencia Pública General GNU/GPL.

PHPTriad

Es una completa herramienta de desarrollo PHP que incluye todas las aplicaciones necesarias para trabajar con este lenguaje de programación en entorno Windows.

PHPTriad instala en el sistema PHP, MySQL, Apache y PHPMyAdmin, creando el ambiente de trabajo necesario con las herramientas básicas para programar en PHP.

Cuenta con las mismas características que Appserv con la diferencia de las versiones de las aplicaciones que cada paquete incluye.

Por todo lo anterior se puede concluir que la tendencia hacia la estandarización de paquetes de instalación y el gran crecimiento de usuarios en Internet, así como la necesidad cada día mas grande de las pequeñas empresa de introducirse en la red se puede deducir que la creación de un paquete que instale una interfaz web de registro de personal utilizando software libre tendría una gran aceptación.

Diseño.

Tomando el modelo clásico de la estructura de una micro o mediana empresa podemos dividir a las personas que trabajan en ella en diferentes niveles, los cuales determinaran sus sueldos, tipo de prestaciones, entre otras cosas. Lo mínimo que se debe saber sobre los empleados de una empresa son sus datos generales, los cuales constituyen la parte principal de todo registro de personal por lo que también lo será en esta ocasión.

Si se toma en cuenta la amplia gama de tamaños y tipos de empresas que desean entrar por primera vez en el mundo de la WEB y al mismo tiempo comenzar su migración de datos de la forma tradicional hacia los medios electrónicos, el sistema a desarrollar debe ser inicialmente básico pero con la capacidad de modificación y actualización suficiente que le permita convertirse en un sistema mucho mas robusto y completo al mismo tiempo que debe ser altamente configurable al gusto y necesidad de la empresa que lo emplee.

La instalación de éste sistema debe ser muy sencilla, tan sólo debe consistir en pocos pasos simples que se realicen sin grandes conocimientos técnicos sobre el mismo. La principal forma de introducción de este tipo de software en el mercado tiene que ver con la facilidad que ofrezca al manejarlo para que su asimilación en el entorno de trabajo sea más rápida.

Así mismo la administración del sistema debe ser amigable y sencilla, en este punto se podría utilizar alguna aplicación externa que nos facilite el manejo de los archivos y de las bases de datos existentes en el sistema. También facilitaría la instalación de la interfaz misma ya que no dependería directamente una de la otra.

Finalmente se establece que las herramientas de software que se utilizaran para el desarrollo de la interfaz serán: el servidor WEB Apache, Lenguaje HTML y PHP y el sistema de gestión de bases de datos MySQL. Todas las herramientas fueron elegidas en base a su estabilidad, seguridad y difusión en el mercado, además de que han demostrado gran flexibilidad individual así como integración en conjunto.

Construcción.

Aprovechando la capacidad de PHP para crear páginas dinámicas y mantener la seguridad y confidencialidad sobre el código de la página se escribirán todas las páginas de la interfaz en este lenguaje.

La creación de la base de datos se hará manualmente pero se rellenará y configurará mediante una rutina en la página *config.php* y a partir de este punto toda información se guardará en ella aunque visualmente su presentación cambie. Esto es que para cada actividad de la interfaz se tendrá una página PHP que mediante consultas a la base de datos utilizando MySQL se podrán agregar, modificar y eliminar los registros del personal dependiendo del nivel de acceso que se tenga.

La página principal, llamada *index.php*, invoca el tema o combinación de colores actual y en base a éste la interfaz puede variar visualmente. Dicha página tiene la función de enlazar la portada de la interfaz de manera dinámica y así se tiene la opción de cambiar dicho enlace en cualquier momento, por ejemplo si se necesitara dar mantenimiento al sistema o si se decidiera cambiar de ubicación el servidor, etc.

Finalmente los temas o pieles del sitio entero pueden ser modificados o agregar nuevos, al mismo tiempo pueden variar la forma de presentar la información. Con ésta característica se puede buscar la mejor manera de presentar la interfaz, hacerla mas completa o minimalista según sea el caso sin tener la necesidad de modificar alguna parte de los datos que se almacenen.

Instalación.

Dependiendo de la función del servidor, es decir, de la manera en que es utilizado, puede contar con administradores de archivos previamente instalados. Si no se cuenta con ellos o se instalará el servidor mismo desde cero es recomendable utilizar algún software que instale todas las herramientas necesarias. Si sólo se necesita un administrador de archivos para el servidor también es recomendable instalarlo antes de continuar con la configuración del sistema. Los requisitos mínimos para la interfaz son:

- Servidor WEB Apache 2.0 o posterior.
- PHP 4.4 o posterior
- MySQL 5.0 o posterior

Toda la interfaz es contenido en un archivo comprimido. Los pasos de instalación se escribirán en la página web desde la cual se puede descargar el sistema y también se incluye una copia de las instrucciones dentro del archivo comprimido (figura II.1).

También es necesario crear una base de datos nueva que utilizará el sistema, tan solo se debe crear vacía y el instalador la rellenará.

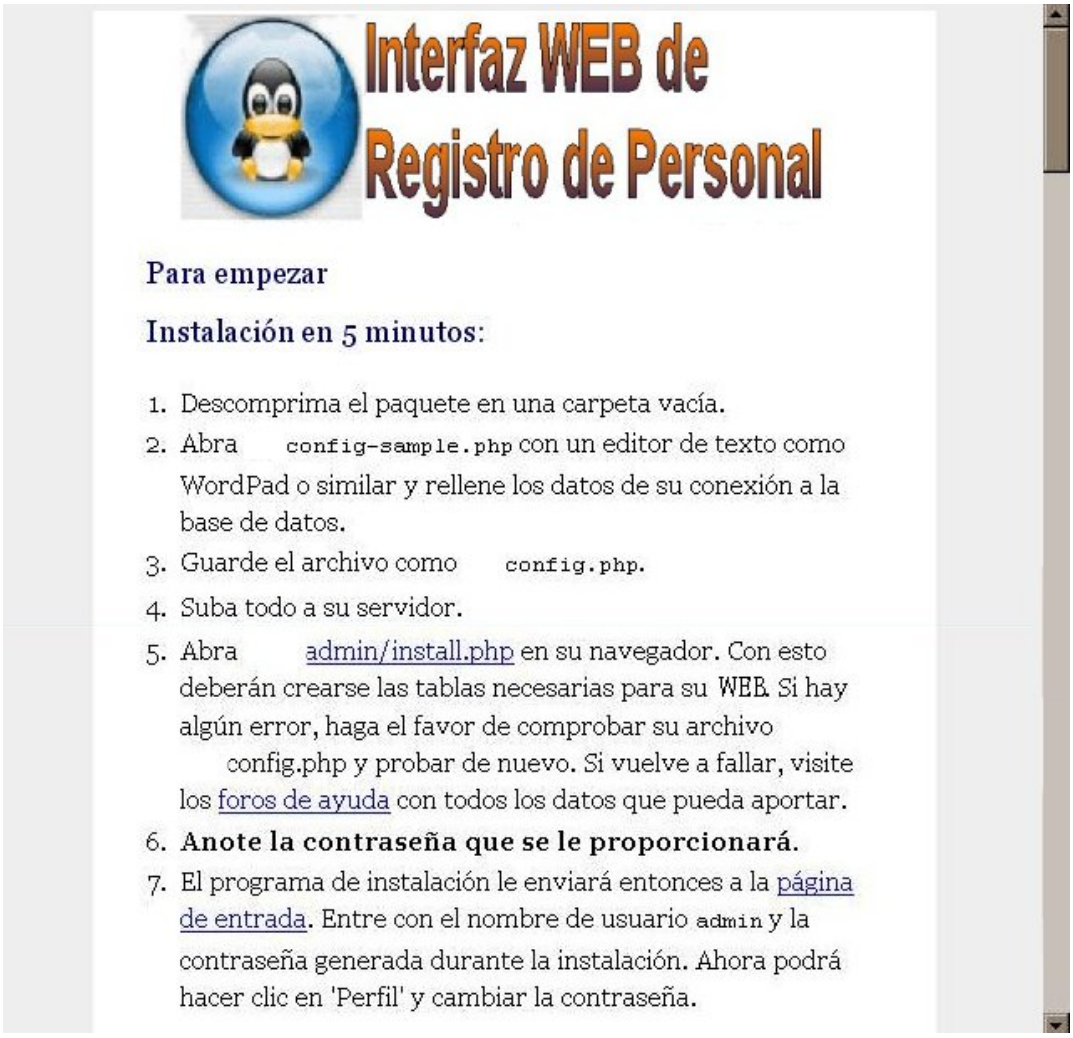


Figura II.1: Vista de la página WEB con las instrucciones de instalación

El paso número 2 se refiere a cambiar los datos del archivo *config-sample.php* con un editor de texto básico (figura II.2).

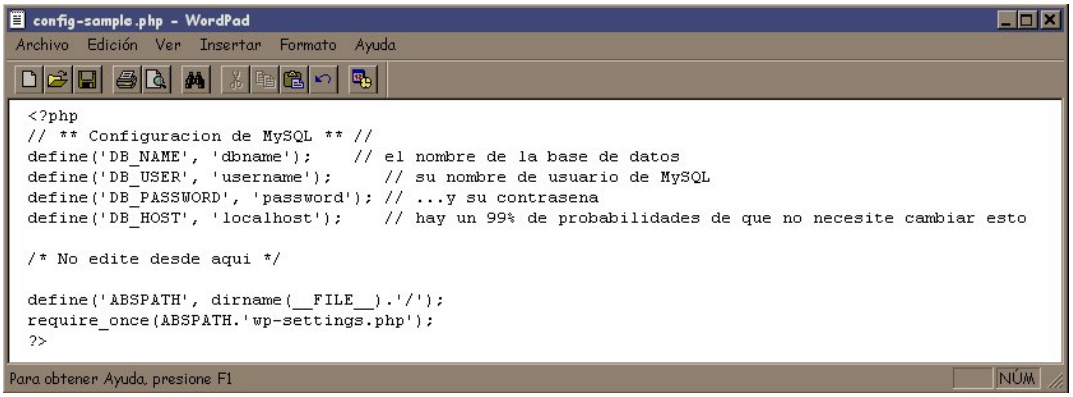


Figura II.2: Edición del archivo *config-sample.php*

En el paso número 5 se solicita abrir la página *install.php* para generar las tablas de la base de datos (figura II.3).

Figura II.3: Vista de la página de bienvenida del instalador

Continuando con la instalación, en el segundo paso se solicitan un par de datos más (figura II.4).

Figura II.4: Primer paso del proceso de instalación

Al terminar este paso se habrá acabado la instalación del sistema. Internamente ya se habrán generado las tablas necesarias para trabajar y la interfaz se encuentra lista para ser rellenada con los datos de los empleados de la empresa.

La última página de la instalación nos muestra el siguiente resultado:

Figura II.5: Segundo paso del proceso de instalación

Como se observa en la figura II.5, el sistema entregará una contraseña aleatoria que servirá para ingresar en el mismo (figura II.6).

Figura II.6: Vista de la página de acceso

Esquema interno.

La interfaz contará con cuatro tablas básicas que estarán relacionadas entre sí: datos, sueldos, antecedentes y prestaciones (figura II.7).

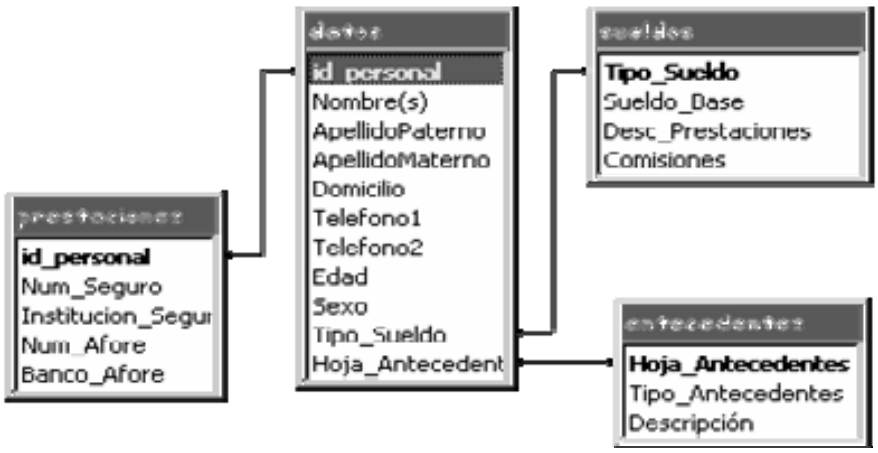


Figura II.7: Estructura interna de la base de datos

La tabla **datos** almacena los datos generales que se deben tener acerca de cualquier miembro del personal en un entorno de trabajo. Es la parte central de la base de datos ya que esta relacionada con las otras tres tablas. El campo *id_personal* es la llave primaria, de tipo entero y se genera automáticamente de forma incremental. Los campos *Nombre(s)*, *ApellidoPaterno*, *ApellidoMaterno*, *Domicilio*, *Tipo_Sueldo* y *Hoja_Antecedente* son obligatorios.

La tabla **sueldos** almacena las cantidades que se le pagan a cada miembro del personal mediante un sistema muy sencillo. El campo *Tipo_Sueldo* es la llave primaria, de tipo entero y se genera automáticamente de forma incremental.

Considerando que es muy factible que en las pequeñas empresas el sueldo base sea el mismo para la mayoría de los empleados, así como los descuentos derivados de las prestaciones, el tipo de sueldo permite que no se muestren cantidades a menos que se realice una consulta más exacta. El campo Comisiones se puede utilizar para ajustar los sueldos en casos especiales.

La tabla **antecedentes** almacena una relación de antecedentes en base a los miembros del personal. El campo *Hoja_Antecedentes* es la llave primaria, de tipo entero y se genera automáticamente de forma incremental. Inicialmente se tendrán los tipos de antecedentes jurídicos, de salud e internos, éste último haciendo referencia a problemas dentro de la empresa, ascensos o cualquier asunto concerniente al desempeño de su trabajo.

La tabla **prestaciones** sirve para guardar números de referencia de las prestaciones que el miembro del personal tiene, tales como números de afiliación al seguro social, clave del numero de su afore, folio de crédito hipotecario, etc. El campo *id_personal* es la llave primaria, de tipo entero y se genera automáticamente de forma incremental.

Escalabilidad

El propósito de presentar una base de datos mínima es que sea suficiente para cualquier empresa pequeña que maneja un volumen pequeño de personal, o para partes de una empresa más grande (zonas de obreros, sucursales pequeñas, departamentos específicos de alguna organización, etc).

Todas las tablas pueden crecer y abarcar más datos del personal, así como se puede incrementar el número de tablas y agregar más relaciones entre las mismas.

Por ejemplo, en el caso de la tabla **prestaciones** se pueden agregar los costos de cada una y mediante una rutina del sistema actualizar el campo *Desc_Prestaciones* de la tabla **sueldos**.

Interfaz

Debido a que se montará en un servidor web, el sistema puede ser consultado en cualquier momento desde cualquier lugar. Por su naturaleza de almacenar datos personales se deben manejar claves de acceso en el servidor. Tales configuraciones se pueden realizar en la configuración de Apache.

Figura II.8: Vista de la interfaz del sistema

Una vez que se tiene acceso al sistema las consultas y modificaciones que se realicen se harán en tiempo real.

Portabilidad

Se pretende que el sistema se pueda implementar en cualquier servidor WWW Apache con PHP y MySQL instalados. Como se había mencionado antes, es muy sencillo levantar un servidor de estas condiciones en una PC con sistema operativo Windows, reiterando que son la arquitectura y plataforma más utilizadas actualmente. Y aún si se migrara a otra arquitectura de hardware, por ejemplo si se pagara un *hosting* en un servidor dedicado o a un sistema operativo distinto el sistema seguiría funcionando de la misma manera.

Difusión

Aprovechando que se trabaja con herramientas *Open Source* y las licencias actuales GNU/GPL, es posible inscribir el proyecto en sitios que apoyan y difunden paquetes de instalación gratuitos. Una vez liberado en Internet el proyecto podría atraer programadores y hasta compañías dedicadas a la creación de *suites* más robustas de las que este proyecto podría formar parte.

CONCLUSIONES

El sistema operativo Linux ha tenido una evolución muy importante que lo ha llevado a ser muy conocido y utilizado actualmente. Desde su concepción se ha caracterizado por su naturaleza de código abierto, características que le ha hecho ganar varios y valiosos colaboradores que trabajan en su desarrollo principalmente por causa de gusto personal y profesional.

Fundamentalmente dirigido al desarrollo de aplicaciones y de sí mismo, ha servido también para generar una cultura mundial de compartimiento, de colaboración, de ayuda y sobretodo de comunicación entre personas de distintos intereses, dedicadas a diferentes áreas y que en general sólo tienen en común la idea de desarrollar y aprender nuevas tecnologías.

Un defecto que se le puede atribuir a Linux es que su uso requiere conocimientos previos. A diferencia de otros sistemas operativos, Linux actualmente no esta dirigido a usuarios en el mundo de la computación. Es muy común el uso de comandos para realizar las operaciones básicas en el sistema, además de que el usuario que se inicia en el uso de un sistema operativo no tiene nociones de administración necesarias para dar mantenimiento mínimo al equipo, como ocurre en otros sistemas operativos que administran los recursos del equipo y liberan al usuario de tales funciones.

Por otro lado es importante saber que el avance continuo de las tecnologías y el software que implementa Linux advierten que es un sistema operativo con mucho futuro. Las distribuciones actuales incluyen muchas utilerías del sistema, programas para diversos fines como educación, desarrollo, entretenimiento, etc.

Finalmente se hace mención a comandos y tareas básicas que se realizan en el sistema. Como se había indicado antes, el conocimiento previo es necesario para hacer un uso correcto de Linux, y por lo tanto es primordial iniciar siempre con la práctica para su mejor asimilación.

Linux puede ser instalado en prácticamente cualquier arquitectura de hardware. Los requerimientos de instalación varían para cada distribución.

El perfil de administrador del sistema requiere principalmente compromiso y conocimiento para poder desarrollar todas las actividades propias del puesto. Una correcta administración generará un sistema estable, usuarios satisfechos y un registro adecuado de eventos y actividades.

Tomando en cuenta que el administrador tiene a su cargo las actividades principales del sistema, una guía básica de instalación es necesaria para comenzar. La instalación del sistema no es tan sencilla como lo es con otros sistemas operativos. Se deben establecer características físicas del equipo, instalar el sistema y posteriormente configurarlo, de forma tal que una instalación completa difícilmente se repite en otro equipo.

Una vez instalado el sistema se comienza a administrarlo, para lo cual se necesita saber comandos orientados al manejo de usuarios, al monitoreo del sistema y del equipo, a la configuración del hardware y también saber que archivos brindan información de utilidad.

El presente informe encapsula la parte teórica del diplomado que cursé y que me brindó la oportunidad de conocer profundamente las tecnologías actualmente más utilizadas y estables para el desarrollo de sitios web dinámicos, comenzando con el conocimiento del sistema operativo Linux y sus ventajas principales como ser un sistema multiusuario y multitarea lo que significa que aprovecha al máximo las capacidades del equipo.

Otro punto muy importante es que algunas distribuciones de Linux cuentan con versiones de Apache, PHP y MySQL precargado por lo que el proceso de adquirir e instalar el software no es necesario. En particular si se habla de Slackware se trata de una distribución estable y muy completa, famosa por su gran parecido al modo consola de UNIX. Si a todo lo anterior se suma la cultura *Open Surce* que apoya el desarrollo y distribución de manera gratuita de muchos productos, entre los cuales se encuentran Apache, PHP y MySQL, se tiene un ambiente muy conveniente para los desarrolladores.

Finalmente puedo decir que el diplomado ha cumplido todas mis expectativas y fue más allá de lo que esperaba. Además de los conocimientos teóricos adquiridos, el desarrollo del proyecto me confirma que todo lo que se me enseñó forma en conjunto una poderosa herramienta para mi desarrollo como profesionista en un futuro cercano. Para muchas personas comenzar su formación en esta materia resulta difícil por la discontinuidad o falta de coherencia que se puede dar si se toman cursos individuales, mientras que para mi fue mucho más sencillo comprender y asimilar cada tema del diplomado en el orden y de la forma en que se me impartió.

FUENTES DE INFORMACION

1. http://www.php.net

2. http://www.apache.org

3. http://www.wikipedia.org

4. http://www.w3schools.com

5. http://www.mysql-hispano.org/

6. http://www.linuxiso.org/

7. http://www.java.com/es/